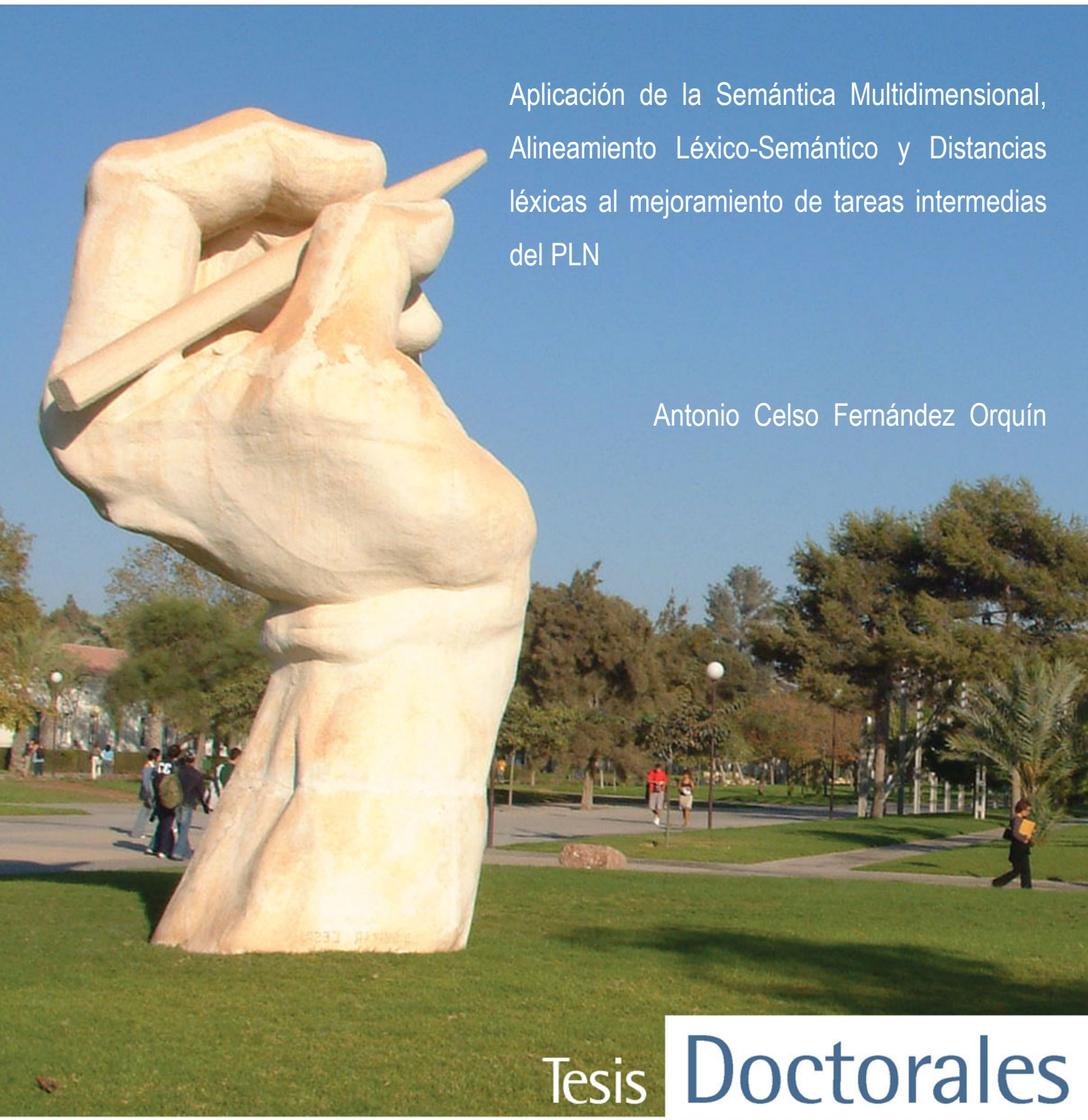
Universitat d'Alacant
Universidad de Alicante

Aplicación de la Semántica Multidimensional, Alineamiento Léxico-Semántico y Distancias léxicas al mejoramiento de tareas intermedias del PLN

Antonio Celso Fernández Orquín

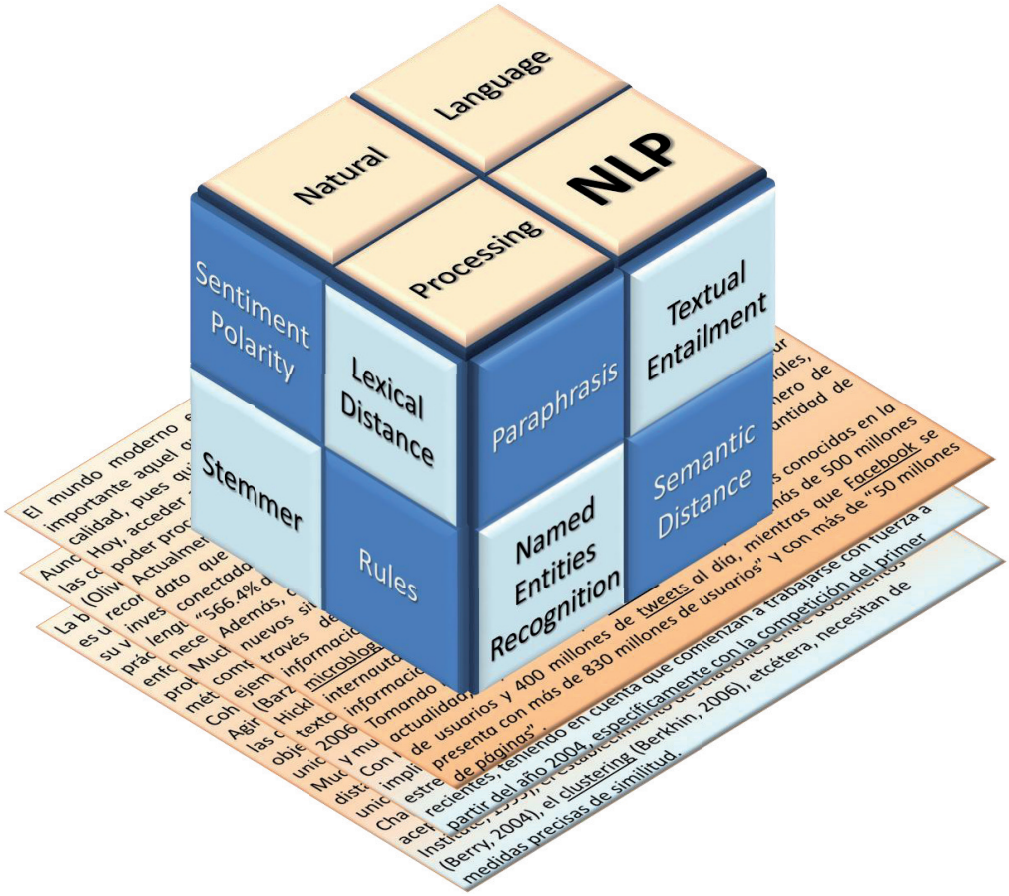


Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Universitat d'Alacant
Universidad de Alicante



Universitat d'Alacant
Universidad de Alicante

Aplicación de la Semántica Multidimensional, Alineamiento Léxico-Semántico y Distancias léxicas al mejoramiento de tareas intermedias del PLN.

Disertación

Presentada al Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante en cumplimiento parcial de los requisitos del título de Doctor en Ciencias.

Autor: Antonio Celso Fernández Orquín

Directores: Dr. Rafael Muñoz Guillena

Dr. Yoan Gutiérrez Vázquez

Alicante, Enero 2014

A mi hija y mi esposa

Agradecimiento

En primer lugar, y no por seguir un orden de importancia, quiero agradecer a la Universidad de Alicante por la oportunidad que me ha brindado de sentirme parte de esta maravillosa institución. También es preciso reconocer que desarrollo de esta tesis ha sido sustentada en gran medida gracias al los proyectos TESMES 2.0 (TIN2009-13391-C04) y ATTOS (TIN2012-38536-C03-03).

Continuaré, sin diferencia alguna por la posición en que los nombres aparezcan, quiero agradecer a alguien que hizo posible no solo que tuviéramos la oportunidad de viajar a España para hacer nuestro período lectivo, sino también de regresar luego a defender el DEA. Alguien que ha guiado este colectivo por caminos de éxito y por eso hoy ostenta el logro de dirigir esa maravillosa universidad. Gracias por las cosas que aprendí contigo en clases, hoy por estar siempre dispuesto a ayudarnos. ¡Muchísimas gracias Manuel Palomar Sanz!

Debo agradecer que sin conocerme me ofreció su apoyo en todo sentido, alguien que con el tiempo me ha demostrado una sincera amistad, que estoy seguro perdurará para siempre. Le doy las gracias por permitirme trabajar con él y guiar mi investigación. Muchas gracias, no solo por la ayuda científica, si no también por todas las cosas en las que me has extendido tu mano durante todos estos años. Te doy las gracias por confiar en mí y arriesgarte a ser mi director, espero no defraudarte. ¡Muchísimas gracias Rafael Muñoz Guillena!

Hay dos personas que no puedo dejar de agradecerle por hacerme la vida más placentera cuando en Alicante me encontraba apabullado por tantos contenidos nuevos para mí, que en ocasiones llegaron hasta obstinarme. Con ellos pude despejar y pasar momentos inolvidables, los que me permitieron enfrentar luego todo con nuevas fuerzas. Les estoy eternamente agradecido por brindarme su amistad ¡Muchísimas gracias Patricio Manuel Martínez Barco y María Paloma Moreda Pozo!

Gracias a esta persona logré mis primeros resultados en el Reconocimiento de Entidades, pues él me donó sus conocimientos sobre Máxima Entropía y también su trabajo. Muchas gracias por la tesis que me dedicaste, me ha servido de mucho. También te agradezco por soportar mis continuas interrupciones en medio de la preparación de tu trabajo. Gracias por todo lo que me aportaste para escribir mi primer artículo sobre Reconocimiento de entidades. ¡Muchísimas gracias Armado Suárez Cueto!

Gracias a esta persona logré muchas veces evacuar mis dudas inmensas sobre Linux, C, C++, los editores de código y todas las cosas nuevas a las que me enfrentaba. Si no hubieras estado siempre dispuesta a ayudarnos hubiera sido muy difícil para nosotros lograr nuestro objetivo. ¡Muchísimas gracias Sonia Vázquez Pérez!

Agradecimiento

Gracias a su bondad, su paciencia y amistad logramos aprender muchas cosas importantes para nosotros. Gracias por cada una de las veces que te molesté con las más diversas preguntas. Sin tu ayuda me hubiera echado mucho rato intentando buscar cosas que siempre sacabas como de debajo de la manga cuando te las preguntaba. Haber podido compartir contigo todo este tiempo ha sido un privilegio, te agradezco tus explicaciones sobre C++, búsqueda de respuestas y muchas cosas más, también por nuestras charlas para despejar de tanto trabajo. ¡Muchísimas gracias David Tomas Díaz!

No puedo dejar de mencionar a quién nos hizo ver muchas de las maravillas del PLN, calmado, como siempre, nos supo hacer descubrir muchas cosas que nunca hubiéramos podido imaginar. Gracias por poner tu experiencia en nuestras manos, por todos los artículos que me diste para leer y los recursos que pusiste en nuestras manos. ¡Muchísimas gracias Antonio Ferrández Rodríguez!

Necesito una cuenta, desde mi máquina no me puedo conectar a los servidores, necesito los instaladores de un programa, tengo problemas con Linux y las conexiones, etc., etc. Todas estas molestias tuvieron que soportar estas dos grandes personas, su disposición a ayudar en todo momento fue impresionante, gracias también por las charlas que tuvimos en nuestros desayunos. Sin su ayuda no hubiera podido hacer mucho. ¡Muchísimas gracias Miguel Ángel Baeza Ripio y Miguel Ángel Varó Giner!

No he conocido a nadie tan dinámico y energético como él. Gracias a esta persona logré entender algo sobre los almacenes de datos y los proceso ETL. También aproveché mucho sus charlas sobre cómo debía encaminar mi trabajo. Compartir con él siempre fue alentador. ¡Muchas gracias Juan Carlos Trujillo Mondéjar!

Sin su ayuda no se hubiera materializado nada, el viaje a España, los trámites, el papeleo, etc. Te agradezco por todos los trabajos que pasaste para conseguir que llegáramos por fin a la Universidad de Alicante. Gracias por todas las horas que debes haber restado al placer de estar con tu familia, debido a nuestros disímiles problemas. ¡Muchas Gracias Jesús Perales Cortés!

Tu capacidad de trabajo y la gran cantidad de éxitos que has alcanzado en tu vida hicieron que escuchara atentamente todos tus consejos. Quiero agradecerte por las clases que pasé contigo y por todas las cosas que aprendí contigo. También tu amistad ha significado mucho para mí, te agradezco infinitamente por preocuparte por el desarrollo de mi trabajo y por la ayuda que has significado para mí. ¡Muchas gracias Andrés Montoyo Guijarro!

Hay quienes no intervienen en tu vida científica, pero son tan importantes que sin ellos no es posible hacer ciencia. Esta persona ha sabido ser más que un amigo un hermano. Sin él hubiera sido

extremadamente difícil mi estancia en Alicante. Te agradezco por ofrecerme tu amistad, por tu hospitalidad, por permitirme compartir con tu familia la que hoy considero como mía. Por ti conocí a muchas personas importantes para mí. No puedo expresar aquí todas las cosas que tendría que agradecerte porque son mucha. Gracias mi hermano. ¡Muchas gracias José Ramón Lillo Beviá!

Hay quienes te ayudan una vez en la vida y son importantes, hay otros que te ayudan varias veces y esos son mejores, pero hay otros que te ayudan durante toda tu vida y esos son imprescindibles. Doy gracias por poder contar con tu amistad y ayuda incondicional para todas las cosas que me hicieron falta desde que nos conocimos. ¡Muchas gracias Roger Pérez Chávez!

Hay cosas que aunque las sepas o te las estudies necesitas debatirlas, confrontarlas y así llegar a tus conlcusiones. En este sentido quiero agradecer a esta persona que simpre supo ayudarme a entender las peculiaridades del la Inteligencia Artificial y en especial del Aprendizaje automático. También por su ayuda con los materiales para estudiar y por las aclaraciones de mis dudas en esta complicada temática. ¡Muchas gracias José Ignacio Abreus Salas!

Quien siembra algo una vez en su vida, algún día recoge los frutos. Yo tuve la oportunidad de sembrar muchos árboles que hoy me han dado sus frutos y más que eso su amistad y la posibilidad de compartir las dificultades de la investigación entre muchos, para que el esfuerzo toque a menos. Sin ustedes no podría haber logrado tantas cosas importantes en este complejo mundo del PLN. La ayuda de todos y cada uno de ustedes no la olvidaré nunca. ¡Muchas gracias a nuestro grupo de investigación, en especial a Yaniseth García, Dayté Rodríguez, Josval Díaz, Abel Pérez, Iván García, Alexander Chávez, Hector Dávila, Armando Collazo, Ángel Cobarrubias, Renier Pérez, Andy González!

Para el final he dejado este personaje, no porque esta posición signifique nada en la importancia de su ayuda, todo lo contrario. Sin ti mi hermano no hubiera podido hacer absolutamente nada. Gracias a ti se han podido lograr la mayoría de los éxitos que hemos tenido en estos últimos tiempos. Gracias por tu ayuda desinteresada, y digo desinteresada porque sé que no me has ayudado para retribuir la ayuda que yo pude darte, pues la tuya es mil veces mayor que la que yo pude ofrecerte. Yo sé que me has ayudado porque eres una persona especial y con un corazón muy grande. Muchas gracias por compartir conmigo y ayudarme a desenredarme y encaminar este trabajo que había dejado dormido por tanto tiempo. Muchas gracias por todo lo que has tenido que hacer y por el tiempo que has robado a tu familia para hacer tuyos mis problemas. ¡Muchísimas gracias Yoan Gutiérrez Vázquez!

RESUMEN

El estudio de los mecanismos para intentar que las computadoras aprendan a entender el lenguaje humano, es un reto que involucra varias disciplinas y está inevitablemente ligado a un gran número de tareas para lograrlo. Este trabajo trata sobre el estudio y el intento por mejorar un grupo de tareas estrechamente vinculadas con el Procesamiento del Lenguaje Natural. Es imposible cubrir todas las tareas que comprende esta disciplina, por esto solo se estudian las que guardan relación con la búsqueda de similitud entre frases, el reconocimiento de entidades, la reducción de palabras a su tallo o raíz (stem) y por último la generación de patrones y reglas a través de lenguajes formales. Para lograr los objetivos propuestos se estructura el trabajo enfocado en dar respuesta a las siguientes preguntas de investigación:

1. ¿Es posible prescindir de los recursos para el análisis morfo-léxico, sintáctico y semántico en el reconocimiento de entidades, para con ello evitar los errores acarreados por los recursos dedicados a estas tareas y, aún así, obtener resultados a la altura de los alcanzados a nivel internacional?

   Para dar respuesta a esta pregunta, en el Capítulo 1 se hace una breve introducción a la problemática que motiva la investigación asociada al reconocimiento de entidades. Luego en el Capítulo 2 se comentan todos los referentes teóricos asociados a este proceso y se citan los trabajos relacionados en este campo. Finalmente en el Capítulo 3, específicamente en el epígrafe 3.7, se hacen una serie de experimentos para validar la premisa de partida, al final del Capítulo 3 se exponen las conclusiones a las que se llegan después del análisis de los experimentos. Más tarde en el Capítulo 4 se da la respuesta a esta pregunta.

2. ¿Puede lograrse un método que permita abstraer las complejidades de los lenguajes formales en la creación de reglas y patrones de extracción, a partir de expresiones regulares para su utilización en diferentes tareas del PLN?

   En este caso -al igual que en el anterior- el Capítulo 1 recoge, a través del epígrafe 1.2, una introducción a esta interrogante, donde se señalan las principales dificultades que intentan resolverse en este trabajo. Seguidamente en el Capítulo 2 se exponen los aspectos teóricos que dan fundamento científico a la propuesta de solución, la que es explicada en detalles en el epígrafe 3.8 del Capítulo 3. Al final de este capítulo se hacen las conclusiones parciales a las que se llega y la respuesta a esta pregunta se realiza en el Capítulo 4.

3. ¿De qué forma mejora a la distancia de edición, utilizada en la medición de similitud, de manera que al comparar palabras con diferencias en la raíz -que provocan cambio en el significado- sean penalizadas consecuentemente?

4. ¿Se podrá utilizar la modificación de la distancia de edición en la generación automática de familias de palabras y obtener resultados por encima de un 90% de precisión en esta tarea?

   Para solucionar estas interrogantes, en el epígrafe 2.6.1 se hace un análisis sobre un conjunto de métricas y se plantan todas las dificultades detectadas en este sentido. También se comentan los detalles de varios trabajos relacionados con la distancia de edición. En el epígrafe 3.1 del Capítulo 3 se explican las transformaciones propuestas a la distancia de edición, que dan respuesta a esta pregunta. Se justifican las mejoras obtenidas mediante los experimentos realizados y se hace el análisis de los resultados obtenidos. En el epígrafe 3.6.1 se comprueba la factibilidad de la utilización de la extensión realizada a la distancia de edición en la tarea del agrupamiento de familias de palabras. Al final del Capítulo 3 se recogen las conclusiones derivadas de la experimentación efectuada. La respuesta estas preguntas se pueden comprobar en el Capítulo 4.

5. Mediante la modificación de la distancia de edición, ¿se podrá obtener un stemmer independiente del lenguaje, a partir del agrupamiento de familias de palabras, que obtenga resultados superiores al 95% de precisión, cobertura y exactitud?

   Luego de obtenida la modificación de la distancia de edición y la comprobación de que es factible su utilización para el agrupamiento de familias de palabras, se comprueba que a partir de estas combinación se puede obtener el stem de las palabras y alcanzar una precisión superior a un 95%. Los detalles para validar esta pregunta se recogen en los epígrafes 1.4, 0, 3.6. También en esta ocasión se realizan los respectivos experimentos de comprobación y se analizan los resultados. Igual que en las demás preguntas se puede ver la respuesta a esta pregunta en el Capítulo 4.

6. ¿Qué influencias tiene la utilización de un alineamiento léxico y semántico desde una perspectiva multidimensional en reconocimiento de la similitud textual?

   Para corroborar la veracidad de esta interrogante, primeramente, en el epígrafe 1.1 se hace un recuento sobre la tema de la similitud. Más adelante, en el epígrafe 2.6 se exponen los referentes teóricos, tanto de la similitud léxica como semántica. Y como elemento necesario para poder comprender esta temática, se describen en el epígrafe 2.6.4 los detalles sobre el alineamiento como técnica para la determinación de similitud. Como en todos los casos

anteriores, en el Capítulo III se describen los métodos de alineamiento implementados y se describen y analizan los experimentos realizados. Para ver finalmente la respuesta que se ha dado a esta pregunta, se puede acudir al Capítulo 4 en el 4.1.

7. ¿Qué influencia tiene la polaridad sentimental en la determinación de la Implicación Textual?

   Esta pregunta inevitablemente tiene que pasar por la rememoración de los conceptos de Polaridad Sentimental e Implicación Textual, así como de los trabajos realizados en este sentido; para ello se utilizan los epígrafes 2.6.32.6.5. En el epígrafe 3.2 se presenta la metodología desarrollada, así como los experimentos para la determinación de la relación entre estos conceptos. Una respuesta a esta pregunta se expone en el apartado 4.1.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

All life is an experiment. The more experiments you make the better.

Capítulo 1 Introducción

El mundo moderno está colmado de información, cada día es más importante aquel que pueda procesar la mayor cantidad posible y con calidad, pues quien posee más información, posee más conocimiento. Hoy, acceder a la información no es el gran problema, tal vez sí lo es el poder procesarla eficientemente.

Actualmente, Internet cuenta con más de "*2405 millones de usuarios*"[1], dato que implica que más del 30% de la población mundial está conectada a la red de redes. Esto representa un crecimiento de un "*566.4% del 2000 al 2012*", según esta misma fuente.

Además, desde la aparición de la Web 2.0 (o Web social), se han creado nuevos sitios Web dónde los usuarios juegan un papel más activo, a través de los que pueden participar, interactuar e intercambiar información con otros usuarios (por ejemplo, foros, blogs, redes sociales, microblogs, etc.). Esto ha dado lugar a que aumente el número de internautas a un ritmo exponencial, así como también la cantidad de información y contenidos que se publican en la Web.

Tomando las estadísticas de dos de las redes sociales más conocidas en la actualidad, se puede ver que Twitter "*cuenta ya con más de 500 millones de usuarios y 400 millones de tweets al día*[2]*, mientras que Facebook se presenta con más de 830 millones de usuarios*"[3] y con más de "*50 millones de páginas*"[4]. Estos son sólo dos ejemplos representativos de dos redes sociales. Si a ello se le suman el resto de sitios Web, incluyendo otro tipo de redes sociales, páginas Web, enciclopedias, blogs, foros, contenido multimedia, etc.; se encontrarán algo más de "*3800 millones de páginas Web indexadas*"[5]. Esto reafirma que existen muchos procesos en la vida del hombre en los que la información juega un papel esencial, por ejemplo, la toma de decisiones.

Sin embargo, el principal inconveniente de toda esta gran cantidad de información es la complejidad en cuanto a su interpretación, sobre todo si el usuario desea obtener información concisa y de calidad acerca de un tema específico. Primeramente, la va a encuentra en diferentes fuentes y de distinta naturaleza, también en muy variados idiomas. Estos factores, junto a la

[1] http://www.internetworldstats.com/stats.htm (Septiembre 2013)

[2] http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/ (Septiembre 2013)

[3] http://www.internetworldstats.com/facebook.htm (Septiembre 2013)

[4] http://www.statisticbrain.com/facebook-statistics/ (Septiembre 2013)

[5] http://www.worldwidewebsize.com/ (Septiembre 2013)

redundancia, disímiles opiniones y hechos contradictorios que aparecen a diario en la Web, hacen que los usuarios inviertan mucho más tiempo de lo deseado buscando la información de su interés.

En parte, esto se debe a que para hallar información de calidad es necesario leer -minuciosamente- grandes y muy diversas fuentes, intentando corroborar que está la información desea. En este sentido, cuando se habla de calidad de la información, se está haciendo referencia a información contrastada que se corresponde, en gran medida, con el objetivo deseado en la búsqueda.

Pero por otra parte, desgraciadamente, si bien es verdad que hoy se tiene más acceso a la información, también se conoce que la mayor parte de ella se encuentra de forma no estructurada o semi-estructurada. Esta es una de las causas que hacen muy difícil el procesamiento automático de la información.

Para intentar resolver esta problemática, es imposible pretender que este proceso quede a cargo de los humanos. Desde hace mucho tiempo las tareas más complejas y rutinarias se vienen encargando a las computadoras; con su ayuda se pude lograr una mejoría en el proceso de análisis de la información.

Para salvar esta situación, así como un modo de reducir el tiempo invertido por los usuarios en el análisis de grandes cantidades de información, está el uso de las Tecnologías del Lenguaje Humano (TLH). Estas tecnologías son fundamentales para manejar información de manera eficiente y efectiva, pues se encargan de procesar el lenguaje de forma automática. Esta área de investigación es una subdisciplina de la Inteligencia Artificial, la que investiga y formula mecanismos computacionalmente para facilitar la interrelación hombre-máquina, permitiendo una comunicación mucho más fluida y menos rígida que los lenguajes formales (Moreno, Palomar et al., 1999).

Con este objetivo, han surgido una gran cantidad de herramientas y recursos desarrollados en los últimos años, los que han permitido mejorar los procesos de búsqueda, recuperación y extracción de información (Allan, Croft et al., 2012; Dalvi, Cohen et al., 2012), clasificación de textos (Wang y Manning, 2012; Iglesias, Seara Vieira et al., 2013; Zhang, Marin et al., 2013), detección y minería de opiniones (Mihalcea, Banea et al., 2012; Montoyo, Martínez-Barco et al., 2012; Thelwall y Buckley, 2013), o síntesis de información (Zheng y Yu, 2012; Chong Tat Chua y Asur, 2013; Christensen, Mausam et al., 2013; Saggion y Poibeau, 2013), así como los procesos intermedios involucrados en cada una de estas tareas, tales como el análisis semántico (Gutierrez, 2010; Li y Joshi, 2012; Gutiérrez, Castañeda et al., 2013), elementos clave en el intento de interpretar el lenguaje humano.

Otro elemento, primordial y de gran importancia es la interpretación de lenguaje humano. Se sabe de la complejidad que tiene -incluso para los propios humanos- lidiar con la adquisición del lenguaje en cada uno de sus componentes: fonológico, morfosintáctico, léxico-semántico y pragmático. Elementos como la polisemia, la ambigüedad (incluyendo todos sus tipos) y otros, hacen que esta tarea sea todo un reto.

Atendiendo a esto, tal vez el obstáculo mayor que debe franquear una computadora a la hora de interpretar el lenguaje humano, es el hecho de que los humanos tienen un conocimiento amplio del medio circundante; por lo que no necesita de un lenguaje demasiado explícito para poder interpretar la información que reciben. Por ejemplo, a la pregunta: ¿hay agua en la nevera?, cualquier humano inferiría que le están preguntando si hay un embase en la nevera, que contiene agua y que esa persona desea tomarla, o al menos cerciorarse de que la hay. Esta, es información del contexto que no se necesita explícitamente en la conversación, el humano es capaz de inferirla.

Sin embargo, ante una pregunta como la anterior una computadora no podría discernir si lo que se quiere es saber si hay agua encima de la nevera, o tal vez dentro. A su vez, le costaría mucho trabajo inferir que la persona desea tomarla, o si solo quiere saber si hay agua para tener esa información presente.

Aunque no es una tarea imposible, sí es bastante difícil implementar que las computadoras interpreten el lenguaje humano. Al respecto, en (Oliveira, Nunes et al., 1998) se plantea: "….*Hoy sabemos que el reconocimiento de patrones aún es un gran desafío para los investigadores de visión artificial; lo mismo sucede con la comprensión del lenguaje humano por una máquina*". Tal es así que, para lograrlo son necesarios los esfuerzos aunados de varias disciplinas.

Muchos de estos elementos, absolutamente necesarios para la compresión del lenguaje humano a través de una computadora son, por ejemplo, el reconocimiento de la paráfrasis y la implicación textual (Barzilay y McKeown, 2001; Dagan, Glickman et al., 2006; Harabagiu y Hickl, 2006; Hickl, Williams et al., 2006), en general de la similitud entre textos (Resnik, 1995; Dagan, Lee et al., 1999; Hontoria, 2003; Venegas, 2006), han sido estudiado con profundidad por los autores referenciados y muchos otros que hoy continúan su estudio.

Con respecto a este tema, el reconocimiento de la paráfrasis y la implicación textual -por mencionar solo estas- son tareas vinculadas estrechamente a la búsqueda de similitud. Son temáticas bastante recientes, teniendo en cuenta que comienzan a trabajarse con fuerza a partir del año 2004, específicamente con la competición del primer PASCAL Recognizing Textual Entailment Challenge (Dagan, Glickman et al., 2005). Conocido en español como Reconocimiento de la Implicación Textual (en lo adelante solo RIT), consisten en determinar si la verdad de una frase es

igual o si implicaría la de otra. Por ejemplo: (frase 1) Antonio ha estado en Alicante en el 2005, (frase 2) Antonio ha estado en España en el año 2005. Es evidente que la verdad de la frase uno implica la de la dos, pues estar en Alicante -una ciudad de España- lógicamente implica que estuviera en España en esa fecha.

Esta situación, para los humanos es una tarea relativamente sencilla, para los ordenadores es algo que aún no está totalmente resuelto (Contreras, 2001; Elmasri y Navathe, 2009; Ontiveros y Pérez, 2013). Como bien se plantea en (Gelbukh, 2010), usualmente no es difícil para un programa encontrar una interpretación para un texto, o tal vez encontrarlas todas, lo difícil es elegir la correcta.

Por ello, intentando que las computadoras mejoren la forma en que interpretan el lenguaje humano, se han creado una serie de líneas de investigación en este sentido. Una de las tareas que con más fuerza se ha trabajado desde hace algunos años es la determinación de la similitud entre frases y en mayor escala entre documentos digitales.

Para esta tarea, como para muchas otras, son necesarias las llamadas tareas intermedias, que no son más que un conjunto de subtareas que permiten llevar a cabo el acometimiento de una tarea mayor o tarea objetivo. Se coincide con (Montoyo, 2009), cuando plantea: *"...se denominan tareas intermedias porque sus resultados únicamente proporcionan información lingüística y nada tienen que ver con lo que el usuario final demanda en última instancia*".

Sin embargo, estos tipos de tareas pueden ser simples o complejas, dependerá de la cantidad y complejidad de los procesos a realizar para lograrla. En ocasiones, cada una de las tareas intermedias se convierte en un procedimiento extremadamente complejo. Dichas tareas se llevan a cabo utilizando diferentes técnicas y aplicando diferentes conceptos teóricos de la lingüística, la estadística, la informática y la matemática.

Entre las tareas intermedias más utilizadas y a su vez más estudiadas se encuentran:

- Análisis morfo-léxico, sintáctico, semántico y pragmático;
- Desambigüación del sentido de las palabras (del inglés Word Sense Disambiguation (WSD));
- El estemizado[6] (del inglés Stemming) y la lematización[7] o lematizado (del inglés Lemmatizing);

[6] Obtienen el tallo o raíz de las palabras, no precisamente el lema.

[7] Obtienen la forma canónica de las palabras.

- Reconocimiento de Entidades Nombradas[8] (REN) (del inglés Named Entities Recognition (NER))
- Reconocimiento de entidades temporales[9];
- La detección de siglas y acrónimos;
- Determinación de similitud, incluyendo aquí el RIT y la paráfrasis.

Toda esta gama de técnicas, conceptualizaciones y tareas, implementadas para intentar que las computadoras entiendan el lenguaje humano, se unen bajo el nombre de Procesamiento del Lenguaje Natural (PLN), conocido en inglés como Natural Language Processing (NLP).

Dentro del PLN, aún cuando hay tareas más generales que precisan de tareas intermedias, también estas utilizan sub-tareas para cumplir su objetivo. Las mencionadas tareas del reconocimiento de la paráfrasis y la implicación textual, así como cualquier tipo de determinación de similitud entre textos, se ven afectadas por la precisión de los recursos que utilizan para lograr su objetivo final.

Con relación a esto, Graña plantea en su tesis doctoral (Graña Gil, 2002): "*El objetivo último que persigue el Procesamiento del Lenguaje Natural es el perfecto análisis y entendimiento de los lenguajes humanos. Actualmente, estamos todavía lejos de conseguir este objetivo. Por esta razón, la mayoría de los esfuerzos de investigación de la lingüística computacional han sido dirigidos hacia tareas intermedias que dan sentido a alguna de las múltiples características estructurales inherentes a los lenguajes, sin requerir un entendimiento completo*". Si bien es cierto que esta afirmación ya tiene 11 años, no es por eso que ha perdido su vigencia. Aún hoy se continúa trabajando por el mejoramiento de las tareas intermedias, elemento obligatorio para una correcta interpretación del lenguaje humano.

Los elementos antes expuestos han `motivado` a la experimentación y la búsqueda de soluciones de algunos de los elementos abordados anteriormente, se describen a continuación algunos conceptos importantes, así como las tareas que con más fuerza han sido estudiadas dentro de la temática de esta investigación.

1.1 La similitud Léxico-Sintáctica y Semántica

La búsqueda de la similitud ha sido aplicada a diferentes niveles. Desde el punto de vista de la magnitud o longitud de los elementos que se comparan, podría ser entre: caracteres, palabras, frases,

[8] Identifica y clasifica las entidades del texto como las de tipo persona, localización, Organización, etc.

[9] Reconoce y normaliza las expresiones de tiempo como: fechas, magnitudes de tiempo, etc.

párrafos o documentos. Cada uno de estos casos tiene una utilidad específica y han sido tratados utilizando una gran diversidad de técnicas.

En dependencia de si el análisis es de origen gramatical, las técnicas pueden ser Morfo-léxicas (solo se analiza la similitud entre los grafemas de los elementos comparados), Sintácticas (se compara la estructura sintáctica de los elementos analizados), Semánticas (se mide el grado de semejanza entre los sentidos de los elementos comparados).

La búsqueda de similitud, por ejemplo, entre caracteres de una cadena, es un problema importante estudiado en la Ciencia de la Computación. A su vez, este proceder tiene un gran número de aplicaciones en la vida práctica y en campos como el PLN. Los investigadores de esta área se han enfocado en diferentes aspectos y conceptos para el estudio de esta problemática. Uno de los más relevantes ha sido la generación de métricas del tipo distancia de edición, o medidas de similitud (Lee, 1999; Cohen, Ravikumar et al., 2003; Deza y Deza, 2006; Agirre, Cer et al., 2012; Agirre, Cer et al., 2013b; Korkontzelos, Zesch et al., 2013). Comúnmente, las distancias, han sido utilizadas para medir la proximidad entre dos objetos. Para el caso específico del PLN las distancias son calculadas entre unidades lingüísticas.

Muchas aplicaciones de tratamiento de la lengua descansan en estas distancias, llamadas también medidas de proximidad o lejanía de unidades lingüísticas de diversa índole (Lee, 1999; Hontoria, 2003; Chapman y Parkinson, 2006). Así, temas como la desambigüación de acepciones (Navigli, 2009), la detección de cadenas léxicas (St-Onge y Institute, 1995), el establecimiento de relaciones entre documentos (Berry, 2004), el clustering[10] (Berkhin, 2006), etcétera, necesitan de medidas precisas de similitud.

Según Rodríguez en (Martí, Fernández et al., 2003), estas técnicas se han utilizado en varias tareas de PLN, tanto de bajo o medio nivel como son: la generalización para la construcción de modelos de lenguaje (Llorens Piñana, 2000; Milone, Rubio et al., 2001) más precisos, la desambigüación del sentido de las palabras (Navigli, 2009; Gutiérrez, Fernández et al., 2010b), la extracción de restricciones de selección (Serra Sepúlveda, 2009), el análisis sintáctico (Carroll, Briscoe et al., 1998), la asignación de grupos preposicionales (Meya, 1991), la estructuración de los compuestos nominales, la resolución de la referencia (Muñoz, Martinez-Barco et al., 1999), la clasificación o la agrupación (Berkhin, 2006), la recuperación de información (Manning, Raghavan et al., 2008), la consulta a Internet, la extracción de información (Doan, Ramakrishnan et al., 2006; Wimalasuriya y

[10] Clustering: tarea de agrupar objetos más similares en un conjunto formando un grupo llamado cluster.

Dou, 2010), el resumen automático (Gupta y Lehal, 2010)o la traducción automática (Huenerfauth, 2003; Koehn y Knight, 2009), etc.

En consecuencia, también Budanitsk y plantea en (Budanitsky y Graeme, 2001) que el problema de formalizar y cuantificar una noción intuitiva de similitud tiene una larga historia en la filosofía, psicología, e inteligencia artificial, y a través de los años se han seguido muchas perspectivas diferentes.

De lo anterior, se puede apreciar la amplia gama de usos y la importancia que cobra el estudio de las métricas entre unidades lingüísticas para el PLN, destacándose también que es una temática de gran actualidad.

En términos de tendencias, un análisis de los trabajos sobre la determinación de la similitud entre textos (Lin, 1998; Budanitsky y Graeme, 2001; Ibrahim, 2003; Dagan, Glickman et al., 2005; Bar-Haim, Dagan et al., 2007; Agirre, Cer et al., 2012; Fernández, Gutiérrez et al., 2012a), así como otros que tratan el tema más específico del Reconocimiento de la Implicación Textual (RIT) (Dagan, Glickman et al., 2005; Bar-Haim, Dagan et al., 2006; Giampiccolo, Magnini et al., 2007; Bentivogli, Dagan et al., 2009; Bentivogli, Clark et al., 2010; 2011; Fernández, Gutiérrez et al., 2012b), permiten ver que esta tarea ha sido enfrentadas principalmente desde dos ópticas generales: el análisis de similitud léxico-sintáctica y la semántica de las unidades lingüísticas. La segunda variante, está influenciada por la riqueza del lenguaje y su relación intrínseca con el mundo, sin descartar que las actividades humanas hagan que la comparación semántica sea una tarea más compleja.

En este sentido, se conoce también que la pragmática (Blum-Kulka, 1996) tiene gran influencia en la captura de la similitud, hasta el momento -tal vez por ser la más compleja de analizar- no ha sido mayoría entre las técnicas utilizadas en las competiciones internacionales en esta temática. Para intentar corroborar esta afirmación, se ha realizado una búsqueda en Google Scholar por los términos: `Semántica`, `Pragmática` y `Sintaxis`. Luego, se le adiciona, a cada uno de estos términos la palabra Computacional. Finalmente, se suma también la palabra similitud. Los resultados se muestran en el ANEXO 13.

En este anexo, se puede apreciar que en los últimos cinco años aparecen una mayor cantidad de artículos que contienen el término pragmática con respecto a semántica y sintaxis, exceptuando el año 2009 en el que los artículos que almacenan el término semántica están en mayor cuantía.

Sin embargo, al adicionarse el término computacional, estos datos cambian radicalmente, pues los artículo que contienen la combinación `semántica + computacional` y `sintaxis +`

computacional, están muy por encima de los que contienen los términos pragmática + computacional.

Pero esta situación cambia totalmente, si además de computacional se adiciona la palabra similitud. En este caso la combinación sintaxis + computacional + similitud, pasa a ser la que aparece en la mayor cantidad de artículos.

Este sencillo experimento, da una medida de que -por lo menos- la producción de artículos que ofrecen una discusión sobre la similitud pragmática desde una óptica computacional, son extremadamente menos que los que comentan sobre los términos similitud semántica y similitud sintáctica, desde una perspectiva computacional. Además se puede ver que el tema de la similitud ha sido abordado por más cantidad de autores, vinculado a la sintaxis.

Abordando otros elementos, también muy utilizados en la búsqueda de similitud está el caso del alineamiento estructural de frases (Lalín, 2012). Es un tipo de alineamiento de secuencias basado en la comparación de formas. Estos alineamientos intentan establecer equivalencias entre los elementos de una frase basándose solo en su forma. De esta manera, para la determinación de similitud entre frases con un análisis léxico-sintáctico se tiene que, si las frases poseen diferencias pequeñas deberían ser reconocidas como muy parecidas. En particular, un solapamiento significativo entre sub-cadenas debería señalar un nivel alto de similitud entre las frases. También se puede considerar que dos frases que contienen las mismas palabras, pero en orden diferente deberían considerarse como parecidas.

Aunque estos tipos de análisis no pasan de medir solo la similitud entre grafemas, sin tener en cuenta la semántica, sin embargo han aportado resultados significativos en la determinación de la similitud entre frases. Una clara demostración de este hecho se puede ver en (Glickman, Dagan et al., 2006). Los resultados de esta técnica se explican porque en la mayoría de los casos dos palabras con distribuciones léxicas o sintácticas próximas, suelen tener también significados (semántica) muy próximos.

En particular, los métodos basados en alineamiento son otra forma de describir el análisis de similitud. Aunque han sido más ampliamente utilizados en la traducción automática (Och, Tillmann et al., 1999; Och y Ney, 2004), en la que se utilizan corpus paralelos alineados; en el reconocimiento de la similitud entre frases -aunque en menor frecuencia- han estado utilizándose métodos de alineamiento léxico (Brockett, 2007).

Con relación a esto, en (Dagan, Glickman et al., 2005), se puede ver el problema del alineamiento de diferentes formas. Mientras que en (Glickman, Dagan et al., 2006), utilizan la medición del

grado de solapamiento entre bolsas de palabras como una forma de alinear las frases. En (Jijkoun y Rijke, 2005) se describe un método de alineamiento que utiliza estadísticas de coocurrencia.

También, trabajos como los de (Tatu y Moldovan, 2007) y (Bar-Haim, Dagan et al., 2007), usan reglas de inferencia, las que codifican -entre otras cosas- conocimiento sobre el relacionamiento léxico. Estas aproximaciones utilizan características implícitas en el relacionamiento entre las frase para realizar su alineamiento.

Se puede constatar, en las últimas competiciones realizadas en la temática de la determinación de la implicación entre textos (Dagan, Glickman et al., 2005; Bar-Haim, Dagan et al., 2006; Giampiccolo, Magnini et al., 2007; Bentivogli, Dagan et al., 2009; Bentivogli, Clark et al., 2010; 2011), que los sistemas que utilizan características explícitas, son los que han obtenido mejores resultados (Marsi y Krahmer, 2005; Hickl, Williams et al., 2006). Sin embargo, estos sistemas han desarrollado sus trabajos adoptando formas de alineamiento poco documentadas y con la utilización de datos propietarios, lo que hace la comparación y reproducción bastante difícil.

En cuanto a la similitud semántica entre frases, aunque existen otras vías, ha estado centrada fundamentalmente en que las unidades a comparar se proyectan sobre un espacio semántico (a menudo WordNet (Miller, Beckwith et al., 1990)) dotado de una determinada métrica.

Aunque la mayoría tiene que usarlo en gran medida, dos de las tareas del PLN que más uso hacen de la búsqueda de similitud entre palabras y frases, ya sea léxica o semántica, son el reconocimiento de la implicación textual y la detección de paráfrasis. A continuación se comenta con más profundidad sobre estas temáticas.

1.1.1 El reconocimiento de la implicación textual

La implicación textual se puede considerar como un caso particular de similitud unidireccional. En el análisis de dos frases normalmente una frase implica la otra, pero esta relación no se da en sentido contrario. Por ejemplo, véanse las siguientes frases:

1. `El nuevo estadio fue construido en Madrid por Construcciones y Contratas S.A.`
2. `Construcciones y Contratas S.A ha construido un estadio en España.`

La frase uno (1) implica a la frase dos (2), pues si el estadio se construyó en Madrid, eso implica que fuese construido en España. Pero la frase dos (2), no puede implicar a la uno (1), pues no necesariamente construir un estadio en España quiere decir que tiene que haber sido en Madrid.

En esencia, se puede decir que el objetivo del Reconocimiento de la Implicación Textual es determinar cuándo la verdad de un texto conlleva o puede ser inferido de otro. La capacidad de hacer tales determinaciones se considera esencial para varias tareas de PLN, tales como Extracción de Información (EI), Resúmenes Automáticos (RA), Búsqueda de Respuesta (BR) y la Traducción Automática (TA) (Bar-Haim, Szpektor et al., 2005).

El RIT, desde el año 2005 ha sido considerado como un marco de aplicación independiente para el modelado de esta tarea (Dagan, Glickman et al., 2005). En este contexto las aplicaciones tienen que decidir si una frase, comúnmente denominada como el `texto` (T), implica a otra a la que se le llama la `hipótesis` (H).

El problema de la determinación de la similitud entre texto es una tarea que aún no ha sido resuelta con la precisión necesaria para una correcta interpretación del lenguaje natural. Los resultados alcanzados, hasta por los sistemas que marchan a la cabeza de estas investigaciones, aún no son los que se quieren para dar por cerrada esta tarea.

Abundando en esto, si se analizan los resultados alcanzados durante los años 2005 al 2011, en las siete competiciones del Recognizing Textual Entailment Challenge (Dagan, Glickman et al., 2005; Bar-Haim, Dagan et al., 2006; Giampiccolo, Magnini et al., 2007; Giampiccolo, Dang et al., 2008; Bentivogli, Dagan et al., 2009; Bentivogli, Clark et al., 2010; 2011) se observa que la exactitud de los sistemas apenas alcanza el 76%, solo en el RTE3 se alcanza un valor de 80%. Pero, también se ve que al modificarse la complejidad de los corpus de prueba, varían los resultados -disminuyendo- de un año a otro, lo que hace pensar que aún quedan cosas por resolver.

Como se explicó anteriormente, el problema de la similitud ha sido tratado utilizando técnicas puramente léxicas y también con la combinación de ellas. Con relación a los tipos de técnicas empleadas para llevar a cabo estos análisis, se encuentran los sistemas que se basan en conocimiento y los que utilizan aprendizaje automático. Para los sistemas basados en conocimiento, un elemento muy común es la escritura de patrones de extracción. A continuación se comenta sobre la importancia de las expresiones regulares en esta tarea, así como de la complejidad que implica la utilización de los lenguajes formales para su utilización en este objetivo.

1.2 Los recursos para la creación de patrones

Entre los tantos recursos utilizados en el PLN, se encuentran las expresiones regulares, ampliamente utilizadas (García, Fernández et al., 2005; Nadkarni, Ohno-Machado et al., 2011; Chang y Manning, 2012; Holzinger, Stocker et al., 2013) para el reconocimiento de patrones lingüísticos, validación de reglas, emparejamiento y remplazos, etc.

Antes de comentar sobre Expresiones Regulares, véase brevemente un concepto estrechamente vinculado con ellas, los Autómatas Finitos (AF). Un AF es una estructura matemática que representa un sistema o máquina abstracta capaz de reconocer símbolos de un determinado alfabeto. La máquina lee estos símbolos de izquierda a derecha. Cada vez que se lee un símbolo, la máquina efectúa un cambio de estado o transición (Brena, 2003). De manera informal, se dice que un AF acepta una palabra de entrada si, comenzando por lo que se conoce como estado inicial, la máquina alcanza un estado final tras leer el último símbolo de la cadena.

Los AFs pueden reconocer, solo, un determinado tipo de lenguaje, comúnmente conocido como lenguaje regular. Estos AF se utilizan para verificar las cadenas de este lenguaje. A su vez, las expresiones regulares proporcionan una forma concisa, aunque mucho menos intuitiva, para describir los lenguajes regulares.

En este punto, es necesario comentar un poco más sobre el término `Expresión Regular` (en lo adelante ER), también conocida con el nombre de patrón, es una expresión que se utiliza para describir un conjunto de cadenas sin enumerar sus elementos. Por ejemplo, en español muchas palabras se escriben de dos maneras distintas (fenómeno conocido como alternancia gráfica), así las palabras: azimut y acimut, se pueden describir mediante el patrón `a(z|c)imut`. Este elemento sería también de gran utilidad para determinar la similitud entre palabras.

Una ER, es una forma de representar a los lenguajes regulares (finitos o infinitos) y es construida utilizando caracteres del alfabeto sobre el que se ha define el lenguaje. Las ER se construyen utilizando operaciones booleanas, para lo que se utilizan los operadores: concatenación, unión, así como clausura de Kleene (Kleene, de Bruijn et al., 1971), las que se explicaran más adelante.

Las ER son una potente herramienta de programación, por lo que presentan aplicaciones diversas que básicamente consisten en el manejo de patrones dentro de un texto, es decir, si se encuentra alguna coincidencia dentro del texto con determinado patrón, previamente declarado, es posible interpretar todo el texto y ejecutar las operaciones que se requieran. De aquí que sean utilizadas en la implementación de recursos para el reconocimiento de entidades, extracción de información y otras aplicaciones de PLN.

Además, las ER tienen múltiples ventajas, por ejemplo:

a) Existe soporte para expresiones regulares en gran variedad de lenguajes de programación. La mayor parte de la sintaxis de las expresiones regulares trabaja igual en una amplia variedad de lenguajes de programación y herramientas.

b) Las expresiones regulares pueden ayudar a escribir código corto y ahorran tiempo. Según

(Friedl, 2006), incluso para quienes no dominan a la perfección la sintaxis, las expresiones regulares son generalmente la forma más rápida de hacer el trabajo, si se compara con el tiempo que tomaría hacerlo todo desde cero con un leguaje de programación (ver ejemplo en ANEXO 5).

c) La mayoría de las expresiones regulares funcionan muy rápido en casi todos los casos. Aunque, es bueno aclarar que si se tienen en cuenta las bases de optimización de las ER, se logran mejores resultados. Las expresiones regulares pueden encontrar prácticamente todo.

Para una mejor claridad de la oportunidad que representan las ER, véase el siguiente ejemplo sobre las ventajas expuestas anteriormente, en especial en el inciso b):

Tabla 1.1. ER para detectar variantes de la frase: objetivos generales educativos.

```
preg_replace("/^<p>(((objetivo(s)?)?(general(es)?)?educativo(s)?)
(\.|:)?)<\/p>/i","<p><Obj_Educ>\\1</p>")
```

Esta ER representada en la Tabla 1.1, es capaz de detectar y etiquetar `objetivos educativos`, o sea, detecta la línea del texto donde estos comienzan y la señaliza con la etiqueta personalizada `<Obj_Educ>`. Esta expresión presenta cierto grado de flexibilidad, o sea, siempre que aparezcan escritos de una de las formas que se han analizado previamente (ver ANEXO 4), los detecta y etiqueta correctamente. Esta ER es capaz de identificar 54 patrones, si aparecen indistintamente en mayúscula o minúscula, pero sólo si representan un subtítulo, o sea, si se encuentran en una línea del documento, no si aparecen embebidos en el contexto, evitando así identificar elementos mencionados dentro de un párrafo.

Para realizar esta tarea, por ejemplo, en un lenguaje de programación como PHP -sin usar las expresiones regulares- hubiesen sido necesarias muchas líneas de código. Además, disminuiría la eficiencia y rapidez de ejecución del programa. Así por ejemplo, para etiquetar los mismos patrones de objetivos educativos, hubiese sido necesaria toda la codificación que aparece en ANEXO 5. Comparando el código que aparece en el anexo con la ER en la Tabla 1.1, se puede ver que, aún cuando ambos códigos logran los mismos resultados (detectar y etiquetar los objetivos educativos), la ER es más eficiente tanto en cantidad de código como en velocidad de ejecución. Además, si el formato del texto cambia, sólo se harían pequeños cambios en la expresión, sin que sea necesario modificar más código para procesar los datos.

Otro ejemplo, pero en este caso que corrobora la complejidad de la escritura e interpretación de ER, así como el tamaño que pueden alcanzar, se muestra en el ANEXO 6. En este anexo aparece una

porción de una ER, utilizada para reconocer algunas expresiones temporales (De la Vega, Pérez et al., 2012).

Otro problema, que también puede ser comentado sobre las ER, es que la sintaxis es algo difícil de recordar. Debido a que los símbolos utilizados para su construcción, generalmente no tienen un significado semejante al que pudieran tener en lenguaje natural o en lenguajes de programación.

A pesar de estas mencionadas desventajas, muchos sistemas de PLN y sus respectivas tareas intermedias hacen un amplio uso de las ER para lograr sus objetivos. Con mucha frecuencia las ER son utilizas para reconocer patrones en un texto, esto se realiza a través de las reglas escritas usando lenguajes regulares.

La gran dificultad de esta utilización radica, primeramente, en que muchos de los sistemas generan sus patrones dentro del código de sus aplicaciones. Esto es, en parte, debido a que si se ofreciera esta posibilidad para las reglas fuesen modificables, entonces estarían agregando una complicación más, los usuarios tendrían que aprender el lenguaje formal con el que se han creado.

Entonces, el objetivo a lograr sería encontrar una forma de generar estas expresiones sin necesidad de ser un experto en lenguajes regulares, así como poder desligarlas del código de las aplicaciones.

A continuación, en el epígrafe 1.3, se comenta sobre otro tema que guarda estrecha relación con la generación de ER y de gran utilidad para la interpretación de textos, el Reconocimiento de las Entidades Nombradas.

1.3 El Reconocimiento de Entidades

Tal vez el trabajo más antiguo en la rama del reconocimiento de entidades -criterio muy particular de este investigador- es el que realizara el egiptólogo francés Jean-François Champollion, conocido como Champollion el joven, al descifrar los primeros cartuchos reales con los nombres de Ptolomeo (Ptolomis) y Cleopatra (Kliopat) en el obelisco de Philae. Tal vez pueda pensarse que el reconocimiento de entidades de hoy dista mucho de aquel que hiciera Champollion en 1821, pero aún permanece su esencia, el afán por descubrir las pistas necesarias que llevan al entendimiento de un lenguaje.

El Reconocimiento de Entidades Nombradas, como se conoce hoy, es una parte importante de muchos de los sistemas de Procesamiento del Lenguaje Natural.

Al reconocer y clasificar las entidades existentes en un texto, se podrá tener una información más clara y por consiguiente una mejor interpretación de dicho documento. Clasificando las entidades se puede saber de quién o de qué se habla. En áreas como la Extracción de Información y la Búsqueda

de Respuestas tendrían una respuesta adecuada las preguntas: ¿quién…?, ¿dónde…?, ¿en qué lugar…?, ¿cuándo…?, etc., si se identificaran correctamente las entidades que dan respuesta a estas preguntas (Fernández, Muñoz et al., 2004).

El termino Named Entity es ampliamente utilizado en el Procesamiento del Lenguaje Natural. Fue en la MUC-6 (Sixth Message Understanding Conference) (Grishman y Sundheim, 1996) que se escuchara por primera vez. Esta conferencia se enfocó en la Extracción de Información, pero al definir la tarea los investigadores se percataron de la necesidad de reconocer la información sobre nombres propios, de personas, organizaciones y localidades, a estas se le llamó ENAMEX. También es necesario reconocer expresiones temporales, las que fueron designadas en las MUC como TIMEX, así como las NUMEX en las que se encuentran las entidades del tipo moneda y porcientos.

Al tipo de entidad que se apartaba de las clásicas ENAMEX, fue designado como Misceláneas, dentro de esta se encuentran títulos de obras teatrales, de libros, nombres de proyectos, etc. Las entidades tipo Misceláneas fueron introducidas en las conferencias CONLL. El reconocimiento de estas entidades fue reconocido como una sub-tarea de la EI y fue nombrada como: Named Entity Recognition and Classification (NERC) (Nadeau y Sekine, 2007), en español Reconocimiento y Clasificación de Entidades Nombradas, en lo adelante (REN).

El problema de reconocer y clasificar una entidad ha sido abordado desde varias perspectivas. En los inicios muchos de estos intentos se basaron en el uso de reglas (Appelt, Hobbs et al., 1993; Gaizauskas, Humphreys et al., 1995; Black, Rinaldi et al., 1997; Humphreys, Gaizauskas et al., 1998; Budi y Bressan, 2003).

Un tiempo después aparecen sistemas basados en aprendizaje de máquina (Borthwick, Sterling et al., 1998; Chieu y Ng, 2002; Bender, Och et al., 2003; Curran y Clark, 2003). Para estos sistemas existen dos clasificaciones, los no supervisados y los supervisados. Para estos últimos es necesario poseer corpus anotados con la información necesaria para que estos algoritmos pudieran realizar el proceso de aprendizaje. En esta área se utilizan diversas técnicas como los Modelos de Probabilidad Condicional de Máxima Entropía (MPCME), Support Vector Machine SVM, Hidden Markov Models (HMM), Redes Neuronales (RN), etc.

Algunos sistemas unen la potencia de ambas técnicas y usan métodos híbridos mezclando reglas con aprendizaje supervisado (Mikheev, Grover et al., 1998; Srihari, Niu et al., 2000; Mansouri, Affendey et al., 2008). Estos sistemas obtienen muy buenos resultados, pero arrastran las dificultades de los sistemas basados en reglas, necesidad de generar las reglas, dependientes del lenguaje, etc.

Los trabajos que basan su investigación en el uso de reglas para la detección de entidades obtienen resultados prometedores. Sin embargo, estos sistemas dependen fuertemente de la habilidad de los creadores de estas reglas para su correcto funcionamiento, así como de la capacidad de cubrir todas las reglas necesarias para resolver el problema. El tiempo de desarrollo de estos sistemas es relativamente amplio. En unión a esto, la mayoría de ellos no permiten la edición de sus diccionarios de reglas, lo que dificulta su uso en dominios diferentes.

Además, aún cuando algunos colocan sus diccionarios de reglas en ficheros independientes, los que podrían ser editados por terceros, el lenguaje que utilizan para representar las reglas es poco conocido por la mayoría de los investigadores. Trabajos como los que se pueden ver en (Fernández, Muñoz et al., 2004; Toral, 2005) y otros, utiliza la potencia de las expresiones regulares para la generación de reglas. Si bien -en cierta forma- esto estandarizaría en algo la escritura, como se ha visto anteriormente, la verdad es que este tipo de lenguaje es bastante críptico para la mayoría de las personas, pues se aleja mucho de la estructura del lenguaje natural.

Otra tarea intermedia que reviste gran importancia, sobre todo para los sistemas de Recuperación de Información es el llamado proceso de Stemming. En el próximo epígrafe se estará comentando sobre este importante tema.

1.4 El Stemming

El Stemming es una tarea de gran importancia, principalmente para los recursos que indexan y recuperan de información. El objetivo que persiguen estos sistemas es mejorar varios aspectos relacionados con la recuperación, a partir del truncamiento de algunas terminaciones de las palabras, reduciéndolas a la raíz para su almacenamiento. En la mayoría de los casos, las variantes morfológicas de un término pueden converger a un único representante de forma, así la interpretación semántica puede ser considerada como equivalente, a los efectos de las solicitudes de los recuperadores de información.

Se dice que los algoritmos de Stemming ayudan a aumentar la cobertura de los sistemas de RI, por ejemplo, una consulta que contenga la palabra `campo` también encuentra documentos en los que aparezca el término `campestre`, `campiña` y otros. Esto, se produce porque el stem de las palabras es el mismo, `camp`. Esta técnica permite reducir el tamaño de los índices y ampliar la pregunta que realizan los usuarios, mejorando así la cobertura de estos sistemas.

Por el contrario, no es solo ventajas lo que involucra este proceso, también se cometen errores asociados a este truncamiento de las palabras. Retomando el ejemplo anterior, utilizando el stem

camp, se recuperarían documentos en lo que aparece la palabra campana, campeonato, campechano, pues también comparten el mismo stem; sin embargo estos términos no tienen que ser exactamente lo que desearía el usuario recuperar.

El algoritmo más utilizado para el Stemming es el de Porter (Porter, 1980). Aunque existen además otros métodos basados en análisis lexicográfico, algoritmos similares como (Lovins, 1968), (Dawson, 1974), (Paice, 1990), (Krovetz, 1993) y otros que serán debatidos en próximos epígrafes.

Haciendo un análisis de estos stemmers, el de Porter en particular, para su funcionamiento necesita una completa definición de afijos[11] del idioma, los cuales son posteriormente truncados. La utilización de una lista de sufijos con diversas normas en el algoritmo de Porter, logra una tasa de éxito para la extracción de sufijos significativamente inferior al 100%, independientemente de cómo se evalúa el proceso. Se ha detectado, en palabras que no tiene un afijo incorporado, que si existe una porción de ella que coincide con un afijo de la lista, es arbitrariamente truncada. Por ejemplo, para la palabra en inglés wander, determinaría como stem la partícula wand. El problema es que la terminación (er) en wander ha sido tratada como un sufijo, cuando en realidad es parte del stem. Se podrían tener en cuenta todas estas excepciones, pero el costo de crear todas estas reglas sería elevado, además de que solo servirían para el idioma específico con el que se esté trabajando.

Por otra parte, el principal problema del enfoque Lovins es su consumo de tiempo y datos. Además, su tabla de afijos no está lo suficientemente completa. En ocasiones es poco fiable y con relativa frecuencia no logra formar palabras a partir de los stems o hacer coincidir los stems de las palabras con significados parecidos (Jivani, 2011).

En cabio, el stemmer de Dawson, aunque es una extensión del de Lovins, cubre una lista mayor de sufijo y es algo más rápido en ejecución (Dawson, 1974). Según (Jivani, 2011), la desventaja mayor de este algoritmos es su complejidad y que adolece de la implementación de reusabilidad.

En base a los errores que se derivan, Paice llegó a la conclusión de que el stemmer de Porter produce una tasa menor de errores que el stemmer de Lovins (Lovins, 1968). Sin embargo observó que el stemmer de Lovins es un stemmer más pesado que produce una mayor reducción de datos (Paice, 1994).

A su vez, el stemmer de Paice tiene como principal desventaja que utiliza un algoritmo muy pesado y suelen producirse errores de truncamiento excesivo (overstemming). Todos estos stemmers tienen la dificultad común de ser dependientes del lenguaje.

[11] Los afijos son secuencias lingüísticas que se anteponen (prefijos), se posponen (sufijos) o insertan (infijos) en una palabra o lexema.

Por su parte, el stemmer de Krovetz es un algoritmo bastante complejo en el que si el tamaño del documento de entrada es grande, no funciona muy eficazmente. El defecto más importante y obvio, para todos los algoritmos basados en diccionario, es su incapacidad de hacer frente a las palabras que no se encuentran dentro del léxico. También, la dificultad de que es necesario crear los diccionarios manualmente, lo que requiere esfuerzos importantes. Este stemmer no se caracteriza por obtener buenos resultados de precisión y cobertura.

En el ANEXO 12 se ofrece una comparación más detallada de la ventajas y desventajas de los diferentes algoritmos de Stemming (tabla tomada de (Jivani, 2011)).

Con todo lo antes expuesto, se puede ver que no es tan sorprendente el hecho de que los programas de PLN resulten ser complejos y su desempeño, en muchos casos, sea mejorable en varios aspectos.

Por esta razón, el aumento del error acarreado por la utilización de diferentes recursos en el reconocimiento de entidades, la dificultad para la generación de patrones de extracción utilizando lenguajes regulares, las imprecisiones de la distancia de edición, los bajos resultados que se vienen experimentando en el reconocimiento de la similitud textual, así como los errores que poseen los Stemming actuales, constituyen el problema científico a resolver en esta investigación.

Esto hace que esta investigación centre su objeto de estudio en las tareas intermedias del Procesamiento del Lenguaje Natural, que inevitablemente se hace necesario acotar debido a la gran cantidad de tareas que se involucran en este proceso. De esta forma el campo de acción a estudiar lo constituyen las tareas intermedias: reconocimiento de entidades, la generación de patrones de extracción a partir de lenguajes regulares, la distancia de edición, las métricas para obtener la similitud entre frases y el proceso de Stemming.

Este trabajo intenta estudiar los problemas plantados con anterioridad y buscar si existen las vías de solución, para ello se propone dar respuesta a las Preguntas Científicas que aparecen a continuación:

1. ¿Es posible prescindir de los recursos para el análisis morfo-léxico, sintáctico y semántico en el reconocimiento de entidades, para con ello evitar los errores acarreados por los recursos dedicados a estas tareas y, aún así, obtener resultados a la altura de los alcanzados a nivel internacional?
2. ¿Puede lograrse un método que permita abstraer las complejidades de los lenguajes formales en la creación de patrones de extracción, a partir de expresiones regulares para su utilización en diferentes tareas del PLN?

3. ¿De qué forma mejora a la distancia de edición, utilizada en la medición de similitud, de manera que al comparar palabras con diferencias en la raíz -que provocan cambio en el significado- sean penalizadas consecuentemente?
4. ¿Se podrá utilizar la modificación de la distancia de edición en la generación automática de familias de palabras y obtener resultados por encima de un 90% de precisión en esta tarea?
5. Mediante la modificación de la distancia de edición, ¿se podrá obtener un stemmer independiente del lenguaje, a partir del agrupamiento de familias de palabras, que obtenga resultados superiores al 95% de precisión, cobertura y exactitud?
6. ¿Qué influencias tiene la utilización de un alineamiento léxico y el semántico en reconocimiento de la similitud textual desde una perspectiva multidimensional?
7. ¿Podrá mejorar la polaridad sentimental la determinación de la Implicación Textual?

Debido al amplio espectro que cubre el PLN esta tesis se enfoca solo en el trabajo con la información contenida en documentos de textos digitales. Se utilizan en este enfoque los conceptos y técnicas asociados al análisis morfo-léxico, sintáctico y semántico, no así los referentes a la pragmática.

Por ello, el `principal objetivo` es el mejoramiento de diferentes tareas intermedias, entre las que se encuentran: el reconocimiento de entidades; proceso de generación de patrones usando lenguajes regulares; la distancia de edición; determinación de la similitud léxico-sintáctica y semántica y el proceso de Stemming.

Para dar respuesta al objetivo principal se proponen las tareas siguientes:

1. Realizar un estudio del estado de la cuestión para cada una de las temáticas a investigar. Abordando también, los referentes teóricos directamente asociados al objeto de estudio.
2. Creación de un método híbrido para el reconocimiento de entidades que prescinda del pre-procesamiento (análisis léxico, sintáctico y semántico) de los corpus.
3. Estudiar el comportamientos de diferentes clasificadores para el aprendizaje automático en el reconocimiento de entidades a partir de corpus no anotados.
4. Crear un método de generación de expresiones regulares que permita evitar la complejidad de los lenguajes regulares.
5. Proponer una modificación a la distancia de edición, que permita superar los errores que esta posee, así como la obtención de varios atributos lexicográficos en la misma iteración

que facilite el agrupamiento en familias de palabras.

6. Proponer un método de obtención del stem a partir del agrupamiento de familias utilizando la extensión de la distancia de edición.
7. Evaluar la influencia de la polaridad sentimental en la Implicación Textual.
8. Evaluar varios métodos de reconocimiento de la paráfrasis que utilicen la menor cantidad posible de recursos y sus resultados se mantengan entre los alcanzados en esta temática a nivel internacional.
9. Crear dos métodos de alineamiento que utilicen la multidimensionalidad para determinación de similitud textual.

1.5 Descripción de los capítulos

Al terminar la introducción se presenta el Capítulo II. En él se abordan todos los referentes teóricos tomados como punto de partida para esta investigación. Se hace una pequeña descripción de las cuatro tareas principales estudiadas en el PLN, en las que los resultados de esta investigación drían tener gran influencia. También se exponen los antecedentes y el estado de la cuestión en cada una de las tareas investigadas, las que se resumen en: la similitud léxico-semántica (Implicación textual y Paráfrasis), la generación de Expresiones Regulares en el PLN, el Reconocimiento de Entidades y el Stemming. Se analizan las fortalezas y debilidades de los trabajos precedentes.

En el Capítulo III, se describen los detalles de la investigación desarrollada y se exponen los métodos realizados para dar solución a la problemática encontrada. Entre estos se encuentran: la descripción de la extensión realizada a la distancia de edición, el estudio sobre la influencia de la polaridad sentimental en la implicación textual, los alineamientos léxicos y semánticos en el reconocimiento de la similitud entre frase, los diferentes métodos implementados para el reconocimiento de la paráfrasis, el método de Stemming utilizando la extensión de la distancia de edición, la experimentación en el reconocimiento de entidades y finalmente un método de generación de Expresiones Regulares para la creación de patrones en tareas de Procesamiento del Lenguaje Natural. En el capítulo IV se recogen las conclusiones generales y los trabajos futuros. También se expone la producción científica, dividida en publicaciones y participaciones en competiciones internacionales.

Capítulo 2 Marco teórico referencial y antecedente

Este capítulo comienza con una breve descripción de las tres tareas fundamentales del Procesamiento del Lenguajes Natural. A continuación se detalla sobre el estado de la cuestión en las diferentes temáticas abordadas. Se resumen elementos esenciales sobre la medición de efectividad y eficiencia en los sistemas de PLN. Se reflejan los elementos teóricos sobre la similitud tanto léxico-sintáctica, como semántica. Se realiza un recorrido por la teoría sobre las Expresiones Regulares y los Autómatas Finitos. También se presentan los principales aspectos teóricos el Reconocimiento de Entidades. Se hace un análisis de algunas aplicaciones que implementan estas técnicas, así como una breve descripción de los principales eventos que auspician la confrontación de estos saberes a nivel internacional. Por último se analizan los elementos teóricos del proceso de Stemming.

2.1 El Procesamiento del Lenguaje Natural

Como se puede apreciar en la Figura 2.1 el PLN se entiende como la fusión de la Lingüística Computacional (LC) y la Inteligencia Artificial (IA). La mayoría de los autores (Valdivia y Teresa, 2004; Guerrero, García et al., 2010; Martí, Quesada et al., 2011) consideran el PLN como una subclase del la IA, pero a criterio de este autor es más una clase que hereda de ambas, la LC y la IA. No es objetivo de esta tesis filosofar sobre estos conceptos, por eso no se entrará en debate.

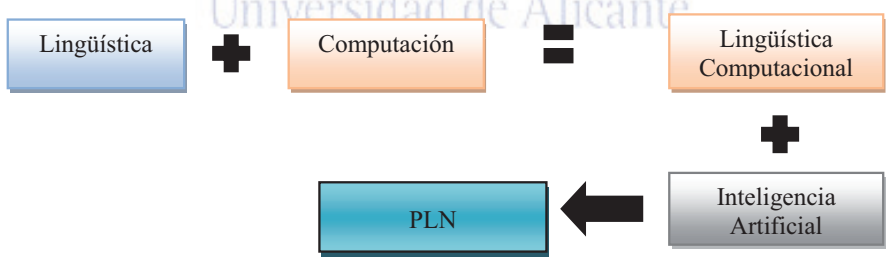


Figura 2.1. Composición del PLN

Definiendo cada una de estas ciencias en particular, la `Lingüística` sería la encargada del estudio científico del lenguaje. A su vez la `Ciencia de la Computación` abarca el estudio científico de soluciones a problemas mediante una computadora, involucrando en este proceso a los algoritmos y lenguajes para esa comunicación. La `Lingüística Computacional`, acuñada así por Hays, es la rama de la Lingüística que emplea técnicas y conceptos computacionales para la solución de problemas lingüísticos y fonéticos (Hays, 1967). En (Gómez Guinovart, 2007), se plantea que puede ser considerada

una subdisciplina de la lingüística teórica, ya que uno de sus objetivos es la elaboración de modelos formales del lenguaje humano, capaces de ser implementados computacionalmente.

El termino `Inteligencia Artificial`, fue acuñado formalmente en 1956 por John Patrick McCarthy. A juicio de este investigador una definición acertada de Inteligencia Artificial, de las tantas que existen, es la planteada en (Charniak y McDermott, 1985), donde expresan que es: el estudio de las facultades mentales mediante el uso de modelos computacionales.

Para el último concepto a definir, el `Procesamiento del Lenguaje Natural`, la mayoría de las definiciones aportadas por los teóricos de esta disciplina se pueden resumir en que el PLN es: la aplicación multidisciplinar de técnicas de tratamiento del lenguaje para la resolución de problemas en diversas áreas, que necesitan de la interpretación del lenguaje humano, por medio de técnicas y métodos computacionales.

Vistas las definiciones anteriores, se hace necesario conocer los distintos niveles de análisis del lenguaje que se aplican en los sistemas de PLN. Según se puede ver en (Moreno, Palomar et al., 1999) las formas de conocimiento de la lengua son: fonética, morfología, sintaxis, semántica y pragmática o gramática de contexto. De acuerdo con la clasificación anterior, los sistemas dedicados al PLN trabajan el análisis de una frase en los niveles siguientes:

Análisis morfológico-léxico: Estudia la estructura interna de las palabras para delimitar, definir y clasificar sus unidades. Los analizadores morfo-léxicos crean tokens[12] de una secuencia de caracteres de entrada para que luego sean procesados por el analizador sintáctico. A su vez, este análisis incluye las siguientes tareas intermedias: El tokenizado (en inglés tokenizing), segmentación de frases, la extracción de categorías gramaticales (POS-tagging (Part of the Speech Tagging)), el Stemming y el lematizado de las palabras, entre otras.

Análisis sintáctico: conocido como parsing, convierte el texto de entrada de unidades léxicas en una estructura tipo árbol o red, las que son más bondadosas para el posterior análisis y captura de la relación jerárquica.

Análisis semántico: a partir de la estructura obtenida por el análisis sintáctico, genera otra estructura o forma lógica asociada que representa el significado o sentido de la frase.

Análisis contextual o de función pragmática: utiliza la forma lógica o estructura semántica de la fase de análisis semántico para desarrollar la interpretación final de la frase, en función del modo en que el contexto influye en la interpretación del significado.

[12] Un token o componente léxico es una cadena de caracteres que constituye un elemento unitario con significado coherente. Al descomponer una oración, cada palabra constituiría un token y a este proceso se le llama tokenizado.

Como se ha podido ver anteriormente, el PLN está compuesto por un gran número de técnicas y procedimientos, los que se aplican con el objetivo de interpretar el lenguaje humano. Dentro de las líneas que se desarrollan en esta temática, se pueden mencionar tres grandes grupos en los que tradicionalmente se han dividido las tareas del PLN: Recuperación de Información (RI) (en inglés Information Retrieval (IR)), Extracción de Información (EI) (Information Extraction (IE)) y Búsqueda de Respuesta (BR) (Question Answering (QA)). A continuación de hace una muy breve descripción del contenido de cada una de estas tareas.

2.2 Recuperación de Información

Según el diccionario de la Real Academia Española[13], recuperación es la acción y efecto de recuperar. También, recuperar significa volver a tener. Por lo tanto, recuperar información significa volver a tener una información. No necesariamente información propia en particular, ni tampoco que se haya tenido con anterioridad, sino aquella que alguna vez, en algún espacio de tiempo, ha sido producida y está disponible para ser recuperada. Esta recuperación se hace a partir de una petición del usuario, conocida también como pregunta del usuario. Esta pregunta es generada a partir de una necesidad de información determinada.

Según (Baeza-Yates y Ribeiro-Neto, 1999) los tres modelos clásicos en la recuperación de información son: el Boolean, el Vectorial y el Probabilístico. Con el transcurso de los años han sido propuestos nuevos modelos, principalmente con el objetivo de mejorar los resultados de la recuperación. De forma general, el proceso de RI se puede resumir en tres fases (ver ANEXO 1). Al igual que todas las tareas del PLN, la RI necesita de varias tareas intermedias como el tokenizado, el Stemming, la indexación, la determinación de similitud, etc.

En este documento no se profundizará en esta temática debido a que no es objetivo específico de esta investigación.

2.3 Extracción de Información

El objetivo principal de los sistemas de Extracción de Información es el procesamiento de textos escritos libremente, con el fin de encontrar información útil respecto a un dominio de interés. La información extraída es transformada en una representación fuertemente estructurada, la que se obtiene al recorrer secciones relevantes del documento para obtener la información útil.

[13] http://lema.rae.es/drae/

Los sistemas de EI precisan para su buen funcionamiento del pre-proceso de los textos en una secuencia de pasos, entre los que se encuentran: El análisis léxico y morfológico, la desambigüación del sentido de las palabras, el reconocimiento de entidades, el análisis sintáctico, así como el semántico, el análisis del discurso, la resolución de anáfora y la correferencia, así como la extracción de eventos relevantes del dominio. Como se puede apreciar este proceso necesita de muchas tareas intermedias para lograr su objetivo.

A diferencia de los sistemas de Recuperación de Información, que extraen los documentos relevantes a una determinada pregunta, los sistemas de EI tiene como objetivo construir modelos que encuentren y relacionen información relevante, mientras ignoran la no relevante. El ANEXO 2, muestra lo que sucede dentro del corazón den un sistema de EI. Este esquema ha sido inspirado en el realizado por Horacio Rodríguez en su material audio visual titulado: Extracción de Información -Tutorial (Rodríguez Hontoria, 2001).

2.4 Búsqueda de Respuesta

Como se puede ver en el ANEXO 1, el recurso más utilizado para la recuperación de información lo constituyen los conocidos buscadores o motores de búsqueda. A través de ellos se pueden extraer de Internet los documentos que contienen la posible respuesta a una determinada pregunta. Una vez recuperados los documentos, aquellos en los que se sospecha que estará la respuesta buscada, se utilizan para indagar en ellos -con mayor profundidad- dónde se encuentra. Esto constituye, en muchas ocasiones, una gran pérdida de tiempo y esfuerzo; en el peor de los casos puede suceder que la respuesta no aparezca en estos documentos.

Esta problemática ha hecho que la comunidad de investigadores en este tema, se vuelque en la búsqueda de sistemas que permitan formular las necesidades de información utilizando el lenguaje natural, donde la respuesta esperada sea precisa y escueta. En lugar de una colección de documentos donde podría estar la respuesta, los usuarios recibirán la ubicación exacta del documento en el que debería estar esa información que busca. A groso modo, este es el objetivo de un sistema de búsqueda de respuesta.

En (Gómez, 2007) se plantea que los principales componentes de un sistema de búsqueda de respuesta son: Análisis de la pregunta, selección de documentos o pasajes y extracción de respuestas. Un esquema general de un sistema de BR sería según (Gómez, 2007) como se muestra en el ANEXO 3.

Cada uno de los componentes individuales que se muestran en el ANEXO 3, hacen uso de muchos recursos y tareas intermedias que le permiten llegar a su objetivo final.

Teniendo en cuenta que la totalidad de las tareas finales del PLN, hacen un amplio uso de una gran cantidad de tareas intermedias, en lo adelante se hace un recorrido por los referentes teóricos y la descripción de aquellas tareas sobre las que se ha investigado. El siguiente epígrafe define algunos conceptos de gran importancia para este trabajo.

2.5 Algunos referentes teóricos de utilidad en esta investigación

Para el desarrollo de cualquier investigación son fundamentales los pilares teóricos en los que se sustenta. Este epígrafe tiene la función de esclarecer los principales elementos de la teoría, en los que se apoya este trabajo, sobre todo aquellos que ayundan a lograr la transformación del objeto de estudio. Se da inicio a esta fundamentación con una descripción de los elementos lingüísticos que se han tenido en consideración.

2.5.1 Elementos de la lingüística de gran importancia para la investigación

Para una correcta interpretación de las técnicas y métodos utilizados en el PLN, son necesarios los conocimientos de varias disciplinas. Dentro de ellos la lingüística, en especial los que tienen que ver con elementos de la morfología y otros muy particulares de la gramática de los lenguajes. A continuación se rememoran algunos que han sido de gran importancia para esta investigación.

2.5.1.1 La etimología de la palabra

Para el español -como en muchas otras lenguas romance- en la elaboración de un algoritmo de cálculo de similitud basado en cadenas de caracteres, se debe hacer un análisis profundo de los elementos constituyentes de las palabras. En el nivel morfológico se deben observar las transformaciones ocurridas en los morfemas lexicales o lexemas y, de alguna forma, diferenciarlas de las transformaciones que ocurren en los morfemas gramaticales o grammemas.

Por el contrario, algunas medidas de similitud basadas en la comparación de palabras, por ejemplo, la distancia de Levenshtein, hacen un análisis de las transformaciones necesarias para poder convertir una palabra en otra. Así se obtiene una medida de cuán cerca o lejos están dichas palabras entre sí. Pero a juicio de este investigador, una medida de este tipo, debería analizar también cuáles de esas transformaciones ocurren en el lexema o raíz de la palabra. Esto permitiría dar una importancia diferente a las transformaciones que ocurren directamente en la raíz, que evidentemente pesan más, pues provocarían un posible cambio de significado de la palabra.

Con relación a esto, en (Fernández, Díaz et al., 2009) se plantea: *"...la contribución léxica de algunos prefijos puede cambiar el sentido de las palabras, causando indicativo de anterioridad, oposición,*

superioridad, inferioridad, exceso, decremento, etc." Esto, evidentemente es un caso de transformación en el lexema de la palabra. Véanse los ejemplos siguientes:

a) Oposición: (anti) + fascista = antifascista

   Fascista: partidario del régimen o doctrina del fascismo.

   Antifascista: contrario u opositor al régimen o doctrina del fascismo.

b) Inferioridad: (infra) + humano =infrahumano)

   Humano: De la humanidad o el ser humano, o con sus características.

   Infrahumano: inferior a lo considerado propio del ser humano.

c) Exceso: (súper) + sónico= supersónico

   Sónico: que pertenece o concierne al sonido.

   Supersónico: que posee una velocidad superior a la del sonido.

d) Decremento: (des) + motivado= desmotivado

   Motivado: interesado o animado por alguna cosa o razón.

   Desmotivado: pierde la animación o el interés por algo.

e) Negación o privación: (i) + repetible= irrepetible

   Repetible: que se puede repetir obteniendo los mismos resultados.

   Irrepetible: que no puede repetirse.

En estos ejemplos se puede ver que es necesario algo más que una medida que controle meramente la cantidad de transformaciones. Para convertir -por ejemplo- `desmotivado` en `motivado`, se obtendría una distancia igual a tres (3) usando la distancia de Levenshtein. Sin embargo, semánticamente estas palabras son totalmente contrarias y merecerían un valor de distancia mucho mayor. O tal vez, si se quisiera comparar con esta misma distancia las palabras `repetible` e `irrepetible`, el valor en este caso sería solamente dos (2). Pero, aún más pequeña sería esta distancia -indicando un parecido muy grande- si se analizan las palabras `tiene` y `viene`, la diferencia par este caso es solo uno (1), sin embargo desde el punto de vista semántico no tienen ningún nexo.

Queda claro con los ejemplos anteriores, que es necesario resolver este problema para intentar reducir este tipo de errores. Para poder cumplir con este elemento la medida desarrollada en esta investigación, utiliza el un término en su expresión que penaliza la posición en que se ha realizado la operación, no permitiendo

que palabras que comparten muchos caracteres, pero con lexemas diferentes sean determinadas como de gran parecido.

Aclarando aún más esta situación, se puede plantear que a nivel morfológico la unidad sígnica más pequeña es el morfema, el cual se define como la unidad mínima con significación. Es sabido que no todos los morfemas de una palabra poseen el mismo valor en su significado. Se pueden separar en, aquellos que aportan un nuevo contenido, o sea, que poseen significación léxica (comúnmente llamados tallo o raíz (en inglés stem)) y por otro lado los que sirven para indicar las categorías gramaticales como género, número, tiempo, persona, modo. Estos, permiten además establecer las marcas de la concordancia morfológica entre las palabras y los sintagmas, es decir, tienen significación gramatical.

De esta manera, a los que aportan la significación léxica se les conoce como morfemas lexicales o lexemas, en cambio a los que tiene que ver más con el aporte gramatical se les conoce como morfemas gramaticales o grammemas (De la Cueva, González et al., 2005).

A los morfemas derivativos se les conoce comúnmente como afijos[14]. "Son las formas ligadas que son añadidas a la derivación a las formas bases, según su posición respecto a la forma base o lexema, se clasifican en prefijos, infijos y sufijos" (Lewandowski, 2000).

Resumiendo, una palabra está compuesta por un lexema y sus grammemas. El lexema aporta la significación principal y los grammemas lo modifican, ofreciendo así información complementaria.

Según se plantea en (De la Cueva, González et al., 2005), los morfemas lexicales poseen mayor carga significativa -constituyendo un inventario infinito y abierto- determinado por la constante creación de nuevos objetos y conceptos, pues aluden a la realidad objetiva. Son dependientes, ya que no aparecen solos; necesitan ser combinados con los morfemas gramaticales para alcanzar una determinada categoría funcional y con esto lograr la forma completa que corresponde a la palabra. Véase el siguiente ejemplo:

Bailadoras `{bail-}{-a-}{-d-}{-or-}{-a-}{-s-}- sustantivo`.

Bailábamos `{bail-}{-á-}{-ba-}{-mos-}- verbo`.

En dependencia de los morfemas gramaticales con los que se ha combinado el lexema `{bail-}`, se obtienen dos categorías funcionales distintas: sustantivo y verbo.

Por el contrario, los morfemas gramaticales integran un inventario limitado, finito y cerrado, ya que el hablante no puede conscientemente crear otros nuevos. Pueden ser dependientes: siempre ligados a un morfema lexical, e independiente: aparecen solos. Por ejemplo: `{pero}`, `{ni}`, etc.

[14] Una o más letras que se agregan al comienzo, dentro, o al final de una palabra o raíz para modificar su función gramatical o para formar una nueva palabra.

Aprovechando las características inherentes a los morfemas derivativos se produce la derivación de las palabras, por ejemplo:

- De la palabra `suave` se pueden obtener: `suavizador`, `suavemente`, `suavidad`, `suavecito`, etc.
- De la palabra `camino` se pueden obtener: `caminador`, `caminar`, `caminata`, `caminante`, etc.
- De la palabra `solo` se pueden obtener: `solito`, `soledad`, `solitario`.

Aprovechando esta propiedad de la morfología derivacional, en este trabajo, se ha ideado una medida léxica, capaz de determinar con bastante exactitud el grado de semejanza entre dos palabras. También, esta distancia ha sido empleada para la determinación del tallo o raíz de las palabras. Lo que le permite ser utilizada en el agrupamiento de familias de palabras.

Vistos estos elementos, también hay que analizar otros factores que también es necesario tener en cuenta en la creación de una métrica entre palabras. En los próximos epígrafes se abundará sobre este tema.

2.5.1.2 Variación del significado en diferentes dominios.

Existen contextos en los que la sustitución de una palabra implica grandes diferencias. Por ejemplo, un número telefónico o una fecha con error en un digito pueden variar el sentido de toda la frase. Pero tal vez en otro contexto no sea muy significativo. Véase este caso, si se dice a una persona que tiene que pagar $2.0001 en lugar de $2.00, no pondrá reparo en eso ya que $2.0001 es aproximadamente $2.00, la diferencia aquí no es significativa. Pero por el contrario, si se necesita llamar a esa persona para que pague y en lugar de marcar 786-333-9069 se marca 776-333-9069, o cualquier equivocación en un dígito, por pequeña que sea, no sería igual el resultado. En este caso, las diferencias sí son significativas.

Igual situación se daría, si ocurre la sustitución de la letra `é` por la letra `e` en la palabra `enséñame`, en dependencia de contexto, no sería de tanta importancia. Si se está tratando de obtener, por ejemplo, información sobre el término en un buscador, se sabe que en muchas ocasiones algunos recursos informáticos hacen caso omiso de la acentuación con diferentes propósitos. Por el contrario, si se está evaluando un examen de español a un escolar, repercutiría en gran medida. Sin embargo, al comparar estas palabras con algunas de las medidas de similitud más utilizadas, darían una diferencia substancial. Por supuesto, que el cálculo de similitud entre palabras es un problema dependiente del dominio del discurso, pero desgraciadamente es bien difícil poder incorporarlo en un análisis de similitud entre palabras. La distancia que se está proponiendo en este trabajo intenta resolver en parte esta situación, a través de la incorporación de pesos a los caracteres. La idea es que se asume que al tener que realizar una

operación de transformación sobre caracteres con pesos muy diferentes, mayor será la penalización y por ende la distancia entre las palabras analizadas.

En ocasiones la sustitución de una letra es más común que sea ocasionada por un error tipográfico o tal vez por algún tipo de abreviatura. Sin embargo, se coincide con Bilenko en su artículo (Bilenko y Mooney, 2003) cuando plantea: "…*adapting string edit distance to a particular domain requires assigning different weights to different edit operations*", en español sería: adaptar la distancia de edición a diferentes dominios requiere asignar diferentes pesos a diferentes operaciones de edición. Este autor también obtiene un efecto importante asignando diferentes pesos a los distintos caracteres, en particular teniendo en cuenta también la operación de edición o transformación que se realiza con el carácter en cuestión. Las demás medidas analizadas no son capaces de lidiar que este problema.

Para esto, la distancia desarrollada propone colocar un peso a los caracteres, basado en su frecuencia de aparición en un diccionario del lenguaje que se esté utilizando. De esta forma, si se quiere flexibilizar el caso analizado anteriormente (`enséñame` – `enseñame`), bastaría con igualar los pesos de los caracteres `e` y `é`. De esta forma no se tendrán en cuenta las diferencias ocasionadas por la presencia o ausencia de la tilde.

También, ocurre otra situación que se manifiesta en varios lenguajes, por ejemplo el español. Este es el caso de la aceptación del cambio de caracteres y en la acentuación gráfica. En el próximo epígrafe se comentará en detalles este aspecto.

2.5.1.3 Alternancia prosódica y gráfica

Para algunos lenguajes, como el español, en las que existe la alternancia prosódica, la comparación entre palabras de una misma familia, puede aportar algunas diferencias. En los siguientes casos la Real Academia de la Lengua Española (RAE), ha aceptado ambas alternativas de alternancia prosódica como válidas, entre otras, están: afrodisiaco – afrodisíaco, atmosfera – atmósfera.

A su vez, en el español muchas palabras pueden tener diferentes escrituras aceptadas (alternancia gráfica), como las palabras: `azimut` y `acimut`, `zeta` y `ceta`. La medida que se propone en este trabajo resuelve la alternancia prosódica asignando el mismo peso a las vocales acentuadas y no acentuadas. También, basándose en el peso de los caracteres se intenta resolver la alternancia gráfica, pero sólo en los casos donde se alterne sólo un carácter que no pertenezca a la raíz de las palabras.

Por el contrario, los casos como Kiosco y Quiosco no son resueltos por el algoritmo, debido a que aquí se ha producido un cambio en el lexema o raíz de la palabra. Aunque una adaptación válida sería colocarlo

como una regla, pero esto ya haría al método dependiente del lenguaje. Al igual que en el caso anterior las demás medidas tampoco resuelven este problema.

Para este algoritmo sería muy simple tratar con caracteres extendidos que existen en muchos lenguajes. Por ejemplo los caracteres: á,à,â,ä,ãâåăă. Por el contrario otras medidas sufren grandes alteraciones en sus resultados ante la presencia de estos caracteres.

Muchos documentos electrónicos, sobre todo aquellos que se obtienen por mediación del proceso de OCR[15] (Object Character Recognition), están sujetos a poseer errores tipográficos, también algunos documentos que se toman de Internet poseen faltas ortográficas. Este elemento será analizado a continuación como uno de los posibles retos a enfrentar.

2.5.1.4 Errores ortográficos y tipográficos

En castellano existen muchas palabras de escritura dudosa, en las que observar errores ortográficos es algo muy común. El simple hecho de sustituir una letra s por una c o una z puede incrementar considerablemente la distancia entre dos palabras; sin embargo en alguno casos esto no cambia su significado, pero en otra sí. En otros casos, en muchos documentos, las diferencias son ocasionadas por errores tipográficos o de escritura de los humanos y en otros provocados por los procesos de OCR.

Por supuesto, la significación del error está en dependencia del dominio como se ha visto anteriormente. Por esta razón es necesario dar la posibilidad a las métricas de aplicar una penalización menos o más costosa en determinados casos. Consecuentemente, con las técnicas de similitud basadas en palabras, se debe penalizar con un costo que tenga en cuenta qué carácter está siendo insertado, borrado o substituido. Este elemento, aunque parece evidente, no ha sido tratado de esta forma en las métricas que se toman como comparación.

Otro elemento a considerar, sobre todo para las métricas que usan el análisis del solapamiento entre frases, ya que constituye un elemento de vital importancia en la determinación de similitud, es el análisis de la negación. Teniendo en cuenta que en este trabajo se realizan estudios sobre la similitud, específicamente en idioma inglés, el siguiente epígrafe hace un repaso de las diferentes formas de negación, particularmente para este idioma que han sido consideradas en este trabajo.

[15] Reconocimiento Óptico de Caracteres (Object Character Recognition).

2.5.1.5 Formas de negación en inglés

Según se puede ver en (Haya, 2006) las partículas usadas para negar los verbos en el Inglés son: not, no, don't, aren't, haven't, isn't, hasn't, mientras que para negar adjetivos y sustantivos son: Non, No, In y Un, en forma de prefijos.

Se utiliza don't para negar verbos. Ejemplo: I don't eat meat, que significa No como carne. Por su parte, NON es un prefijo que se utiliza para negar el significado de una palabra determinada (sustantivos o adjetivos), pero nunca de un verbo. Ejemplo: The Non-Smokers Protection Act, que significa La Ley de Protección de los no fumadores.

En otro orden de cosas, ya que se trabaja en la medición de similitud, ya sea entre palabras o entre frase, es necesario definir cómo se harán las comprobaciones de efectividad y eficiencia de estas mediciones. En el siguiente epígrafe se describen las ecuaciones generales de las que se parte en este sentido.

2.5.2 Medida de efectividad

Las medidas más utilizadas por los especialistas dedicados a las tareas de PLN son las llamadas Precisión (en inglés Precision), cobertura (en inglés Recall) y F-Medida (en inglés F-Measure). Las formulas generales de las que parten muchos investigadores son las que aparecen reflejadas en las ecuaciones 2.1, 2.2 y 2.3. Estas medidas se han ido modificando y adaptando a contexto en el que son utilizadas. En este trabajo se hace una adaptación de estas medidas para el caso particular de la recuperación y clasificación de entidades.

Estas medidas fueron utilizadas originariamente para medir la eficiencia del los sistemas de RI. Para ese contexto son determinadas de la forma siguiente:

$$precisión = \frac{documentos\ relevantes\ recuperados}{documentos\ recuperados} \quad 2.1$$

$$cobertura = \frac{documentos\ relevantes\ recuperados}{documentos\ relevantes\ en\ la\ colección} \quad 2.2$$

Donde:

$documentos\ relevantes\ recuperados$: son lo documentos de la colección que el sistema recupera y que son relevantes a una determinada pregunta del usuario.

$documentos\ recuperados:$ son todos los documentos que el sistema recupera, sean relevantes o no.

documentos relevantes en la colección: son todos los documentos que realmente son relevantes, dentro de la colección, para una determinada consulta del usuario.

Como se podrá observar el valor de los dos parámetros se encuentra entre 0 y 1. El valor del cobertura cuantifica la capacidad del sistema de identificar el mayor número de documentos relevantes, por lo que un valor de cobertura bajo, indicará que el sistema ha identificado un número reducido de documentos.

El valor de precisión cuantifica la capacidad del sistema de no presentar documentos no relevantes. Un valor de precisión bajo indica que el sistema ha identificado un gran número de documentos incorrectamente. Un valor de precisión igual a uno, quiere decir que todos los documentos recuperados son relevantes.

Otra medida utilizada es la llamada F-Medida o F-Measure en inglés (ver ecuación 2.3). El parámetro β en esta medida se utiliza para ajustar el balance entre precisión y cobertura, normalmente para dar la misma importancia a la precisión que a la cobertura se escoge $\beta=1$. Para cualquier valor de β la F-medida se encontrará en el rango de [0,1], aunque dos F-medidas con el mismo valor no indican iguales valores de precisión y cobertura (Troyano, 2004).

$$\text{F-medida}=\left(\beta^2+1\right) * \frac{\text{precisión} * \text{cobertura}}{(\beta^2 * \text{precisión}) + \text{cobertura}} \qquad 2.3$$

En los siguientes epígrafes se comenta sobre otros algoritmos y medidas ampliamente utilizadas en el Procesamiento del Lenguaje Natural. En este caso para determinar problemas de asignación y de similitud.

2.5.3 Método Húngaro de asignación de costo mínimo

El algoritmo Húngaro (Kuhn, 1955) es un algoritmo de optimización que resuelve problemas de asignación en tiempo $O(n^3)$. La primera versión conocida del método Húngaro fue publicada por Harold W. Kuhn en 1955. Este fue revisado por James Munkres en 1957, y ha sido conocido desde entonces como el algoritmo Húngaro, el algoritmo de la asignación de Munkres, o el algoritmo de Kuhn-Munkres (Jonker y Volgenant, 1987).

El algoritmo desarrollado por Kuhn está basado fundamentalmente en los primeros trabajos de otros dos matemáticos Húngaros: Dénes König y Jenő Egerváry. La gran ventaja del método de Kuhn es que es fuertemente polinómico.

Pasos para la aplicación del método húngaro:

El algoritmo para la este método es basado en la idea reflejada en (Burkard, Dell'Amico et al., 2012).

Paso 1: Primero, es necesario encontrar el elemento más pequeño en cada fila de la matriz de costos $m * m$; al restar de cada costo el costo mínimo de cada fila, se construirá una nueva matriz; luego, es preciso encontrar para esta nueva matriz, el costo mínimo en cada columna. Después, al restar de cada costo el costo mínimo de su columna, se construirá una nueva matriz (llamada matriz de costos reducidos).

Paso 2: Se traza el número mínimo de líneas (horizontales o verticales o ambas) que se requieran para cubrir todos los ceros en la matriz de costos reducidos; si se necesitan m líneas para cubrir todos los ceros, se tiene una solución óptima entre los ceros cubiertos de la matriz. Si se requieren menos de m líneas para cubrir todos los ceros, se debe continuar con el paso 3. El número de líneas para cubrir los ceros es igual a la cantidad de asignaciones que hasta ese momento se pueden realizar.

Paso 3: Ahora, es necesario encontrar el menor elemento diferente de cero (llamado k) en la matriz de costos reducidos, que no está cubierto por las líneas dibujadas en el paso 2; luego se restará k de cada elemento no cubierto de la matriz de costos reducidos y se sumará k a cada elemento de la matriz de costos reducidos cubierto por dos líneas (intersecciones). Por último se regresa al paso 2.

Paso 4: En caso de no encontrar una solución factible con los pasos anteriores aplicar entonces lo siguiente:

1. Trácese el número mínimo de líneas horizontales y verticales en la última matriz reducida que cubrirá todas las entradas cero.
2. Selecciónese el elemento no cubierto más pequeño y réstesele de todos los elementos no cubiertos; después, súmesele a todos los elementos en la intersección de dos líneas.
3. Si no es posible encontrar una asignación factible entre las entradas cero resultantes, repetir el paso. De lo contrario regresar al paso 3 para determinar la asignación óptima.

Este algoritmo será utilizado en el capítulo 3 para la asignación del camino mínimo entre los sentidos de dos palabras comparadas.

Debido a que en este trabajo se hace un fuerte uso de las distancias y medidas de similitud, tanto léxicas como semánticas, es necesario abordar algunos referentes teóricos importantes para la comprensión de lo trabajos realizados. De esto se estará tratando en los siguientes epígrafes.

2.6 Similitud Léxica y Semántica

La búsqueda de similitud entre las diferentes piezas componentes del lenguaje escrito, ha sido de gran importancia para entender el significado de los textos escritos por el hombre. Ya se ha visto que esta comparación se puede dar a nivel léxico, sintáctico o semántico. En este epígrafe se exponen los referentes teóricos sobre la medición de similitud, específicamente de las distancias y medidas de similitud léxica y semánticas. Se tratan también detalles sobre la implicación textual y la paráfrasis como casos particulares de similitud.

2.6.1 Las distancias de edición y medidas de similitud léxica

En ocasiones se tienden a confundir los términos distancia y similitud, pero son dos cosas diferentes como se verá a continuación. Existen muchos otros tipos de relaciones que pueden existir entre unidades léxicas, para este trabajo solo se explican los términos y conceptos asociados a las distancias métricas y la similitud propiamente dicha.

En este sentido, hay que ser especialmente cuidadosos al utilizar el término distancia, ya que la acepción matemática del término le confiere unas propiedades específicas. Se prefiere hablar en general de divergencia y reservar el término distancia al caso en que se satisfacen las propiedades métricas.

Al hablar de distancia obligatoriamente se piensa en la acepción matemática del término, específicamente esto es lo que la diferencia de la similitud. Rodríguez en (Martí, Fernández et al., 2003) expone claramente las reglas que tienen que cumplirse para poder hablar de una distancia métrica como tal.

Una función de distancia D con valores reales no negativos, definida en el producto cartesiano $X \bullet X$ del conjunto X es llamada una métrica de X, si para cada valor $x, y, z \in X$ se cumple:

- $d(x,y) \geq 0$
- $d(x,x) = 0$
- $d(x,y) > 0$ cuando $x \neq y$
- $d(x,y) = d(y,x)$ (simetría)
- $d(x,z) \leq d(x,y) + d(y,z)$ (desigualdad triangular)

Se le da el nombre de distancia o disimilitud entre dos individuos (x) y (y) a una medida, indicada por $d(x,y)$, esta mide el grado de desigualdad, entre ambos individuos, en relación a un cierto número de características. El valor de $d(x,y)$ es siempre un valor no negativo y cuanto mayor sea, mayor será la diferencia entre los individuos (x) y (y).

Rodríguez, en este mismo artículo plantea: "…*para muchas aplicaciones el uso de una distancia no es realmente necesario y podemos utilizar el concepto más laxo de divergencia (y su opuesto, similitud)*" (Martí, Fernández et al., 2003).

Existe una relación evidente entre una medida de similitud y distancia. Para convertir una medida en otra, comúnmente se utiliza la transformación 2.4.

$$sim_{dist}(A,B) = \frac{1}{1 + dist(A,B)} \qquad 2.4$$

A continuación se describen algunos trabajos vinculados a la creación de distancias de edición y medidas de similitud que guardan relación con el trabajo realizado.

2.6.1.1 Trabajos relacionados con la Distancia Extendida

Ya que la métrica propuesta, así como otras que le anteceden, parte de la Distancia de Edición (DE) o Distancia de Levenshtein (Levenshtein, 1966), es necesario describir en detalle cada una de sus características.

La distancia de Levenshtein es un valor numérico que establece la diferencia entre dos palabras. Wagner y Fisher en su trabajo titulado "*The string-to-string correction problem*" (Wagner y Fisher, 1974) la utilizaron para evaluar la distancia entre dos cadenas de caracteres y se basaron el método de la Programación Dinámica (PD) (Kaufmann y Cruon, 1967). Este algoritmo involucra el uso de una matriz de enteros de `(n+1)x(m+1)`, donde `n` y `m` son las longitudes de las cadenas. Cuando la matriz esté completamente llena, la celda inferior derecha contendrá el costo mínimo de la transformación para convertir una palabra en la otra. A continuación se muestra una implementación iterativa de este algoritmo:

```
Entero Dist_Levenshtein (carácter s[1..m], carácter t[1..n])
{
 //para todo i y j, d [i, j] almacenará la distancia Levenshtein entre los
 //primeros i caracteres de s y de los primeros j caracteres de t, téngase en
 //cuenta que d tiene (m +1) * (n +1) valores.

 declarar Entero d[0..m, 0..n]

 //borrar todos los elementos en d haciendo cero cada elemento

 para i desde 1 hasta m {
   d[i, 0] := i
 }

 para j desde 1 hasta n {
   d[0, j] := j
 }
```

```
 para j desde 1 hasta n {
   para i desde 1 hasta m {
    si s[i] = t[j]
     entonces
       d[i, j]:= d[i-1, j-1]
     si no
       d[i, j]:= mínimo
           (
            d[i-1, j] + 1, // para borrado
            d[i, j-1] + 1, // para inserción
            d[i-1, j-1] + 1 // para substitución
           )
   }
 }

 returnar d[m, n]
}
```

A pesar de ser una distancia ampliamente difundida, como ya se ha explicado no siempre se obtiene un resultado favorable. Por ejemplo al analizar las cadenas `luego` y `juego` con el algoritmo de Levenshtein la distancia entre ellas sería uno (1), sin embargo, las cadenas `campo` y `campesino`, semánticamente más cercanas, darían una distancia de cuatro (4), este valor producto de las transformaciones de eliminación de las letras `e`, `s`, `i`, `n`. Como este algoritmo no tiene en cuenta dónde se producen los cambios, no puede determinar que las palabras `luego` y `juego` no tienen una relación semántica, ya que han existido cambios en el lexema o raíz de la palabra.

Por otro lado, las palabras `campo` y `campesino` comparten la raíz `camp`, por lo que debería existir una menor distancia entre ellas.

Unos años más tarde, Needleman y Wunsch extendieron el algoritmo de la DE obteniendo una mayor similitud sensitiva (Neeedleman y Wunsch, 1970). Ellos usaron penalizaciones lineales, permitiendo secuencias de caracteres erróneos en el alineamiento de dos cadenas. Al final, lo que hacen es adicionar un costo a las operaciones de inserción y borrado.

Luego, en 1981 surge una nueva modificación de la distancia de Levenshtein descrita en (Smith y Waterman, 1981). Esta fue creada para identificar alineamientos óptimos entre cadenas de ADN y secuencias de proteínas. Al igual que la anterior, también utilizó costos de penalización por errores para inserción y borrado. A diferencia de Needleman y Wunsch, la similitud final es computada a partir del máximo valor al que se le resta el costo de penalización.

Posteriormente, surge una modificación a la distancia de Smith y Waterman realizada por Gotoh en el año 1982. Esta versión permite que existan caracteres no alineados en la secuencia analizada, a lo que ellos llaman affine gaps within the sequence (Gotoh, 1982).

Pocos años después, surge la métrica de Jaro, esta métrica condiciona que dada dos cadenas s_1 y s_2 la distancia d_j entre ellas se puede través de la ecuación 2.5 (Jaro, 1989):

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \quad 2.5$$

Donde:

d_j – distancia según (Jaro, 1989)

m - es el número de caracteres que machean

t - es el número de transposiciones

Luego en el años 1999, surge la llamada distancia de Jaro-Winkler (Winkler, 1999), esta medida propone que dos caracteres desde s_1 a s_2 respectivamente, son considerados como coincidentes sólo si no están tan lejos como:

$$\left[\frac{max(|s_1|,|s_2|)}{2}\right] - 1 \quad 2.6$$

Para conceder valores más adecuados a los caracteres que machean desde el inicio para un determinado prefijo l, Jaro y Winkler usaron el prefijo p

$$d_w = d_j + \left(lp\left(1 - \mathrm{d_j}\right)\right) \quad 2.7$$

Donde:

d_w - es la distancia según (Winkler, 1999)

d_j - es la distancia de Jaro entre las cadenas s_1 y s_2

p.- es la longitud del prefijo común al inicio de la cadena hasta el máximo de cuatro caracteres.

l – es un factor de escala constante para indicar cuánto se ajusta el valor por tener prefijos comunes. El valor estándar para esta constante es indicado en el trabajo de Winkler como $p=0.1$

Otra métrica muy utilizada es el índice de Jaccard, también conocido como el coeficiente de similitud de Jaccard (originalmente acuñado coefficient de communauté por Paul Jaccard), es una estadística que se usa para comparar la similitud o la divergencia de conjuntos de muestra. Se define como el tamaño de la intersección dividido entre el tamaño de la unión de la muestra (Jaccard, 1908) (ver ecuación 2.8):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad 2.8$$

La distancia de Jaccard, que mide la disimilitud entre conjuntos de muestra, es complementaria al coeficiente de Jaccard (ver ecuación 2.9) y se obtiene restándole uno al coeficiente de Jaccard, o, de manera equivalente, dividiendo la diferencia de los tamaños de la unión y la intersección de los dos conjuntos por el tamaño de la unión:

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cap B| - |A \cup B|}{|A \cup B|} \quad 2.9$$

Otra distancia que obtiene muy buenos resultados es la llamada QGrams Distance. Esta medida genera `grams`, subconjuntos de una cadena, a través del `desplazamiento` de una longitud `q`. O sea, los `q-grams` (n-gramas) no son más que subcadenas de longitud `q` de una palabra dada; el concepto fue utilizado por primera vez por Shanno en su artículo: A mathematical theory of communication (Shannon, 2001).

Considerando el siguiente ejemplo, los `q-grams` de longitud `q=3` para la cadena `sam chapman` son: *f*`(1,##s), (2,#sa), (3,sam), (4,am ), (5,m c), (6, ch), (7,cha), (8,hap), (9,apm), (10,pma), (11,man), (12,an%), (13,n%%),` donde `#` y `%` indican el inicio y el fin de las cadenas respectivamente. Ejemplo tomado de (Gravano, Ipeirotis et al., 2001).

Como resultado la distancia es basada en el conteo del número de ocurrencias de diferentes `q-gram` en las dos cadenas; las cadenas están más relacionadas mientras más `q-grams` en común tengan.

Desde hace varios años, otra distancia que ha sido muy utilizada es la llamada Block Distance, considerada como Taxicab geometry por Hermann Minkowski en el siglo 19. Es una forma de geometría en la que se sustituye la métrica habitual de la geometría euclidiana por una nueva métrica en la que la distancia entre dos puntos es la suma de las diferencias (absoluta) de sus coordenadas. Representa las entidades (strings) como vectores en el espacio y la distancia entre los vectores. Esta métrica es también llamada L1 Distance, Manhattan o City Block Distance. La métrica es calculada a partir de la suma de las distancias en los ejes.

Definición: La distancia entre dos puntos medida a lo largo del eje X en ángulo recto. En un plano con p_1 en (x_1, y_1) y p_2 en (x_2, y_2), sería $|x_1 - x_2| + |y_1 - y_2|$ (Krause, 1987).

De forma general quedaría de la forma siguiente:

$$d_{ij} = \sum_{k=1}^{n} |X_{ik} - X_{jk}| \quad 2.10$$

Con el objetivo de medir -en estudios ecológicos- el grado en que dos especies están asociadas en la naturaleza Lee R. Dice utiliza la medida llamada Dice's coefficient (Dice, 1945), también conocido como Sorensen-Dice coefficient: es una medida de similitud entre conjuntos. Cuando esta medida es usada para comparar cadenas de caracteres, el coeficiente puede ser calculado a partir de dos secuencias de caracteres *x* y *y* usando bi-gramas. Se calcula de la siguiente manera:

$$s = \frac{2 * n_t}{n_x + n_y} \quad 2.11$$

Donde:

n_t es el número de caracteres bi-gramas encontrados en ambas cadenas, n_x el número de bi-gramas en la cadena *x* y n_y es el número de bi-gramas en la cadena *y*.

Tal vez por su sencillez, una de las distancias más comúnmente utiliza es la llamada Distancia Euclidiana. En la mayoría de los casos cuando los investigadores comparan una distancia se refieren a la Distancia Euclidiana. Se define como la raíz de la sumatoria de las diferencias cuadradas entre las coordenadas de un par de objetos.

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (X_{ik} - X_{jk})^2} \quad 2.12$$

Otra métrica, también muy utilizada en el PLN es la medida del Coseno, es una métrica de similitud calculada a partir del coseno del ángulo entre dos vectores. No es considerada una distancia, pues no cumple con la desigualdad triangula, es por ellos que se dice que es una medida de similitud. El coseno de cero grado (0°) es uno (1), y es inferior a uno (1) para cualquier otro ángulo; el menor valor del coseno es menos uno (-1). El cálculo del coseno entre dos vectores se puede realizar si ambos están orientados en la misma dirección. Por esta razón esta medida es un veredicto de orientación y no una magnitud en sí. Por ejemplo, dos vectores que se encuentran con una orientación entre ellos de 90°, no serán considerados similares y su valor de similitud es cero (0). Por el contrario si los vectores tienen la misma orientación tendrán una similitud de uno (1). Mientras que dos vectores que sean opuestos totalmente tendrán una similitud de menos uno (-1).

Comúnmente, se utiliza la similitud del coseno sobre todo en un espacio positivo, donde el resultado se limita al intervalo [0,1]. Por ejemplo, en la Recuperación de la Información, a cada término se asigna teóricamente una dimensión diferente y un documento es caracterizado por un vector, donde el valor de cada dimensión se corresponde con el número de veces que el término aparece en ese documento. Luego se calcula el valor del coseno entre una pregunta del usuario y los documentos almacenados, de esta forma se recuperan aquellos con mayor relevancia, o sea, los que forman un menos ángulo con la pregunta.

Una métrica que por su importancia y utilidad en muchos trabajos se describe, es la creada por los investigadores Álvaro E. Monge y Charles P. Elkan, proponen el siguiente esquema de emparejamiento recursivo para comparar dos cadenas A y B. En primer lugar, A y B se dividen en sub-cadenas A = $a_1 \ldots a_A$ y B= $b_j \ldots b_B$. De esta forma, la similitud según (Monge y Elkan, 1996) se define como:

$$match(s,t) = \frac{1}{|A|}\sum_{i=1}^{|A|} \max_{j=1}^{|B|} match(A_i, B_j) \qquad 2.13$$

El algoritmo de coincidencias recursivo (matching Coefficient) de Monge y Elkan tiene complejidad cuadrática. Dado A y B, cada subcampo en A debe ser comparado con cada subcampo en B. En el nivel más bajo, cada cadena atómica de A se compara con cada cadena atómica de B.

Otra métrica, muy utilizada en el análisis de redes sociales es el Coeficiente de Solapamiento, más conocido como Overlap Coefficient (Matsuo, Tomobe et al., 2004), es una medida de similitud muy relacionada con el índice Jaccad. Esta medida computa los solapamientos entre dos conjuntos. Se define de la manera siguiente:

$$overlapCoefficient\ (X,Y) = \frac{|X \cap Y|}{min(|X|,|Y|)} \qquad 2.14$$

Si el conjunto X es un subconjunto de Y o viceversa, entonces el coeficiente de solapamiento será igual a uno (1).

Una métrica muy sencilla, que ha sido utilizada en esta investigación es el coeficiente de emparejamiento simple, conocido en inglés como Simple Matching Coefficient (SMC), fue propuesto por (Sokal y Michener, 1958) para usarlo en una taxonomía numérica, pero también ha sido utilizado por (Kaesler, 1966) en la comparación de bio assemblages. Se puede utilizar, por supuesto, sólo cuando se comparan conjuntos con tres o más variables a contrastar (Sokal y Michener, 1958). En la Tabla 2.1 se muestra el esquema de asignación para esta métrica.

Tabla 2.1. Esquema de asignación del Simple Matching Coefficient *.

	Objeto 1	
Objeto 2	*Número de variables con categoría 1*	*Número de variables con categoría 2*
Número de variables con categoría 1	*a*	*b*
Número de variables con categoría 2	*c*	*d*

* extraído de (Mooi y Sarstedt, 2011)

El cálculo en cuestión se realiza a través de la ecuación 2.15 que se muestra a continuación:

$$SMC = \frac{a + d}{a + b + c + d} \qquad 2.15$$

Otras dos medidas muy simples, ya que se basan en la longitud de las cadenas comparadas son las llamadas Chapman Length Deviation y Chapman Mean Length. A continuación se muestran los dos procedimientos para el cálculo de estas distancias, extraídos de la documentación de SimMetric (Chapman y Parkinson, 2006):

Chapman Length Deviation:

```
public override double GetSimilarity(string firstWord, string secondWord) {
 if ((firstWord != null) && (secondWord != null)) {
    double firstLength = firstWord.Length;
    double secondLength = secondWord.Length;
    if (firstLength >= secondLength) {
       return secondLength/firstLength;
    }
    else {
       return firstLength/secondLength;
    }
 }
 return 0.0;
}
```

Chapman Mean Length:

```
defaultMismatchScore = 0.0;
defaultPerfectScore = 1.0;
public override double GetSimilarity(string firstWord, string secondWord) {
    if ((firstWord != null) && (secondWord != null)) {
       double bothLengths= secondWord.Length + firstWord.Length;
       if (bothLengths > chapmanMeanLengthMaxString) {
          return defaultPerfectScore;
       }
       else {
          double oneMinusBothScaled=(chapmanMeanLengthMaxString-
          othLengths)/chapmanMeanLengthMaxString;
          return defaultPerfectScore-
          oneMinusBothScaled*oneMinusBothScaled*oneMinusBothScaled*oneMinusBothScaled
          ;
       }
    }
```

```
        return defaultMismatchScore;
    }
```

Resumiendo sobre el comportamiento de estas distancias, se puede plantear que Needleman y Wunch, así como la modificación realizada por Gotoh, tienen en cuenta penalizaciones y permiten secuencias de caracteres erróneos en el alineamiento, pero este factor es estático y además no tiene en cuenta la importancia del tipo de transformación que se realiza. No deberían considerarse con igual importancia todas las operaciones. Tampoco es igual la penalización que habría que asignar por el cambio de una vocal sin acento por una acentuada, comparado con la sustitución de dos consonantes totalmente diferentes. Ejemplo: Para el español, la comparación de las palabras `enseñar` con `enseñár` y `enseñar` con `ensekar`, deben recibir una penalización diferente y mayor la sustitución de la `ñ` por la `k`, que la vocal acentuada `á` por la vocal `a` sin acento.

El algoritmo de Jaro-Winkler (Winkler, 1999) en su mejora a la distancia de Jaro (Jaro, 1989), plantea el tratamiento especial de las transformaciones que se producen en el ámbito de una longitud fija de prefijo. Mediante una constante se ajusta favorablemente la distancia, para aquellos que compartan el mismo prefijo. Pero, precisamente el problema radica en que esta métrica adopta una longitud fija del prefijo y un factor de escala estático, lo que haría fallar al algoritmo en los casos donde las palabras en análisis tuviesen un prefijo que no cumpliese con el valor preestablecido por el algoritmo.

Lo mismo ocurre con las demás distancias analizadas, no tienen en cuenta las aspectos morfológicos de la lengua, por lo que están ajenas y son vulnerables a los fenómenos de la flexión, la derivación y composición de las palabras.

Para la evaluación y comparación de estas distancias se usa la API (Application Program Interface) SimMetrics library v1.5 para .NET 2.0[16]. Esta librería contiene las métricas siguientes:

1. BlockDistance: una implementación de la métrica Block Distance, conocida también como la métrica Taxicab, Rectilinear Distance, L1 Distance, City Block Distance, Manhattan Distance, o Manhattan Length.
2. ChapmanLengthDeviation: implementación de la métrica determinada por la diferencia entre las longitudes de las cadenas analizadas.
3. ChapmanMeanLength: Implementa el algoritmo de longitud media de Chapman, proporcionando una medida de similitud entre dos cadenas a partir del tamaño de la longitud media de los vectores.

[16] Copyright © 2006. Chris Parkinson disponible en http://www.sourceforge.com.

4. CosineSimilarity: implementa la métrica de Similitud del Coseno.
5. DiceSimilarity: implementa la medida de Similitud de Dice.
6. EuclideanDistance: implementa la métrica de la Distancia Euclidiana.
7. JaccardSimilarity: implementa la medida de similitud de Jaccard
8. Jaro: implementa la medida de Jaro entre cadenas de caracteres.
9. JaroWinkler: implementa la modificación a la métrica de Jaro que realiza Winkler.
10. Levenstein: implementa la distancia de edición de Levenshtein.
11. MatchingCoefficient: implementa la métrica del Coeficiente de coincidencias.
12. MongeElkan: implementa la métrica de Monge-Elkan.
13. NeedlemanWunch: implementa la distancia de edición de Needleman y Wunch.
14. OverlapCoefficient: Implementa la métrica del coeficiente de solapamiento.
15. QGramsDistance: implementa la métrica QGram usando el tokenizador QGRam.
16. SmithWaterman: implementa la distancia de edición de Smith-Waterman.
17. SmithWatermanGotoh: implementa la extensión Gotoh de la distancia de Smith-Waterman.
18. SmithWatermanGotohWindowedAffine: implementa la extensión Gotoh de la distancia de Smith-Waterman incorporando deficiencias afines en las cadenas

Hasta este punto todas las métricas analizadas son absolutamente léxicas. En el siguiente epígrafe se hace un recorrido por los diferentes aspectos de la medición de la similitud semántica.

2.6.2 Similitud semántica

En este punto, es preciso destacar que en la tarea del entendimiento de un texto, no solo bastaría con el análisis de la similitud léxica entre sus palabras. Sin embargo, tampoco se puede ver la semántica aislada totalmente del análisis léxico. Como bien se plantea en (Glickman, Shnarch et al., 2006), es común ver que el alineamiento semántico se aborda a nivel léxico. En este nivel, el objetivo es identificar si el significado de un elemento léxico de un texto se expresa también en el otro.

Se coincide con Corley y Mihalcea en su artículo (Corley y Mihalcea, 2005) cuando asegura que este proceso se determinación de similitud, en este caso léxica, falla al intentar capturar la similitud semántica. Este planteamiento de Corley y Mihalcea puede ser mejor entendido a través del siguiente ejemplo. Véase que existe una gran similitud semántica entre las frases (1): `Los cetáceos son grandes`

mamíferos y la frase (2): Una ballena es un mamífero grande. Sin embargo, los métodos léxicos son incapaces de poder identificar esta similitud. Simplemente podrían equiparar los términos mamíferos con mamífero y grandes con grande, pero serían incapaces de identificar la similitud entre cetáceos y ballena; por lo que adjudicarían cierto valor de disimilitud a estas frases por las coincidencias encontradas, pero no la similitud real que existen entre las frases.

A juicio de este autor, el éxito, y no completo, estaría en la posibilidad de interpretar la similitud léxica, pero también semántica del texto en análisis. Y véase que se ha remarcado éxito no completo, pues hay más de un ejemplo para demostrar que el conocimiento del contexto (pragmática) es, en mucho caso, también es de vital importancia. Por ejemplo, en la frase: Ya se ha hecho muy tarde, desde el punto de vista semántico, tienen un significado muy claro de interpretar; sin embargo, pragmáticamente podría tener -dependiendo del contexto- otros sentidos (no puedo quedarme, no quiero quedarme, me quiero ir, etc.).

La medición de la similitud semántica entre textos está siendo muy utilizada en las aplicaciones de PLN. La idea de la medición semántica consiste en derivar automáticamente una puntuación que indica la similitud a nivel semántico entre dos segmentos de texto. Aunque no se puede descartar que una medida global de similitud semántica debe tener en cuenta tanto las relaciones entre palabras, como las distintas entidades que participan en la interacción descrita por cada uno de los dos textos analizados (Corley y Mihalcea, 2005).

2.6.2.1 Trabajos relacionados con la determinación de similitud semántica

En (Corley y Mihalcea, 2005) se intenta modelar la similitud semántica de los textos como una función de la similitud semántica de las palabras componentes. Esto lo realizan mediante la combinación de métricas de similitud palabra a palabra y modelos de lenguaje, en una fórmula que constituye un indicador de la similitud semántica de los dos textos de entrada.

En realidad, se podrían dividir en dos grandes grupos los sistemas que intentan medir la similitud en este sentido. En el primero estaría aquellos que intentan usar la medición palabra a palabra -o mejor- para ser exacto, entre conceptos. Por otro lado, estarían aquellos que buscan la similitud a partir de mediciones en redes semánticas.

Entre los trabajos más representativos en la medición de la similitud semántica se encuentran (Leacock y Chodorow, 1998), (Lesk, 1986), (Wu y Palmer, 1994), esta medidas están basadas en la longitud de los caminos entre conceptos. Utilizan la taxonomía de WordNet (Miller, Beckwith et al., 1990) para realizar sus cálculos de similitud.

En (Lesk, 1986) la similitud entre dos conceptos es una función que mide el solapamiento entre las correspondientes definiciones de estos conceptos. Este algoritmo fue propuesto para la determinación del sentido de las palabras.

De manera diferente, en (Leacock y Chodorow, 1998) se propone una medida que intenta encontrar el camino más corto entre dos conceptos y originan ese valor buscando el camino de longitud máxima en la jerarquía `es-un` en la que ocurre. (Ver la ecuación 2.16 para más detalle)

$$Sim_{lch}(c_1, c_2) = -\log \frac{long(c_1, c_2)}{2 * D} \qquad 2.16$$

Donde:

$Sim_{lch}(c_1,c_2)$ – es valor de similitud entres dos conceptos (synsets[17] de WordNet) según (Leacock y Chodorow, 1998).

c_1, c_2 – son el concepto 1 y el concepto 2 para calcular su similitud.

$long\,(c_1,c_2)$- es la longitud del menor camino entre los synsets.

D- es la profundidad de la taxonomía.

Otro trabajo, publicado en (Wu y Palmer, 1994), intenta encontrar la longitud del camino al nodo raíz desde el mínimo común incluido (en inglés: least common subsume (LCS)) de los dos conceptos, el cual será el concepto más específico que comparten con un ancestro. El LCS viene a ser el primer concepto de la taxonomía que contiene a las dos palabras, o sea, el primer nodo común para el que se puede encontrar un camino entre dichas palabras. Este valor se origina mediante la suma de las longitudes de la trayectoria de los conceptos individuales a la raíz. (ver la ecuación 2.17 para más detalle)

$$Sim_{wp}(c_1, c_2) = \frac{2 * prof(LCS)}{prof(c_1) + prof(c_2)} \qquad 2.17$$

Donde:

$Sim_{wp}(c_1,c_2)$ – es el valor de la similitud entre los conceptos c_1 y c_2 según (Wu y Palmer, 1994)

$prof(LCS)$ – es la profundidad en la taxonomía del LCS.

$prof(c_1)$ – es la profundidad en la taxonomía del concepto c_1.

$prof(c_2)$ – es la profundidad en la taxonomía del concepto c_2.

[17] Un synset es

Otros trabajos como (Resnik, 1995), (Lin, 1998), (Jiang y Conrath, 1997), se basan en la información de contenidos. Son medidas, basada en corpus, de la especificidad de un concepto. Según se plantea en (Resnik, 1995) (ver ecuación 2.20), la similitud es computada partiendo del ancestro común más cercano y la distancia de las dos entidades desde el nodo raíz. Tanto en (Lin, 1998) como en (Jiang y Conrath, 1997) se argumenta la información de contexto del LCS de los dos conceptos, con la suma de la información de los dos conceptos individualmente. En (Lin, 1998) se contabiliza la información de contexto del LCS a través de esta suma (ver ecuación 2.18). También en (Jiang y Conrath, 1997) (ver ecuación 2.19) substraen la información de contexto del LCS de esta suma y toman el inverso para transformar una distancia en una medida de similitud.

$$Sim_{lin}(c_1, c_2) = \frac{2 * IC(LCS)}{IC(c_1) + IC(c_2)} \quad 2.18$$

Donde:

$Sim_{lin}(c_1,c_2)$ – es el valor de la similitud entre los conceptos c_1 y c_2 según (Lin, 1998).

$IC(LCS)$ – es la información de contexto en la taxonomía del LCS.

$IC(c_1)$ – es la información de contexto en la taxonomía del concepto c_1.

$IC(c_2)$ – es la información de contexto en la taxonomía del concepto c_2.

$$Sim_{jc}(c_1, c_2) = \frac{1}{IC(c_1) + IC(c_2) - 2 * IC(LCS)} \quad 2.19$$

Donde:

$Sim_{jc}(c_1,c_2)$ – es el valor de la similitud entre los conceptos c_1 y c_2 según (Jiang y Conrath, 1997).

$IC(LCS)$ – es la información de contexto en la taxonomía del LCS.

$IC(c_1)$ – es la información de contexto en la taxonomía del concepto c_1.

$IC(c_2)$ – es la información de contexto en la taxonomía del concepto c_2.

La medida de similitud utilizada por Resnik se muestra en la ecuación 2.20.

$$Sim_{resk}(c_1, c_2) = \max_{c \in S(c_1,c_2)} [-\log p(c)] \quad 2.20$$

$$p(c) = \frac{freq(c)}{N} \quad 2.21$$

Donde:

$Sim_{resk}(c_1,c_2)$ - es el valor de la similitud entre los conceptos (c_1 y c_2) según (Resnik, 1995).

$S(c_1,c_2)$ - es el conjunto de conceptos que incluyen a (c_1 y c_2).

$frep(c)$ - es el número de ocurrencias del concepto en la jerarquía.

N - es el total de conceptos.

Una de las tareas que necesita de la medición de similitud es la implicación textual. En el siguinete epígrafe se amplía sobre esta temática.

2.6.3 La implicación textual como un caso particular de determinación similitud

La implicación textual puede verse como un caso muy particular de similitud unidireccional entre frases, en el que la verdad de una de ellas es implicada por la verdad de la otra, no siendo así el caso contrario.

En (Dagan, Glickman et al., 2005) se define la implicación textual como una relación direccional entre pares de expresiones de texto, denotada por T - el texto y H- la hipótesis. Se dice que T implica H, si el significado de H se puede inferir a partir del significado de T, como sería normalmente interpretado por las personas.

La relación de implicación puede ser de diferente naturaleza. De ello depende la precisión que obtienen los diferentes sistemas utilizados para reconocer este tipo de relación. Por ejemplo, puede darse el caso en que la relación de implicación entre las frases sea léxica, este caso podría resolverse simplemente con una relación de los solapamientos entre ambas frases. Véase el siguiente caso del par con id=836, extraído del corpus annotated.xml del RTE1.

<t>A Union Pacific freight train hit five people.</t>. En español: Un tren de carga de Unión Pacifico golpeó a cinco personas.

<h>A Union Pacific freight train injured five people.</h>. En español: Un tren de carga de Unión Pacifico lesionó a cinco personas.

Las únicas palabras que no coinciden son hit y struck, pero como existe un alto grado de solapamiento entre las frases, la mayoría de los sistemas que basan su funcionamiento en el léxico, lo detectarán como relación de implicación.

Igual sucedería si la hipótesis fuese:

<h> Five people were beaten by a Union Pacific freight train.</h>. En español: Cinco personas fueron golpeadas por un tren de carga de Unión Pacifico.

En este caso, aunque los componentes de la oración han cambiado, sigue existiendo el mismo nivel de solapamiento. Por lo que también resultaría relativamente fácil descubrir el relacionamiento entre estas frases.

Pero por el contrario, si la comparación fuese con una hipótesis como la siguiente:

<h> Five pedestrians were affected by a big rolling monster from a freight railroad company.</h>.

En español: Cinco peatones se vieron afectados por un gran monstruo rodante de una compañía de ferrocarril de carga.

Sería extremadamente difícil para un sistema que solamente se base en elementos léxicos, descubrir el relacionamiento que existe entre las frases. Esto estaría motivado por las diferencias lexicográficas entre people y pedestrians, affected con cualquiera de las variantes: hit, injured, beaten; además de que no podría relacionar big rolling monster con train, ni tampoco Union Pacific con freight railroad company, este análisis no sería realizado eficientemente por un simple emparejamiento de palabras.

Por esta razón, se hace imprescindible la incorporación de elementos de la semántica para resolver muchos de los casos que pueden presentarse.

Para entender mejor esta problemática se analizan en el siguiente epígrafe algunos de los trabajos relacionados con esta compleja temática, resaltando de cada uno de ellos los resultados alcanzados.

2.6.3.1 Trabajos relacionados con el Reconocimiento de Implicación Textual

El Reconocimiento de la Implicación Textual es una temática relativamente nueva, al menos desde el punto de vista de la lingüística computacional. A criterio de este investigador, el primer trabajo en relación con el RIT que se ha podido encontrar es el titulado Light-Weight Entailment Checking for Computational Semantics (Monz y de Rijke, 2001). En este trabajo se presenta un enfoque de inferencia semántica que funciona en representaciones mínimas que pueden ser generadas fácilmente en dominios arbitrarios. Proporcionan salidas graduadas en lugar de decisiones estrictas de sí / no. Para esto utilizan la conocida medida `idf` (inverse document frequency).

Con relación a este tema, desde el año 2004 al 2011 han sido celebradas varias versiones de competición para el RIT. A criterio muy personal de este autor, la más importante han sido las series de conferencias del Recognizing Textual Entailment Challenge. Las tres primeras competiciones se llevan a cabo a través

de la European PASCAL Network of Excellence, las subsecuentes formaron parte de las NIST (Text Analysis Conference).

La primera de las conferencias se efectúa desde junio del 2004 hasta abril del 2005 y se llamó Pascal Recognizing Textual Entailment Challenge[18], comúnmente conocida como RTE1 (Dagan, Glickman et al., 2005). Es el primer intento por crear una tarea independiente para el RIT. Se evalúan 17 variantes de diferentes grupos. Los resultados en cuanto a exactitud fueron discretos, alcanzándose valores entre un 50% y un 60% de exactitud. En esta competición el trabajo más destacado fue (Delmonte, Tonelli et al., 2006), en el que logran un 60% de exactitud utilizando para ellos WordNet (Miller, Beckwith et al., 1990), emparejamiento sintáctico e inferencia lógica.

La segunda edición del Pascal Challenge[19], RTE2 (Bar-Haim, Dagan et al., 2006), se realiza desde octubre del 2005 hasta abril de 2006. Ya en esta versión se presentan 23 equipos y los resultados oscilan entre un 53% y un 75% de exactitud. Son de destacar aquí, los trabajos (Adams, Nicolae et al., 2007) quienes obtienen un 62% de exactitud solo utilizando un alineamiento léxico; también (Tatu, Iles et al., 2006) que obtienen un 73% y finalmente el trabajo (Hickl, Williams et al., 2006) quienes, utilizando una gran cantidad de recursos, logran el mejor resultado de la competición con un 75% de exactitud, un 15% superior a lo que se había logrado en el RTE1.

Luego, en junio del 2007 se lleva a cabo la tercera competición, RTE3 (Giampiccolo, Magnini et al., 2007). En esta participan 26 equipos y se envían 44 corridas. A diferencia de las anteriores, en esta se introducen frases más largas con el objetivo de acercarse más a la realidad esta tarea. Además, se ofrece un repositorio de recursos, por lo que se fomenta el uso compartido de estas herramientas. También se crea una tarea piloto con el objetivo de distinguir entailments desconocidos, a partir de identificar contradicciones y proporcionar justificaciones de las decisiones globales del sistema. A esta tarea se le llamó Extending the Evaluation of Inference Texts (Giampiccolo, Magnini et al., 2007). Ya en esta competición se alcanzan resultados de exactitud que oscilan entre un 49% y un 80%, aunque la mayoría de los trabajos estuvieron entre 59% and 66%. Al igual que en el año anterior el trabajo (Hickl y Bensley, 2007) obtiene el mejor resultado alcanzando un 80% de exactitud, un 8% más que el segundo lugar alcanzado por (Tatu y Moldovan, 2007).

En el 2008, por primera vez un RTE es propuesto como parte de las conferencias TAC (Text Analysis Conference)[20], aunque su nombre se designa como una continuación de los anteriores, RTE4. Otra

[18] http://www.pascal-network.org/Challenges/RTE/

[19] http://www.pascal-network.org/Challenges/RTE2

[20] http://www.nist.gov/tac/

innovación importante introducida en esta edición es la llamada three-judgment task. Esta variante exige que los sistemas, además de la clásica decisión de si/no de las ediciones anteriores, hagan distinción entre pares donde no hay implicación, debido a que el contenido de H contradice del contenido de T. Y también, se distinguen los pares donde la implicación no se puede determinar porque la verdad de H no se puede verificarse en el contenido de T. De forma resumida esta variante se nombró 3-way task y a la forma tradicional se le llamó 2-way task.

La distribución de acuerdo a esta forma de anotación, tanto en la configuración individual como la global fue 50% Entailment, 35% Unknown, 15% Contradiction. En esta edición participan 26 equipos, siete solo participan en la 3-way task, doce solo en 2-way task y siete en ambas tareas.

En esta competición, en la tarea 3-way, el mejor valor de exactitud fue 68.5% y el más bajo 30.7%, obteniéndose un resultado promedio de un 50.65%. Esto demuestra que la tarea constituyó todo un reto. La tarea 2-way obtuvo, aunque bajos, mejores resultados oscilando la precisión alcanzada entre 49.7% y un 74.6%; generándose un exactitud promedio de 58.03%. Esto resultados no pueden ser comparados estrictamente con los de años anteriores debido a que los corpus utilizados son muy diferentes.

Haciendo un análisis de los recursos utilizados en esta competición, WordNet (Miller, Beckwith et al., 1990), EuroWordNet (Vossen, 1997) y finalmente eXtended WordNet[21] (Moldovan y Rus, 2001), afloran como los más utilizados por los participantes. Las técnicas de aprendizaje automático, en especial SVM, también se destacan como las más utilizadas.

Más tarde, en noviembre del 2009, se celebra la quinta conferencia para el Reconocimiento de la Implicación Textual, RTE5. Además de las ya tradicionales dos tareas 2-way y 3-way, que se mantienen como tareas principales, se incorporan en esta versión las pruebas de ablación sobre recursos de conocimiento utilizados por los sistemas que participan en la tarea principal. La tarea piloto de búsqueda es situada en el contexto de aplicaciones de resúmenes. Se introduce una nueva modificación, pues la tarea de reconocer la implicación dentro de un corpus consiste en encontrar todas las frases en un conjunto de documentos que impliquen una hipótesis dada.

En esta competición del RTE5, participan 21 equipos con 54 corridas para la tarea principal y 8 para la tarea piloto de búsqueda. El mejor resultado alcanzado en la tarea 3-way fue de 68.33% de exactitud, mientras que el más bajo fue 43.83%. Para la tarea 2-way se alcanzan resultados ligeramente superiores de 73.5% para el mejor y 50% como valor de exactitud más bajo. Mientras que para la tarea piloto los resultados son 45.59% para el mejor y 17.51% para la peor corrida.

[21] http://xwn.hlt.utdallas.edu

Un año después, en el 2010, se celebra el RTE6. Esta vez fue introducida una innovación importante, la tarea principal tradicional fue sustituida por una nueva tarea, aunque similar a la tarea piloto del RTE-5; la que Implicación Textual se realiza en un corpus real en escenarios de actualización de resúmenes. También fue propuesta una subtarea con el objetivo de detectar información novedosa. En esta versión se crea una tarea piloto de validación a la que llamaron KBP (Knowledge Base Population). Esta tarea se basó en la tarea del TAC de rellenado de ranuras en la población de bases de conocimiento (McNamee y Dang, 2009). Esta tarea se implementa para comprobar la importancia de la implicación en el KBP.

También en esta competición, se propone la tarea principal de llevar a cabo pruebas de ablación en los recursos de conocimiento que se utilizan. Esto se realiza con el objetivo de estudiar la pertinencia de dichos recursos en el reconocimiento de Implicación Textual. Este año las pruebas de ablación también se extendieron a las herramientas, tales como analizadores, solucionadores de correferencia, reconocedores de entidades nombradas (Bentivogli, Clark et al., 2010).

Los resultados para la tarea principal fueron bastante discretos, solo se alcanza un 48.01% de F-Medida para la mejor corrida y 11.60% la peor. Los resultados para la tarea Novelty Detection fueron 82.91% de F-Medida para la mejor corrida y 43.98% la peor. En la justificación de los resultados se alcanzan resultados mucho más bajos, 48.26% y 3.79% de F-Medida para la mejor y la peor corrida respectivamente.

Para la nueva edición en año 2011 -El RTE-7- replica el ejercicio propuesto en RTE-6; no se realiza ningún cambio en esta nueva edición. Treinta equipos participan en la tarea principal, enviado 33 corridas. Cinco equipos participan en la tarea Novelty Detection con 13 corridas. Finalmente la tarea del KBP fue cubierta solo por dos participantes, los que enviaron 5 corridas (Bentivogli, Clark et al., 2011). Los resultados continúan siendo bajos también en este año. Para la tarea principal el mejor resultado que se reporta es de un 48% de F-Medida, mientras el más bajo obtiene apenas un 14.72%. En cambio para la tarea Novelty Detection se obtienen buenos resultados, alcanzando la mejor corrida un 90.95% de F-Medida y la peor 81.82%. Para la tarea Justification se obtienen valores bastante bajos de F-Medida con un 38.22% y 26.56% para la mejor y peor corrida respectivamente. Para la tarea del KBP solo se logra como mejor valor de F-Medida un 19.02% (Bentivogli, Clark et al., 2011).

Mucho más reciente, en junio del 2013, se celebra en Japón la conferencia de reconocimiento de inferencias en texto (RITE-2) en el National Institute of Information Test Collection for Information Retrieval Systems**,** NTCIR-10 (Watanabe, Miyao et al., 2013). La primera edición para los sistemas que reconocen relaciones semánticas entre oraciones para japonés y chino, se organizó en NTCIR-9, donde se alcanza un discreto 58% de exactitud. Para la nueva edición, las tareas del RITE-2 consistieron en la clasificación binaria (CB) del la implicación, clasificación multiclases (CM) incluyendo paráfrasis y

contradicción, la subtareas de examen de ingreso, pruebas unitarias y la subtarea del RITE para Question Answering. Los resultados de esta segunda edición mejoran, con un 81.64% para el valor más alto de exactitud en la tarea CB y un 70% para CM.

Han existido otros marcos donde también se pone de manifiesto los intentos por reconocer la implicación textual, por ejemplo, las competiciones sobre validación de respuestas, en inglés Answer Validation Exercise (AVE). Se han llevado a cabo tres ediciones:

- El AVE-2006 (Peñas, Rodrigo et al., 2007) en el que se obtiene como mejor resultado de un 43.93% de F-Medida;
- El AVE-2007 (Peñas, Rodrigo et al., 2008) donde uno de los objetivo en esta competición fue llevar los sistemas basados en Implicación Textual al problema de la generación automática de hipótesis que, aunque no es en sí misma parte de la tarea de reconocimiento de Implicación Textual (RTE), si constituye una necesidad para el ajuste de los sistemas de validación respuesta;
- Y por último, el AVE-2008 (Rodrigo, Peñas et al., 2009). Al igual que en el año anterior la implicación textual no es una tarea individual, pero los participantes en esta competición, excepto dos equipos, reportaron utilizar RIT en sus sistemas.

Otra competición muy reciente, en la que se ha incluido la tarea del reconocimiento de similitud, es la denominada Semantic Textual Similarity (STS) dentro del marco de SemEval-2012; donde aparece también el reconocimiento multilingüe de la implicación textual (Cross-Lingual Textual Entailment task (CLTE)). En esta competición se presentan 35 equipos y 88 corridas (Agirre, Cer et al., 2012). Luego en el 2013, se vuelve a realizar este evento, con elementos nuevos como:

- La STS es seleccionada como la tarea oficial compartida del * SEM 2013;
- Se ofrecen recursos comunes para STS como: anotación compartida y productos para inferencia;
- Se ofrece una lista completa de las tareas de evaluación y bases de datos, software (incluyendo un baseline de código abierto como DKPro) y documentos relacionados con el STS[22].

Por la importancia que tiene para esta investigación y teniendo en cuenta que los métodos y sistemas creados en este trabajo se pone a prueba en esta competición, a continuación se describe en detalle la tarea del STS.

Según se describe en (Agirre, Cer et al., 2012), la tarea piloto número seis en Similitud Textual Semántica, de SemEval 2012 –también aplicable al 2013- proporcionan un conjunto de pares de frases

[22] http://www-nlp.stanford.edu/wiki/STS

obtenido a partir de un corpus segmentado. Para cada par de frases, s_1 y s_2, todos los participantes tienen que cuantificar la similitud de s_1 y s_2, proporcionando una puntuación de similitud.

La salida de los diferentes sistemas se compara con las puntuaciones manuales proporcionadas por el archivo estándar de oro de SemEval-2012 y SemEval-2013, las evaluaciones van de cero a cinco (0 a 5) de acuerdo con los siguientes criterios:

- (5) Las dos frases son equivalentes, pues significan lo mismo.
- (4) Las dos frases son en su mayoría equivalente, pero algunos detalles sin importancia difieren.
- (3) Las dos frases son más o menos equivalentes, pero alguna información importante difiere o falta.
- (2) Las dos frases no son equivalentes, pero comparten algunos detalles.
- (1) Las dos frases no son equivalentes, pero son sobre el mismo tema.
- (0) Las dos frases son sobre diferentes temas.

Las comparaciones entre los sistemas participantes fueron hechas a través del Coeficiente de correlación Pearson de entre las puntuaciones de los sistemas y las hechas por humanos. Se proponen también otras dos variaciones de esta medida, para más detalle puede consultarse (Agirre, Cer et al., 2012).

Otra tarea que se incluye en SemEval-2013 es la evaluación de frases semánticas (Evaluating Phrasal Semantics) (Korkontzelos, Zesch et al., 2013). Se presentan dos subtareas diseñados para evaluar los modelos:

a) La similitud semántica de las palabras y frases compuestas;

b) Evaluación de la composicionalidad de frases en contexto.

El objetivo de estas tareas es ofrecer una oportunidad para reunir los enfoques de los numerosos problemas relacionados bajo un conjunto común de evaluaciones. Se pretende que después de la competición, se ajuste la evaluación y los conjuntos de datos constituyan un punto de referencia para la evaluación de estos modelos.

Para la similitud semántica de las palabras y frases compuestas, la meta es evaluar cuán bien los sistemas pueden juzgar la similitud semántica de una palabra y una secuencia breve de (dos o más) palabras. Por ejemplo, en el par de secuencia de palabra: (contact, close interaction) el significado de la secuencia como un todo está semánticamente próximo a la acepción de la palabra. Contrariamente, en el par de secuencia de palabras: (megalomania, great madness) la acepción de la palabra es semánticamente diferente al significado de la secuencia, aunque no sea enteramente inconexo.

Esta subtarea, se ocupa de un problema medular debido a que las interpretaciones satisfactorias parecen ser fundamentales para los modelos composicionales de significado, además es controversialmente la base para las subtareas subsiguientes. En este contexto, el criterio de evaluación del desenvolvimiento de los sistemas participantes se mide en términos de precisión/ cobertura/ F-medida.

En otro orden de cosas y continuando con el tema de la similitud, se hace necesario abundar sobre una técnica que se ha comenzado a utilizar en algunos de los trabajos que tratan la similitud entre frases, el alineamiento. El siguiente epígrafe hace una breve reseña del estado de esta cuestión en la búsqueda de la similitud textual.

2.6.4 El alineamiento como una técnica para la determinación de similitud

Aunque la mayor cantidad de artículos publicados, que tratan el alineamiento de palabras han sido sobre el alineamiento de corpus paralelos en la temática de la traducción automática -en ocasiones- también el problema de la implicación ha sido enfocado desde la óptica del alineamiento. En la mayoría de los casos alineamiento léxico.

Aunque no igual que en la traducción automática, en las competiciones del RTE se ha abordado el problema de alineación en una variedad de formas, aunque a menudo sin distinguirlo como un sub-problema independiente. Algunos trabajos usan un enfoque basado en la medición del grado de solapamiento léxico entre bolsas de las palabras (Dagan, Glickman et al., 2005; Jijkoun y Rijke, 2005).

En trabajos, como los descritos en (Marsi y Krahmer, 2005; MacCartney, Grenager et al., 2006), se aboga por un componente de la alineación distinto, una estrategia crucial para los sistemas de alto rendimiento. Sin embargo, como se plantea en (MacCartney, Galley et al., 2008) cada uno de estos sistemas ha llevado a cabo la alineación de forma particular y muy poco documentada, a menudo usan datos propios, que hacen las comparaciones y el desarrollo más difícil.

De otro lado, en el área de la Inferencia del Lenguaje Natural (Natural Language Inference, NLI), la investigación sobre el alineamiento se ha visto obstaculizada por la escasez de datos disponibles desde los cuales aprender. Afortunadamente, esto cambia en el año 2007 con el lanzamiento, por Microsoft Research (MSR), de corpus generados por humanos con anotaciones de alineamiento, propuesto en (Brockett, 2007) para problemas de inferencia a partir del RTE en su segunda versión RTE-2 (Bar-Haim, Dagan et al., 2006). Éste es el primer trabajo en explotar estos datos para el entrenamiento y evaluación de los modelos de alineamiento.

También en este sentido, en (Glickman, Dagan et al., 2006) se describe un modelo de alineamiento para la implicación léxica, que utiliza estadísticas de co-ocurrencia en la Web, en una representación en bolsa de

palabras. Esta es una propuesta simple, pues no utiliza un análisis profundo. Archivan una precisión de 59% y una precisión promedio de 57%.

Un años después, en el 2007, Adams y otros investigadores reflejan en (Adams, Nicolae et al., 2007) que comparan dos métodos de alineamiento, uno léxico al que llaman Extended Lexical Overlap (ELO) y otro léxico-semántico al que titulan Lexico-Semantic Matching (LSM).

El primer método es basado solamente en la medición del solapamiento entre palabras para formar un mapeo entre los pares Texto-Hipótesis, del que extraen una serie de atributos para entrenar un clasificador basado en árboles de decisión (J48). El segundo, intenta el alineamiento de los chunks entre el Texto y la Hipótesis, representando a ambos como árboles de dependencia para su comparación. Utilizan también como atributos relaciones de WordNet (Miller, Beckwith et al., 1990). También realizan una combinación de los dos métodos. Contradictoriamente, el método ELO supera los resultados de los otros dos métodos obteniendo una exactitud general de 67%, al ser evaluado en los corpus del RTE1, RTE2 y RTE3. Al final, argumentan que el funcionamiento del método LSM se ve afectado por la precisión de los recursos que utiliza para resolver la correferencia y los parsers de dependencias y semántico.

Algo más alejado de este tema y a su vez una temática más reciente, aplicada también dentro del marco del PLN, es la determinación de la polaridad sentimental, elemento clave dentro de área del Opinion Mining (Minería de Opiniones). Esta técnica se basa en poder determinar los sentimientos expresados en un texto. La clasificación -hasta el momento- se ha enmarcado en cuantificar el grado de positividad, negatividad, objetividad o neutralidad que posee una determinada frase. A continuación se comenta sobre los trabajos que tienen relación con esta temática.

2.6.5 Trabajos relacionados con la polaridad sentimental en el RIT

Aunque no se puede plantear que existe una amplia gama de trabajos en los que se haya estudiado la relación de la polaridad sentimental y en el RIT, se comentará sobre aquellos que más cerca están y por tanto guardan cierta relación con el tema.

Esta investigación, está parcialmente relacionada con el trabajo descrito en (Nairn, Condoravdi et al., 2006), en el que los autores presentan una estrategia para detectar el compromiso del autor con la verdad/falsedad de las cláusulas de complemento, en función de su tipo sintáctico y en el significado de su predicado incorporado. Además, demuestran que las implicaciones de un predicado en una profundidad arbitraria de inclusión con relación a su cláusula de complemento, depende de una noción global determinada de la polaridad relativa. Por último, estos autores demuestran que las diferentes clases de verbos que toman el complemento, tienen un efecto diferente en la polaridad de sus cláusulas de complemento y que este efecto depende de forma recursiva de su propia inclusión.

Otras investigaciones como (Balahur y Montoyo, 2010), han llevado a cabo el análisis de una metodología para detectar frases relacionadas con diferentes características y el empleo de RTE para la minería opinión. Estos autores ponen a prueba la relación de implicación en una ventana de tres frases consecutivas. En este trabajo se puede ver, analizando los resultados obtenidos, que la implicación textual puede ser útil en el momento de realizar minera opinión basada categoría. Sin embargo, tal y como ellos mencionan, aún queda mucho por hacer en el ámbito de la similitud semántica entre los textos de opinión.

Otras trabajos como (Schmerling, 1971; Progovac, 1993; Giannakidou, 2002) también estudian el tema de la polaridad en el RIT, pero enfocándose más en la polaridad asociada a la existencia de la negación del verbo en las frases analizadas.

No hay evidencias, al menos encontradas por este investigador, de que haya existido una investigación previa de la influencia de la polaridad sentimental en el RIT. Solo se han visto algunos trabajos que tocan el tema desde otras perspectivas. Por lo que se llega a la conclusión de que no se han hecho de la misma manera, ni con el mismo propósito.

Pasando a nuevo tema, pero aún enmarcado en la similitud, se puede ver que existe otra forma de comprobar la similitud entre frases. Este es el caso de la búsqueda de la paráfrasis, en el siguiente epígrafe se argumenta sobre este concepto.

2.6.6 La paráfrasis

En el diccionario de la Real Academia Española de la lengua (Academia Española, 2001), la paráfrasis se define de tres formas:

1. *"Explicación o interpretación amplificativa de un texto para ilustrarlo o hacerlo más claro o inteligible"*.
2. *"Traducción en verso en la cual se imita el original, sin verterlo con escrupulosa exactitud"*.
3. *"Frase que, imitando en su estructura otra conocida, se formula con palabras diferentes"*.

Un sistema de detección de paráfrasis intenta obtener pares (o conjuntos) de expresiones que constituyan paráfrasis entre sí. Según Regina Barzilay en su trabajo publicado en (Barzilay y McKeown, 2001), plantea que es un conjunto de expresiones intercambiables que son ligeramente diferentes en su matiz, verbosidad, etc.

Para Dekang Lin y Patrick Pantel constituye un conjunto de patrones para capturar varios tipos de información a partir de documentos (Lin y Pantel, 2002). Sin embargo, en (Yamamoto, 2002) se

considera que es el grado de replicabilidad de dos expresiones E_1 y E_2, las que son diferentes entre sí en algunos sentidos.

En otro extremo, Yusuke Shinyama, Satoshi Sekine y Kiyoshi Sudo, exponen una idea mucho más escueta en (Shinyama, Sekine et al., 2002). Para ellos las paráfrasis son simplemente expresiones similares, pero el primer autor, Yusuke Shinyama en un trabajo posterior (Shinyama, 2005), las considera ocurrencias de pares de caminos extraídos, mientras que Takaaki Hasegawa, Satoshi Sekine y Ralph Grishman piensan y exponen en (Hasegawa, 2005), que son un conjunto de frases las cuales expresan la misma cosa o evento.

Otras definiciones son la de Yves Lepage y Etienne Denoual, presentadas en (Lepage, 2005): oraciones diferentes que tienen la misma traducción. A diferencia de Chris Callison, Philipp Koehn y Miles Osborne que la definen como: formas alternativas de expresar la misma información dentro de un lenguaje (Callison-Burch, Koehn et al., 2006).

Después del estudio de las diferentes definiciones, este autor no concuerda totalmente con estas formulaciones. Por tanto, se define el `concepto de paráfrasis` como: frases que pueden o no tener pequeñas variaciones en su estructura interna y también ser formuladas con palabras diferentes, pero siempre manteniendo el mismo sentido semántico.

2.6.6.1 Clasificación de la paráfrasis

Las paráfrasis pueden clasificarse para su estudio en:

- **Paráfrasis mecánica**: Consiste en sustituir alguna palabra por sinónimos o frases alternas con cambios sintácticos mínimos.

  Sintáctica: Está dada básicamente por la relación existente entre lo activo, lo pasivo y lo relativo. Ejemplo:

  `El gato arañó a la niña.` I

  `A la niña la arañó el gato.` II

  `La niña fue arañada por el gato.` III

  Semántica: Está dada por la utilización de sinónimos para expresar el mismo significado de palabra, sin alterar la estructura de la oración. Ejemplo: para la oración I, `El gato araño a la niña`; utilizando las combinaciones, sólo de los sinónimos, que a continuación se presentan se pueden obtener más de ciento veinte variantes de la oración.

El	gato	arañó	a la	niña.
	felino	rasguñó		nena.
	miau	rasgó		infanta.
	minino	raspó		pequeña.
	gatico	desgarró		chica.
	bigotudo	hirió		chiquilla.

- **Paráfrasis constructiva:** Esta otra en cambio, reelabora el enunciado dando origen a otro con características muy distintas, conservando el mismo significado. Ejemplos:

Sólo el 30% de los estudiantes aprobó el examen.	I
La mayoría de los estudiantes desaprobaron el examen.	II
Emma lloró.	I
Emma estalló en lágrimas.	II

La detección de paráfrasis se encarga precisamente de determinar estos pares o conjuntos de frases con el mismo significado de forma automática.

2.6.6.2 Aplicaciones de la paráfrasis

La gran importancia que se le confiere hoy en día a la detección de paráfrasis está dada precisamente por la magnitud de sus aplicaciones y el peso que estas tienen para el PLN.

- Extracción de Información: conocer si una información parafrasea a otra que ya fue almacenada, sería un elemento importante para evitar información redundante en las bases de datos. También, si se desea extraer una información, que lógicamente está escrita de alguna forma específica, pudiera ser importante buscarla por sus formas alternativas.
- Generación de Resúmenes: Comparar las frases de aquel o aquellos artículos a resumir, para garantizar que no falte alguna idea importante o se repita una redundante, garantiza la calidad y el tamaño óptimo del resumen.
- Traducción Automática: la paráfrasis multilingüe. Conocer que dos frases dichas o escritas en lenguajes diferentes, con palabras y estilos propios de cada autor, conservan el mismo significado sería de gran importancia.
- Recuperación de Información: Difícil resulta en muchas ocasiones conocer el término exacto que utilizó un autor para referirse a determinado concepto. Incluso, dependiendo de la región geográfica, las cosas pueden tomar nombres diferente, incluso dentro un mismo documento se utilizan sinónimos para evitar la redundancia. Un sistema que pueda lidiar con la paráfrasis,

permitiría una mejor recuperación de los documentos con la información buscada.

- Búsqueda de Respuesta: Al igual que en la RI, si la pregunta que se hace pudiera contestarse, a pesar de contener palabras y estructuras diferentes, mejoraría la precisión de estos sistema. También, si se puede comparar un significado parecido o igual al de alguna pregunta anterior, no hay necesidad de volver a buscar -la respuesta ya se sabe- es la misma.
- Detección de plagios: Con un sistema de detección de paráfrasis se podría facilitar la ardua labor de desenmascarar el clásico cortar y pegar información ajena, lo que contribuiría al cuidado de la propiedad intelectual.

2.6.6.3 Trabajos relacionados

La paráfrasis forma parte de un campo que ha cobrado singular importancia en los últimos años. Entre los estudiosos del tema se encuentran algunos que sobresalen por sus resultados y cuyos trabajos serán referidos a continuación.

Entre los investigadores más destacados en esta temática están Regina Barzilay y Kathleen R. McKeown, ellos presentan en (Barzilay y McKeown, 2001) un algoritmo de aprendizaje no supervisado para la identificación de paráfrasis en una colección de traducciones al inglés de una misma fuente. Atendiendo al principio de que cada autor, por muy creativo que sea, para contar una misma historia -en el mismo idioma- debe mantener un conjunto de palabras constantes, es decir, buscar pares de expresiones A C B y A D B con los mismos argumentos (A, B), o con aquellos que brinden la misma información; entonces C y D tienen una buena probabilidad de formar parte de una paráfrasis.

Luego, se establece el contexto que define esa paráfrasis (A, B) y se buscan otras oraciones que se emparejen con este patrón. De esta manera analizan la paráfrasis léxica y la sintáctica.

Otros como Shinyama, Sekine y Kiyoshi en (Shinyama, Sekine et al., 2002) parten de la misma idea presentada en (Barzilay y McKeown, 2001), pero utilizan entidades, es decir, obtienen paráfrasis a partir de artículos de noticias que describen el mismo evento, asumiendo que para contar un mismo hecho, el mismo día, es necesario mantener constante un conjunto de entidades (nombres de organizaciones o personas, fechas, localizaciones, expresiones numéricas, etc.), las que son preservadas a través de la paráfrasis.

Primeramente, utilizan un sistema de RI para obtener pares de artículos de una clase dada de eventos: asesinatos o asuntos personales. A estos le aplican un conjunto de algoritmos para detectar las entidades y comparan todas las oraciones de cada artículo por separado, buscando pares de oraciones que compartan estas entidades. Si el número de entidades compartidas entre dos oraciones excede un cierto umbral, se

consideran paráfrasis automáticamente, de lo contrario se extraen las porciones adecuadas de las oraciones y se forman los árboles de dependencia, que son los que se analizan para determinar la paráfrasis y después permiten reconstruir la frase original.

De esta manera, estos autores obtienen un patrón y buscan con quien emparejarlo en un documento paralelo -claro está- en este caso se pasa a considerar el dominio de las expresiones, para no introducir mucho ruido.

En la Figura 2.2, se puede observar el gráfico de la estructura general de este método, tomado de la página 2 del artículo: Automatic Paraphrase Acquisition from News Articles (Shinyama, Sekine et al., 2002).

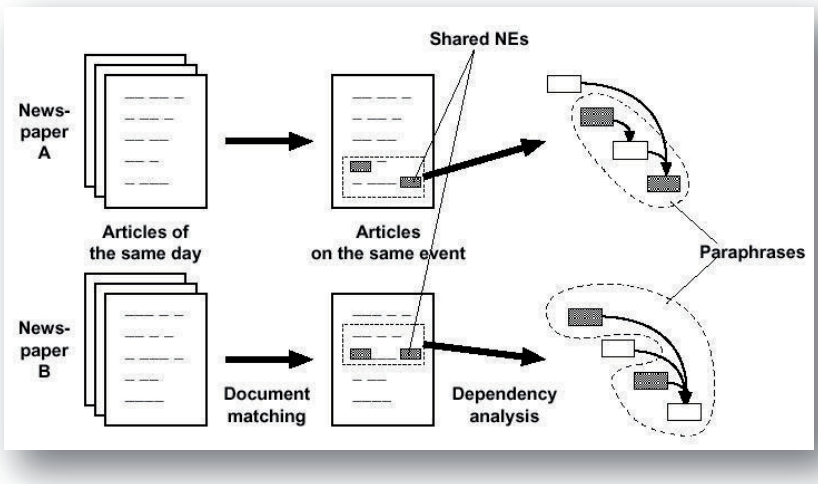


Figura 2.2. Método de adquisición de paráfrasis de (Shinyama, Sekine et al., 2002)

Por otra parte, Kazuhide en (Yamamoto, 2002) propone un método de adquisición automática de paráfrasis basado en el conocimiento del contexto de las palabras. Utiliza como entrada artículos de periódicos, los cuales analiza morfológicamente y parsea[23] para obtener relaciones triples: (c_1, r, c_2) donde la palabra c_1 depende de c_2 por la relación r. c_1, c_2 y r son verbos y sustantivos y definen un contextos, pero contextos iguales no garantizan similitud semántica, por lo que utilizan diccionarios de pares de antónimos para una mejor precisión, puesto que considera que los sinónimos y los hiperónimos introducen mucho ruido. Luego convierte cada tripleta en un dígrafo: (c_1, r $\rightarrow c_2$) y (c_2, r $\leftarrow c_1$) que le permite medir el grado de similitud de dos contextos en término de sus relaciones, atendiendo al principio de que mientras menor sea la frecuencia de una relación en el corpus, mayor será la probabilidad de que sean paráfrasis.

[23] Anglicismo que indica acción que ejecuta un parser o analizador sintáctico, que concluye con la construcción de un árbol sintáctico.

Otros como Ibrahim, Katz y Lin, continúan en (Ibrahim, 2003) la idea de Barzilay y McKeown en (Barzilay y McKeown, 2001), pero a diferencia de esta, que utiliza patrones rígidos con palabras contiguas a lo sumo separadas por otra palabra, aplica la idea de Dekang Lin y Patrick Pantel de implementar un árbol de dependencia, para así hacer más flexible el patrón (Lin y Pantel, 2002); considerando que siempre que exista un camino de dependencia entre las palabras que forman el patrón, hay paráfrasis.

Unos años después, Thierry Poibeau en el trabajo (Poibeau, 2004) describe un método supervisado, a quien le tiene que proporcionar un ejemplo inicial. Los resultados dependen en gran medida del ejemplo dado, pues de éste obtiene el patrón que busca en el corpus para determinar paráfrasis por similitud semántica.

La comparación comienza por el sustantivo cabecera y si éste sobrepasa cierto umbral continúan analizando el verbo y los complementos. Finalmente obtiene una tabla con las estructuras predicativas que son semánticamente equivalentes al patrón del ejemplo inicial.

El proceso usa los corpus y la red semántica como dos fuentes de conocimiento complementarias diferentes. La primera le brinda expresiones posibles y filtra las irrelevantes, mientras que la segunda le provee información acerca de las semánticas léxicas y las relaciones entre palabras.

En la Figura 2.3, se puede observar el gráfico de la estructura general del método, tomado de la página 2 del artículo: Automatic extraction of paraphrastic phrases from medium size corpora (Poibeau, 2004).

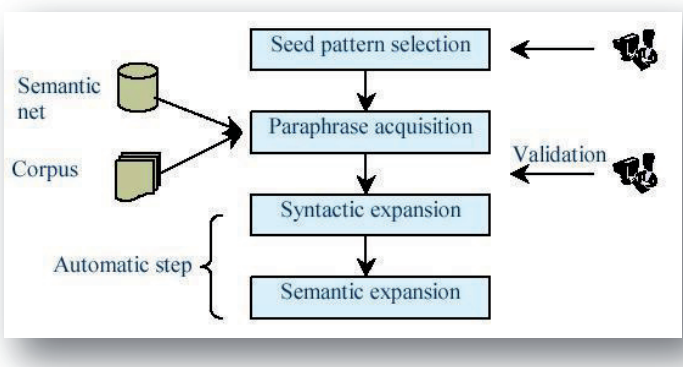


Figura 2.3. Método de adquisición de paráfrasis de (Poibeau, 2004)

Nuevamente, Barzilay, esta vez en compañía de Lillian Lee, presenta una investigación en (Barzilay y Lee, 2005) para trabajar con la paráfrasis. En este caso parte de su idea anterior, pero la utiliza para generar paráfrasis, por lo que su mayor aplicación está dirigida a los sistemas de generación de texto. Una vez analizados los corpus comparables y extraídos los patrones forma grupos con estos, así, llegada una

oración al sistema, se analiza a qué grupo pertenece y se originan un grupo considerable de paráfrasis, es decir, de opciones o formas de expresar la misma idea.

En la Figura 2.4, se puede observar el gráfico de la estructura general del método, tomado de la página 3 del artículo: Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment (Barzilay y Lee, 2005).

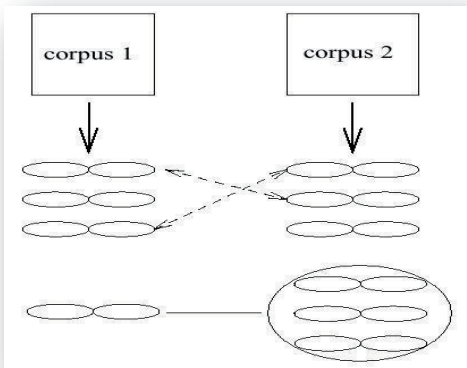


Figura 2.4. Método de adquisición y generación de paráfrasis de (Barzilay y Lee, 2005)

Autores como Takaaki Hasegawa, Satoshi Sekine, y Ralph Grishman en (Hasegawa, Sekine et al., 2005) implementan un método no supervisado para descubrir paráfrasis que contengan dos entidades desde un gran corpus no etiquetado. Al igual que en (Shinyama, Sekine et al., 2002), utilizan un etiquetador de entidades para determinarlas, pero escogen las dos entidades cuando estas se encuentran relacionadas por menos de cinco palabras y aparecen más de 30 veces en el corpus. Luego aplican la medida del coseno al vector resultante del contexto y forman grupos jerárquicos con los pares de entidades respetando el nivel de similitud, para poder producir paráfrasis. En la Figura 2.5 se puede observar el gráfico de la estructura general del método, tomado de la página 3 del artículo: Unsupervised Paraphrase Acquisition via Relation Discovery (Hasegawa, Sekine et al., 2005).

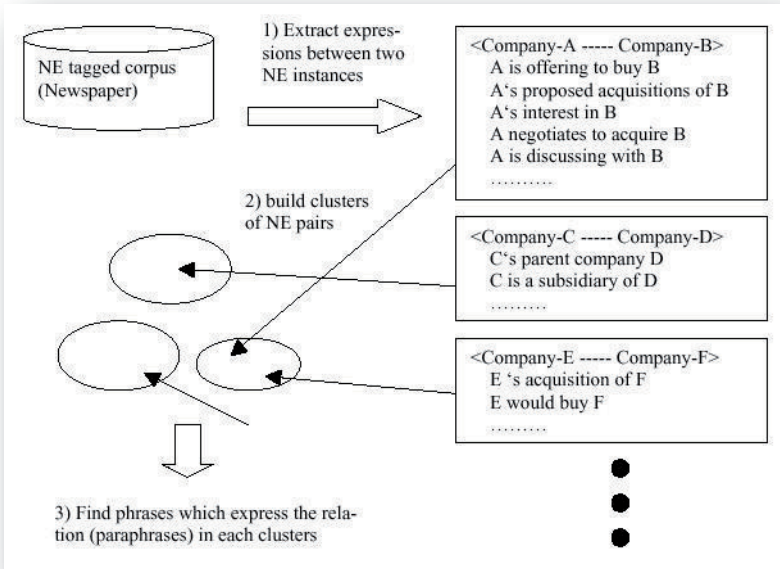


Figura 2.5. Método de adquisición de paráfrasis de (Hasegawa, Sekine et al., 2005)

También en el 2005, Weigang junto a otros investigadores en (Li, Liu et al., 2005), describen una nueva representación de paráfrasis en plantillas y un método de generalización. Para ello reciben como entrada un conjunto de paráfrasis que son analizadas con un diccionario semántico que desambigua el sentido de las palabras y define los espacios en blanco con un código, es decir, los lugares donde podrá ir otra palabra que mantenga cierta similitud con la que había según ese código. Como es obvia la limitación que introduce un código único, aplican una ingeniería de búsqueda (un parser de dependencias) en cada ejemplo para extender los grupos de palabras de los espacios y generalizar los ejemplos.

En otra investigación, Yusuke Shinyama y Satoshi Sekine en (Shinyama y Sekine, 2005), se centran en expresiones que podrían producir la misma información, aplicándolas especialmente a la extracción de información. Estas expresiones las adquieren a partir de corpus comparables (artículos del mismo día o el mismo tema), cuyas palabras con valor semántico son utilizadas para calcular la medida del coseno a la primera oración de cada artículo; de esta forma pueden conocer si su grado de similitud es mayor a cierto umbral.

Luego, una vez escogido el par de corpus de fuentes diferentes con que va a trabajar, seleccionan la primera oración de cada corpus, por ser usualmente la más importante y las parsean. Luego utilizan un solucionador de correferencias que recibe como entrada un par de oraciones parseadas y prueba todas las

combinaciones posibles entre las dos frases hasta encontrar las anclas (entidades que son compartidas por ambas oraciones).

De esta forma, cuando se han encontrado los puntos de contactos, comienzan a generar paráfrasis a través del recorrido por los caminos de los árboles. En la Figura 2.6 se puede observar el gráfico de la estructura general del método, tomado de la página 4 del artículo: Using Repeated Patterns across Comparable Articles for Paraphrase Acquisition (Shinyama y Sekine, 2005).

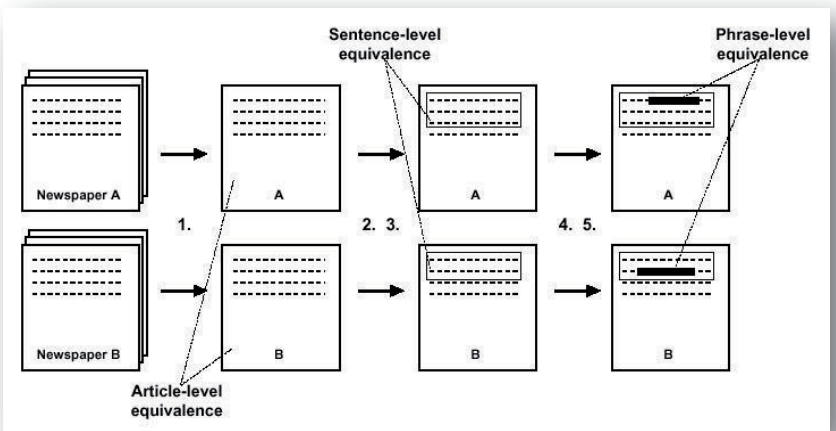


Figura 2.6. Método de adquisición de paráfrasis de (Shinyama y Sekine, 2005)

Contrario a los anteriores, Jesús Herrera de la Cruz implementa en (Herrera de la Cruz, 2005) un sistema para una de las tareas de El PASCAL RTE[24] Challenge: la implicación textual, específicamente la detección automática de implicación semántica entre parejas de textos en lenguaje natural (monolingüe inglés), en el que también incluye adquisición de Paráfrasis.

El sistema extrae del corpus de entrada los textos e hipótesis y alimenta el analizador de dependencias Minipar[25] para que este genere los árboles. Luego ejecuta un módulo de implicación léxica sobre los nodos y obtiene como salida una lista de pares <T, H>, donde la unidad léxica de T implica a la unidad léxica de H. Esta implicación a nivel léxico se determina teniendo en cuenta relaciones de WordNet (Miller, Beckwith et al., 1990) de sinonimia e hiperonimia. La negación no genera implicación, pero se analiza utilizando la relación de antonimia. WordNet, unido a un reconocimiento difuso de la coincidencia entre candidatos a multipalabras, mediante la distancia de edición de Levenshtein, permiten

[24] Textual Entailment Recognition, el reconocimiento de inferencias textuales fue propuesto como una tarea genérica que intenta capturar las mayores referencias semánticas necesitadas a través de las diferentes aplicaciones del PLN.

[25] Analizador léxico sintáctico desarrollado por Dekang Lin.

el análisis de las multipalabras. Finalmente interviene un módulo de evaluación de solapamiento, que busca ramas en el árbol de dependencias de las hipótesis, conformadas por nodos lexicalmente implicados por nodos del texto. Cuando el árbol de dependencias de una hipótesis muestra un porcentaje de solapamiento de nodos mayor o igual al 50%, considera que hay similitud entre el par texto-hipótesis. En la Figura 2.7 se puede observar el gráfico de la estructura general del método, tomado de la página 36 de la suficiencia investigativa titulada: Un Modelo Fundamentado en Análisis de Dependencias y WordNet para el Reconocimiento de Implicación Textual (Herrera de la Cruz, 2005).

Figura 2.7. Método de adquisición de paráfrasis de (Herrera de la Cruz, 2005)

A diferencia de los anteriores, Yves Lepage y Etienne Denoual en (Lepage y Denoual, 2005) enfocan su método hacia la traducción automática, por lo que utilizan un corpus multilingüe donde la detección de paráfrasis se reduce a la búsqueda de oraciones que compartan la misma traducción. Luego generan paráfrasis basados en el hecho de que cualquier oración puede compartir permutaciones con otras oraciones del corpus. Finalmente, aplica un algoritmo de filtrado a las paráfrasis candidatas para eliminar las que no tengan una semántica lógica.

Yitao Zhang y Jon Patrick, continúan extendiendo su idea presentada en años anteriores y en (Zhang y Patrick, 2005) ofrecen un método para la detección de pares de paráfrasis que, en alguna medida, transforma el texto para que sea más genérico y simple que el original. Por ejemplo, pasan la voz activa a voz pasiva, que en el idioma inglés resulta mucho más común. La idea principal es que si dos oraciones son paráfrasis entre sí y se transforman, tienen una mayor oportunidad de ser transformadas en oraciones similares que si no son paráfrasis. Los pasos son:

1. Reemplazar todas las entidades por etiquetas genéricas.
2. Consulta el parser Minipar para obtener la estructura del árbol de dependencias del texto, donde buscar los verbos en voz activa y delimitar sujeto y el predicado. Luego reestructurar el árbol de dependencia colocando el verbo en voz pasiva y recorrerlo para obtener la oración con la voz del hablante cambiada.
3. Compilar una lista de palabras comunes y estructuras de frases que suelen introducir futuro, para

cambiarlas por una sola palabra, por ejemplo: plan to, be expected to, se sustituye por will.

Utilizan, en el estado de aprendizaje supervisado, el módulo de árbol de decisión de Weka[26] y aplican las siguientes características:

- La subcadena común de máxima longitud: Medida de longitud del mayor string común compartido por dos oraciones. Es una secuencia consecutiva de palabras.
- La subsecuencia común de máxima longitud: Medida de la mayor secuencia común de cadenas compartida por dos oraciones. Esta secuencia no requiere ser consecutiva en el texto original.
- La distancia de edición: Describe cuántas operaciones de edición (adición, eliminación o reemplazo) son requeridas para convertir un texto fuente en un texto destino.
- La precisión modificada N-Gram[27]: Es también una métrica importante adoptada a partir del algoritmo BLEU para la evaluación de las traducciones automáticas.

Un año después, Chris Callison-Burch, Philipp Koehn y Miles Osborne en (Callison-Burch, Koehn et al., 2006) también utilizan la paráfrasis para la traducción automática. La frase original que se desea parafrasear la traducen a un idioma conocido (inglés) y luego extraen palabras de esta para volverlas a traducir al lenguaje fuente. Estas nuevas palabras son sustituidas en la oración original.

En otro contexto, David Kauchak y Regina Barzilay en (Kauchak y Barzilay, 2006) exploran el uso de los métodos de paráfrasis en las técnicas de refinamiento de evaluación automática. Dada una oración de referencia y una oración generada por la máquina (paráfrasis) devuelven otra que esté más cercana a la oración de referencia que la anterior. Para ello tratan de sustituir la mayor cantidad posible de palabras de la oración generada en la oración de referencia, sin que esta última pierda su sentido, buscando si existe relación de sinonimia entre las palabras que no se repiten de ambas oraciones. Como no en todos los contextos los sinónimos son intercambiables, utilizan un clasificador de dependencias contextuales, el cual entrena con un gran corpus de oraciones que contienen la palabra usada en ese sentido e igual cantidad de oraciones para el caso contrario.

Ya en el año 2007, conducido por el investigador Samuel Fernando en (Fernando, 2007), se hace un estudio de trabajos recientes en el área de la identificación de paráfrasis y a continuación se mencionan algunos:

- matrixLin: implementa la métrica Lin de WordNet Similarity (Lin, 1998) (ver ecuación 2.18).

[26] http://www.cs.waikato.ac.nz/~ml/weka/

[27] N-gramas o N palabras.

- Mihalcea06: presenta un método de búsqueda de similitud semántica de segmentos de textos cortos Mihalcea et al. (2006). La motivación detrás de este método es que mientras más sofisticadas sean las medidas de similitud entre palabras, más exactos serán los valores de similitud entre oraciones. Por lo que se prueba un conjunto de métodos para mejorar la medida de similitud palabra-palabra, resultando ser los mejores las seis métricas implementadas en WordNet Similarity (lesk, lch, wup, resnik, lin, jcn).
- Qiu06: identifica los trozos de información común y los aparea (Qiu, Kan et al., 2006). Luego utiliza un parser sintáctico que le permite extraer tuplas de información y usarlas como argumento del predicado. Estas se comparan directamente y se les da un factor de peso, lo que se considera el indicador de equivalencia más importante. Las tuplas con mayor similitud son apareadas hasta un cierto umbral y las que queden sin aparear son analizadas con un método heurístico basado, principalmente, en los sustantivos.
- Zhang05: una vez delimitado el contenido que presenta probabilidad de similitud, se realizan transformaciones a las oraciones y se van comparando lexicalmente hasta obtener una equivalencia (Zhang y Patrick, 2005). Utiliza tres transformaciones: reemplazo, activo-pasivo, futuro.

Haciendo un aparte en este punto, se puede plantear que para la identificación de patrones y la escritura de reglas en los sistemas que en alguna medida buscan la similitud entre los textos, se ha hecho un amplio uso de las expresiones regulares. Por esta razón el siguiente epígrafe dedica un espacio a tratar esta temática.

2.7 Generación de expresiones regulares

Para hablar de las expresiones regulares, primero es necesario remontarse a lo que en lingüística se ha definido como la Jerarquía de Chomsky (JC) (Chomsky, 1956; 1957). Esta no es más que la clasificación jerárquica -que Chomsky ofrece- de distintos tipos de gramáticas formales, las que a su vez generan lenguajes formales: tipo 0 (gramáticas sin restricciones), incluye a todas las gramáticas formales; tipo 1 (gramáticas sensibles al contexto), estas gramáticas generan los lenguajes sensibles al contexto; tipo 2 (gramáticas libres del contexto), estas generan los lenguajes independientes del contexto, tipo 3 (gramáticas regulares) estas gramáticas generan los lenguajes regulares.

Por su uso en diferentes esferas del PLN, este trabajo centra más su atención en estas gramáticas tipo 3. Los lenguajes que utilizan estas gramáticas son aquellos que pueden ser aceptados por un AF. Además son los lenguajes que pueden ser obtenidos por medio de expresiones regulares.

Esta gramática de tipo 3 o gramáticas regulares, también conocida como de estados finitos, se ha aplicado en la lingüística computacional, por ejemplo, en el reconocimiento léxico, así como en la morfología.

Según se plantea en (Contreras, 2001) referenciado a (Roche y Schabes, 1997): "*...Debido a que las reglas de flexión forman un conjunto casi cerrado y mucho más pequeño que las reglas de sintaxis en cualquier lengua. En definitiva se ha usado en muchas tareas finitas del lenguaje, proporcionando un método muy eficiente para el computador*".

Debido a que en este trabajo se propone una forma de facilitar la generación de las Expresiones Regulares, se hace necesario comentar sobre sus principales meta-caracteres y su función; así como explicar la forma en que los interpretan los motores de ER. Entre estos caracteres se encuentran:

El Punto (`.`) –El punto es interpretado por el motor de búsqueda como cualquier otro carácter, excepto los caracteres que representan un salto de línea, a menos que se le especifique esto al motor de Expresiones Regulares. Por lo tanto si esta opción se deshabilita en el motor de búsqueda que se utilice, el punto le dirá al motor que encuentre cualquier carácter incluyendo los saltos de línea. Si se le dice que busque (`g.t`) en la cadena tomada de[28]: `el gato de piedra en la gótica puerta de getisboro goot`, el motor de búsqueda encuentra: `gat`, `gót` y por último `get`. Se puede apreciar que el motor de búsqueda no va a encontrar la palabra `goot`; esto sucede debido a que el punto (.) representa solo y únicamente un carácter.

La barra inversa o contra barra (`\`) –Se utiliza para marcar (en muchos casos conocido como escapar) el siguiente carácter de la expresión de búsqueda, de forma que este adquiera un significado especial o deje de tenerlo. O sea, la barra inversa no se utiliza nunca por sí sola, sino en combinación con otros caracteres. Al utilizarla, por ejemplo, en combinación con el punto (`\.`), este deja de tener su significado normal y se comporta como un carácter literal. De la misma forma, cuando se coloca la barra inversa seguida de cualquiera de los caracteres especiales que se discutirán a continuación, estos dejan de tener su significado especial y se convierten en caracteres de búsqueda literal:

`\t` — Representa un tabulador.

`\n` — Representa la nueva línea, el carácter por medio del cual una línea da inicio.

`\d` — Representa un dígito del cero al nueve (0 al 9).

`\w` — Representa cualquier carácter alfanumérico.

`\s` — Representa un espacio en blanco.

[28] http://www.buenastareas.com/ensayos/Sin-Escolaridad/7825797.html

Los corchetes (`[]`) –La función de los corchetes en el lenguaje de las expresiones regulares es representar clases de caracteres, o sea, reunir caracteres en grupos o clases. Son útiles cuando es necesario buscar uno de un grupo de caracteres. Dentro de los corchetes es posible utilizar el guión (`-`) para especificar rangos de caracteres, por ejemplo: `[a-z]`, esto quiere decir que se desea aceptar los caracteres en minúscula desde a letra `a` hasta la `z`.

La barra (`|`) –Sirve para indicar una de varias opciones. Por ejemplo, la expresión regular (`a|e`) encontrará cualquier (`a`) o (`e`) dentro del texto. La expresión regular `este|oeste|norte|sur` permitirá encontrar cualquiera de los nombres de los puntos cardinales. La barra se utiliza comúnmente en conjunto con otros caracteres especiales.

El acento circunflejo (`^`) –Este carácter tiene una doble funcionalidad, que difiere cuando se utiliza individualmente y cuando se utiliza en conjunto con otros caracteres especiales. En primer lugar su funcionalidad como carácter individual, representa el inicio de la cadena (de la misma forma que el signo de dólar (`$`) representa el final de la cadena). Por tanto, si se utiliza la expresión regular `^[a-z]` el motor encontrará todos los párrafos que den inicio con una letra minúscula. Cuando se utiliza en conjunto con los corchetes de la siguiente forma `[^\w]` permite encontrar cualquier carácter que NO se encuentre dentro del grupo indicado. La expresión indicada permite encontrar, por ejemplo, cualquier carácter que no sea alfanumérico o un espacio, es decir, busca todos los símbolos de puntuación y demás caracteres especiales.

El signo de interrogación (?) –Indica que puede existir cero o una ocurrencia de lo que precede al símbolo, por ejemplo para encontrar cero o una ocurrencia de `www`, se utiliza el patrón (`www\.`)?

El asterisco (`*`) –El asterisco sirve para encontrar algo que se encuentra repetido cero o más veces. Por ejemplo, utilizando la expresión `[a-zA-Z]\d`*, será posible encontrar tanto `H` como `H1`, `H01`, `H100` y `H1000`, es decir, una letra seguida de un número indefinido de dígitos.

Es necesario tener cuidado con el comportamiento del asterisco, ya que por defecto trata de encontrar la mayor cantidad posible de caracteres que correspondan con el patrón que se busca. De esta forma, si se utiliza `\(.*\)` para encontrar cualquier cadena que se encuentre entre paréntesis y se aplica sobre el texto: `ver (Fig.1) y (Fig.2)`, se espera que el motor de búsqueda encuentre solo los textos entre paréntesis `(Fig.1)` y por otra parte `(Fig.2)`, sin embargo, debido a las características que tiene la utilización del símbolo (*), en su lugar encontrará el texto completo `(Fig.1) y (Fig.2)`, incluyendo la conjunción (`y`). Esto sucede porque el asterisco le dice al motor de búsqueda que llene todos los espacios posibles entre dos paréntesis. Para obtener el resultado deseado se debe utilizar el asterisco en

conjunto con el signo de interrogación de la siguiente forma: `\(.*?\)`. Esto es equivalente a decirle al motor de búsqueda que encuentre un paréntesis de apertura y luego encuentre cualquier carácter repetido hasta que encuentre un paréntesis de cierre.

El signo de suma (+) −Se utiliza para encontrar una cadena que se encuentre repetida una o más veces. A diferencia del asterisco, la expresión `[a-zA-Z]\d+` encontrará `H1`, pero no encontrará `H`. También es posible utilizar este meta-carácter en conjunto con el signo de interrogación para limitar hasta dónde se efectúa la repetición.

El símbolo `{x}` −Indica x ocurrencias del carácter que lo precede, por ejemplo (`www.`), podría ser representado con el patrón (`w{3}\.`).

Existen algunos meta-caracteres que tienen a confundir porque significan lo contrario que sus homónimos en minúscula:

El símbolo `\S` −Indica que se selecciona cualquier carácter excepto un espacio en blanco.

El símbolo `\D` −Indica que se selecciona cualquier carácter excepto un dígito numérico.

El símbolo `\W` −Indica que se selecciona cualquier carácter no alfanumérico.

Esto son solo un pequeño grupo de los meta-caracteres que más se utilizan en la generación de ER, de esta pequeña muestra se podrá apreciar la complejidad de este lenguaje tan potente, pero a su vez tan críptico y difícil de interpretar a primera vista.

Se comentó en el epígrafe 1.2, que muchas de las aplicaciones para el PLN se basan en el principio de la aplicación de reglas. Estas reglas pueden estar definidas de muchas formas; una forma muy eficiente de hacerlo es utilizar las expresiones regulares para este fin.

Igualmente, de mucha utilidad sería tener dichas reglas fuera de las aplicaciones, de esta forma cualquiera podría modificarlas (incrementando o editando reglas) sin necesidad de modificar la aplicación.

A pesar de todas las ventajas que se han visto, obtener y probar ERs es una tarea difícil, muchos investigadores (Lesk y Schmidt, 1975; Appel, Mattson et al., 1989; García Vasconcelos, Fernández Orquín et al., 2005; Lin, Huang et al., 2006; Hurtado Carmona y Sarabia Agámez, 2008; Pérez Martínez, Fernández Orquín et al., 2011) se han planteado resolverla mediante herramientas que faciliten el trabajo en este sentido. Algunos, sólo se limitan a exponer una interfaz gráfica donde se introduce la expresión y un texto de prueba. Muchos otros ofrecen recursos para facilitar la construcción, como generación a partir de asistentes, resaltado con colores de las coincidencias, etc. Otros van más allá y permiten almacenar las expresiones regulares en librerías para que estén disponibles cuando se les necesite.

El hecho de intentar proveer a los investigadores de herramientas para la generación de ER, radica en la complejidad intrínseca de este proceso y en la búsqueda de la forma idónea de acercarlo a un procedimiento de construcción menos complejo.

Teniendo en cuenta lo antes mencionado, es de gran utilidad viabilizar la forma de construir expresiones regulares, abstrayendo a los usuarios de las complejidades de la representación de las gramáticas regulares.

Para resolver el problema de la generación de ER, ofreciendo una vía menos complicada que la escritura directa usando una gramática regular, es necesario partir de las definiciones matemáticas de AFs y ERs propiamente.

Aunque es imprescindible dejar claras también otras definiciones, necesarias para una mejor compresión del tema abordado. Además, es preciso tener claros conceptos tales como: símbolo, alfabeto, palabra, lenguaje, lenguaje regular y máquina de estados finitos. Estos elementos serán discutidos con más profundidad en el siguiente epígrafe.

2.7.1 Algunos conceptos preliminares

Todos esto conceptos y definiciones han sido extraídas del libro de Brena, Autómatas y Lenguajes (Brena, 2003).

La noción más primitiva de símbolo es la que lo define como una representación distinguible de cualquier información. Un símbolo es una entidad indivisible. Pueden ser cualquier carácter, por ejemplo: `w`, `9`, `#`, etc.

Se puede decir que, un alfabeto es un conjunto no vacío de símbolos. Así, el alfabeto del idioma español, `E = {a, b, c, . . . , z}`, es sólo uno de tantos alfabetos posibles. Generalmente se utiliza la notación $\sum$ para representar un alfabeto.

Con los símbolos de un alfabeto es posible formar secuencias o cadenas de caracteres, tales como `mxzxptlk`, `balks`, `r`, etcétera. Las cadenas de caracteres son conocidas también como palabras. Un caso particular de cadena es la palabra vacía, ε, la que no contiene ninguna letra.

Se tendrá en cuenta que, la longitud de una palabra es la cantidad de letras que la conforma, incluyendo las repeticiones, se denota por |w| para una palabra w. Por ejemplo, para la palabra `|entidad|`, su longitud es |w|=7.

Así también, la concatenación de palabras da origen a nuevas palabras cuya longitud es la suma de las longitudes de las palabras originales. Por ejemplo, si w= abra y v= cada, entonces wvbra es la palabra abracadabra. Esta operación es asociativa, pero no conmutativa.

También, se dice que una palabra v es una subcadena de otra palabra w, cuando existen cadenas x, y -posiblemente vacías- tales que xvy= w. Por ejemplo, abra es subcadena de abracadabra, y ε es subcadena de toda palabra.

Entonces, un lenguaje es simplemente un conjunto de palabras. Así, [[abracadabra]] es un lenguaje (de una sola palabra), [[ali, baba, y, sus, cuarenta, ladrones]] es otro.

Dado que los lenguajes son conjuntos, entonces se pueden efectuar con ellos todas las operaciones conocidas sobre conjuntos (unión, intersección, diferencia). Además, la operación de concatenación de lenguajes, es escrita como: L1 • L2, como una extensión de la concatenación de palabras: L1 • L2 = [[w|w= xy, x ∈ L1, y ∈ L2]].

Otra operación, es la llamada estrella de Kleene o cerradura de Kleene (Hopcroft y Ullman, 2001), en honor al matemático norteamericano S. C. Kleene, quien la propuso: Si L es un lenguaje, L* -llamado estrella de Kleene de L- es el más pequeño conjunto que contiene:

- La palabra vacía, ε
- El conjunto L
- Todas las palabras formadas por la concatenación de miembros de L*.

Los lenguajes regulares deben su nombre a que sus palabras contienen regularidades o repeticiones de los mismos símbolos y palabras.

Definición: Un lenguaje L es regular si y sólo si se cumple al menos una de las condiciones siguientes:

- L es finito.
- L es la unión o la concatenación de otros lenguajes regulares R1 y R2, L=R1 ∪ R2 ó L=R1R2 respectivamente.
- L es la estrella de Kleene de algún lenguaje regular, L=R*.

Teorema de Kleene: Un lenguaje es regular si y sólo si es aceptado por algún AF (Hopcroft y Ullman, 2001).

2.7.2 Autómatas finitos

Según (Hopcroft y Ullman, 2001), una máquina de estados finitos puede ser visualizada como un dispositivo con los siguientes componentes:

a) Una cinta de entrada
b) Una cabeza de lectura
c) Un control

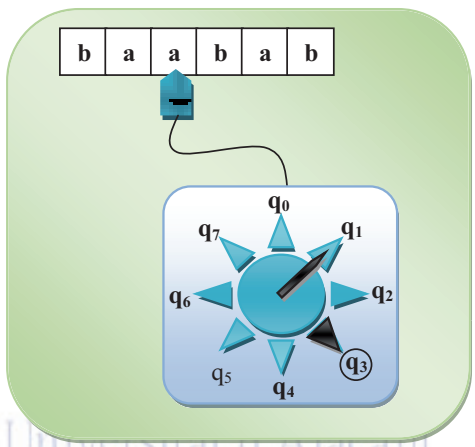


Figura 2.8. Representación general de una máquina de estados finitos.

La cabeza lectora se coloca en los segmentos de cinta que contienen los caracteres que componen la palabra de entrada, y al colocarse sobre un carácter lo lee (apunta al carácter que se encuentra inmediatamente a su derecha) y manda esta información al control (indicado por una carátula de reloj (Figura 2.8), donde se determina cuál es la nueva posición de la aguja. Se supone que existe forma de saber cuándo se acaba la entrada (por ejemplo, al llegar a una posición vacía en la cinta de entrada). La aguja del control puede estar cambiando de posición, y hay algunas posiciones llamadas finales (como la indicada por un punto, q_3) que son consideradas especiales, porque permiten determinar si una palabra es aceptada o rechazada (Brena, 2003).

Según (Brena, 2003), un AF es un modelo matemático de un sistema que recibe una cadena constituida por símbolos de un alfabeto y determina si esa cadena pertenece al lenguaje que el autómata reconoce.

La definición formal que aparece a continuación es la que da Hopcroft en (Hopcroft y Ullman, 2001).

Definición: Un Autómata Finito A es una quíntupla $(Q, \sum, \delta, s, F)$, donde:

- Q es un conjunto de estados.

- $\sum$ es el alfabeto de entrada.
- $s \in K$ es el estado inicial.
- $F \subseteq Q$ es un conjunto de estados finales.
- δ: $K \times \sum \rightarrow Q$ es la función de transición, que a partir de un estado y un símbolo del alfabeto obtiene un nuevo estado.

En el inicio del proceso de reconocimiento el AF se encuentra en el estado inicial, para una cadena de entrada cualquiera, a medida que se procesa cada símbolo de la cadena, cambia de estado de acuerdo a la función de transición. Cuando el último de los símbolos de la cadena de entrada haya sido procesado, el autómata se detiene. Si el estado en el que se detuvo es un estado de aceptación, entonces la cadena pertenece al lenguaje reconocido por el autómata; en caso contrario, la cadena no pertenece a ese lenguaje.

El estado inicial de un AF siempre es único, mientras que los estados finales pueden ser más de uno. Puede ocurrir también que un estado final se corresponda con el mismo estado inicial. La función de transición indica a qué estado se va a pasar sabiendo cuál es el estado actual y el símbolo que se está leyendo.

Ahora bien, si partiendo de un estado y un símbolo del alfabeto hay un y sólo un estado siguiente, entonces el autómata es `determinista` (AFD) (Hopcroft y Ullman, 2001).

Por el contrario, si de cada estado parten varias transiciones con el mismo símbolo del alfabeto, es decir, que para un símbolo hay más de una transición posible al siguiente estado, entonces el autómata es `no determinista` (AFN) (Hopcroft y Ullman, 2001).

Por otra parte, si existen transiciones vacías (o transiciones ε), es decir, que se permite cambiar de estado sin procesar ningún símbolo de la entrada, entonces el autómata es `no determinista con transiciones vacías` (AFN-ε). Cuando el autómata llega a un estado, se encuentra en ese estado y en los estados a los que apunte éste mediante una transición ε.

Dado que los AFN tienen menos restricciones que los AFD, resulta que los AFD son un caso particular de los AFN, por lo que todo AFD es de hecho un AFN. Para todo AFN-ε existe un AFN equivalente y para todo AFN existe un AFD equivalente.

Este proceso de conversión actualmente está resuelto, desde hace tiempo existen algoritmos para transformar un autómata en otro (Moore, 1956). Los AFD son los más sencillos de construir, por tanto,

puede ser útil diseñar un autómata complejo como AFN-ε o AFN, para luego transformarlo en AFD para su implementación, solo sería necesario decidir la forma de su representación.

Existen tres formas de representar un AF. Además de notar un AF a través de su definición formal, es posible representarlo a través de otras notaciones que resultan más cómodas. Las más usuales son:

- **Las Tablas de Transiciones:** donde se muestran por las filas los estados del AF y por las columnas los símbolos del alfabeto y en las intersecciones de fila y columna se muestra el próximo estado, la Tabla 2.2 muestra un ejemplo de tabla de transiciones de un AF determinista con dos estados (S_1 y S_2) y dos símbolos de entrada (0 y 1), donde se puede observar que a partir de cada estado y símbolo del alfabeto se puede pasar a un próximo estado. Tomado de (Hopcroft y Ullman, 2001).

Tabla 2.2 Ejemplo de tabla de transiciones de un AFD.

	0	1
S_1	S_2	S_1
S_2	S_1	S_2

- **Los Diagramas de Transiciones:** puede decirse que es la representación más comprensible para los humanos, dado que es una forma completamente visual donde se pueden observar los estados como círculos, el estado inicial se muestra marcado con una flecha que entra, el estado final se distingue por un doble círculo, pueden coincidir estados inicial y final como el de la Figura 2.9 obtenida de (Hopcroft y Ullman, 2001), que muestra el diagrama equivalente la Tabla 2.2. Las transiciones se representan por flechas, desde el estado de donde parten al estado que llegan, acompañadas de una etiqueta que indica qué símbolo del alfabeto es necesario para que se realice dicha transición.

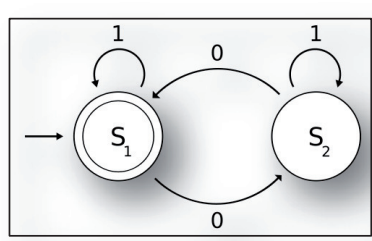


Figura 2.9. Ej. Diagrama de transición de estados, AF determinista.

Este AF está definido sobre el alfabeto Σ= [[0,1]], posee dos estados s_1 y s_2, y sus transiciones son $\delta(s_1,0)= s_2$, $\delta(s_1,1)= s_1$, $\delta(s_2,0)= s_1$ y $\delta(s_2,1)= s_2$. Su estado inicial es s_1, que es también su único estado

final. El lenguaje regular que reconoce puede expresarse mediante la Expresión Regular (00|11|(01|10)(01|10))*, la que como se puede apreciar es un tanto más difícil de interpretar que su respectivo autómata.

- **Las Expresiones Regulares:** es la representación mediante una palabra que define el lenguaje regular aceptado por dicho autómata. Se demuestra que dado un autómata de estados finitos, existe una expresión regular que lo representa. En el siguiente epígrafe se abundará un poco más sobre el tema.

2.7.3 Expresiones Regulares

En este punto, es necesario aclara qué es un Expresión Regular teóricamente hablando. Las siguientes definiciones son tomadas del libro Autómatas y Lenguajes (Brena, 2003), donde se ha modificado la terminología para una mejor comprensión. El símbolo λ (expresión regular vacía) en la bibliografía aparece como ^ y este último tiene un significado especial en la mayoría de los motores de expresiones regulares, por esta razón ha sido cambiado.

Definición: Sea ∑ un alfabeto. El conjunto ER de las expresiones regulares sobre ∑ contiene las cadenas en el alfabeto ∑U [[λ, |, •, *, (,), Ø]] que cumplen con lo siguiente:

a) λ y Ø ∈ ER

b) Si σ ∈ ∑, entonces σ ∈∑ ER.

c) Si E1, E2 ∈ ER, entonces (E1 | E2) ∈ ER, (E1 • E2) ∈ ER, (E1)* ∈ ER.

Definición: El significado de una expresión regular es una función L: ER→2^{Σ^*} (una función que toma como entrada una expresión regular y entrega como salida un lenguaje), definida de la manera siguiente:

a) L(Ø) = Ø (el conjunto vacío)

b) L(λ) = [[ε]] (la palabra vacía)

c) L(σ) = [[σ]], σ ∈ ∑

d) L((R•S)) = L(R)L(S), R, S ∈ ER

e) L((R | S)) = L(R) ∪ L(S), R, S ∈ ER

f) L((R)*) = L(R)*, R ∈ ER

Según se aprecia en (Brena, 2003), al momento de buscar una expresión regular que satisfaga un lenguaje regular particular, es necesario tener en cuenta que debe cumplir dos características:

- **Corrección**: La expresión regular propuesta debe representar solamente las palabras que satisfagan la condición, es decir, no deben representar palabras que no pertenezcan al lenguaje regular.
- **Completez**: La expresión regular propuesta debe representar todas las palabras que satisfagan la condición, es decir, no debe haber palabras que pertenezcan al lenguaje que no sean representadas por dicha expresión regular.

2.7.4 Trabajos relacionados con la generación automática de ER

Existen varios trabajos encaminados a facilitar la generación de expresiones regulares. La mayoría de los investigadores se han planteado resolver este problema mediante herramientas que hagan abstracción de las dificultades de los lenguajes regulares, algunos se limitan a la creación de una interfaz gráfica donde se puede introducir y probar las ER mediante un texto de prueba. En algunos se resaltan con colores las coincidencias para identificar que la expresión está correcta. Otros van más allá y permiten almacenar las expresiones en librerías para reutilizarlas. Después de una búsqueda en varias fuentes de información se detecta que la generalidad de estos trabajos se pueden dividir en dos grupos: aplicaciones de escritorio y aplicaciones web.

2.7.4.1 Aplicaciones de escritorio

Visual RegExp v3.1[29]: Funciona con el motor de expresiones regulares Perl y compatibles. No implementa opción de exportar e importar desde librerías, sólo el texto de la expresión regular desde un fichero de texto. No existe interfaz para diseñar expresiones regulares, hay que teclearla. Presenta Interfaz para probar expresiones regulares, muy útil la utilización de colores para delimitar los grupos de la expresión regular. En cuanto a interfaz para remplazar y etiquetar el texto, sólo posee para remplazar. Obtiene una ER a partir de varios ejemplos de texto, realmente esta opción no se considera muy útil de la forma en que está implementada, debido a que para la mayoría de los casos, el software tendría que `adivinar` lo que el usuario desea obtener como patrón. Algunas ERs pueden consumir mucho tiempo de CPU; esto parece ser causada por el uso combinado de las banderas -all, -inline e -indices. Además, cuando una sub-expresión no coincide (empty match), el último carácter del macheo anterior se colorea. Se distribuye bajo licencia GPL, es open source (código abierto, se refiere a que se distribuye también su código fuente) y está disponible para Windows y Linux.

[29] http://laurent.riesterer.free.fr/regexp/visual_regexp-3.1.exe

RegExp Coach v0.8.5[30]**:** Funciona con el motor de expresiones regulares Perl y compatibles. No implementa opción de exportar e importar desde librerías, sólo el texto de la expresión regular. No existe interfaz para diseñar expresiones regulares, hay que teclearla. Presenta Interfaz para probar expresiones regulares, además se puede comprobar paso a paso. Respecto a interfaz para remplazar y etiquetar el texto, da la posibilidad de remplazar y cortar. Además muestra el árbol de evaluación de la expresión regular. Se distribuye libremente para uso privado y no comercial.

Expresso v3.0[31]**:** Funciona con el motor de expresiones regulares de .NET. Implementa la opción de exportar e importar desde librerías, en dicha librería no se agrupan según ningún criterio, solo se almacenan las expresiones regulares y se pueden adicionar las expresiones que se están probando. Presenta interfaz bastante compleja para diseñar expresiones regulares mediante un menú con los componentes léxicos de una expresión regular. Se escogen a conveniencia del usuario y se insertan en el área de edición de la expresión regular. Como dificultad el usuario tiene que saber exactamente cómo será la expresión regular que desea conformar, pues la interfaz no valida que los componentes se inserten de forma incorrecta, sino que valida que la expresión resultante sea sintácticamente correcta. Presenta interfaz para probar expresiones regulares mostrando sus coincidencias en forma de árbol. También hay interfaz para remplazar y cortar el texto. Genera ensamblados .NET compilados con la expresión encapsulada para ser utilizada por el programador. Realiza pruebas de rendimiento a las acciones de match e iteraciones sobre las coincidencias. Es software privativo, se realizó el análisis con una versión de evaluación.

Regular Expression Editor v1.4.0[32]: Funciona con el motor de expresiones regulares de PHP, que es el mismo de Perl, lo usa internamente para hacer las evaluaciones. No implementa la opción de exportar e importar desde librerías. No existe interfaz para diseñar expresiones regulares, hay que teclearla. Presenta Interfaz para probar expresiones regulares, pero realmente es muy precaria. Respecto a interfaz para remplazar y etiquetar el texto, sólo da la posibilidad de remplazar. Se distribuye utilizando la licencia QPL (Q Public License) debido a que es la que utiliza PHPEdit[33].Ver ANEXO 2.

RegexDesigner v1.0.2727[34]: Utiliza el motor de expresiones regulares de .NET. No implementa opción de exportar e importar desde librerías, sólo guarda en proyecto (*.rep). No existe interfaz para diseñar expresiones regulares, hay que teclearla. Presenta Interfaz para probar expresiones regulares donde se pueden ver coloreadas las coincidencias. Incluye interfaz para remplazar y cortar el texto, pero no para

3030 http://nnm.me/blogs/Kornh/the_regex_coach_v085/
31 http://www.ultrapico.com/Expresso.htm
32 http://www.waterproof.fr/products/RegExpEditor/
33 PHPEdit es un entorno de desarrollo para PHP.
34 http://www.sellsbrothers.com/

etiquetarlo. Genera ensamblados compilados .NET con la expresión encapsulada para ser utilizada por el programador. Además genera código C# listo para utilizarlo. Es gratis y open source.

RegexWorkbench 2.0[35]**:** Utiliza el motor de expresiones regulares.NET. Al parecer implementa opción de exportar e importar desde librerías, es decir, existe el menú para librerías, pero no realizan ninguna acción. Presenta una interfaz para diseñar expresiones regulares mediante un menú de opciones para escoger los componentes léxicos de la ER e ir armándola según convenga. Muy útil la idea de usar tooltips (etiqueta emergente que muestra información contextual) para describir cada parte de la ER. Tiene interfaz para probar expresiones regulares y para remplazar y cortar el texto. Genera ensamblados compilados.NET con la expresión encapsulada para ser utilizada por el programador. Por la forma y distribución que tienen los componentes en la ventana se puede decirse que falta acabado al software, parece ser una versión de prueba. Es gratis y open source.

JRegexpTester 0.37[36]: Funciona con el motor de expresiones regulares de Java. Usa una Base de Datos para almacenar expresiones regulares y contiene la opción de exportar e importar los datos. No presenta ninguna interfaz para diseñar expresiones regulares, hay que teclearla directamente. Se pueden probar expresiones regulares y no tiene interfaz para remplazar y etiquetar el texto. Tiene una opción para dar formato al texto de salida según las coincidencias utilizando la sintaxis de sprintf (método que incluye la plataforma Java para dar formato a una cadena y tiene su sintaxis propia). Se distribuye bajo licencia GPL, es open source y como está desarrollado sobre Java es multiplataforma.

RegexBuddy3[37]: Funciona con múltiples motores de expresiones regulares, incluso uno propio (JGsoft). Posee opción para exportar e importar desde librerías, cada entrada de la librería se almacena código, descripción y ejemplos de coincidencias. Para diseñar expresiones regulares utiliza un TreeView para la estructura jerárquica, se escogen las diferentes partes de las expresiones regulares y se colocan en su debido lugar, se puede arrastrar y colocar para reordenar su posición, al mismo tiempo se va observando la expresión generada. Esta implementación es bastante acertada pues una ER, como cualquier expresión matemática, tiene forma jerárquica debido al agrupamiento por paréntesis. No obstante se considera que no es lo suficientemente clara como para abstraer al usuario del conocimiento del lenguaje regular. Esto se debe a que la información que se manipula en la zona de edición es prácticamente la expresión regular que se está editando, pero separada en partes y cada parte explicada. Tiene interfaz para probar expresiones regulares y para remplazar y cortar el texto. Además se pueden depurar ERs mostrando una traza completa de todos los pasos que se hicieron para evaluar la cadena de prueba. Genera código en diferentes lenguajes y con diferentes motores de expresiones regulares, listo para utilizarlo en situaciones

[35] http://archive.msdn.microsoft.com/RegexWorkbench/Release/ProjectReleases.aspx?ReleaseId=406
[36] http://jregexptester.sourceforge.net/
[37] http://www.regexbuddy.com/buynow.html

de uso común como iterar sobre las coincidencias y grupos de captura sobre un texto. Exporta a formato HTML una descripción de la ER resaltando las diferentes partes al pasar el ratón por encima y mostrando la descripción correspondiente. Es software privativo, se realizó el análisis con una versión de evaluación que tiene funcionalidades limitadas.

2.7.4.2 Aplicaciones sobre web:

Regextester[38]**:** Utiliza el motor de expresiones regulares de .NET para procesar del lado servidor y el de JavaScript para trabajar en el cliente. No existe opción de exportar e importar desde librerías ni interfaz para diseñar expresiones regulares. Se pueden probar expresiones regulares, no así para remplazar y cortar el texto. Se puede cargar el texto desde fichero en el cliente.

Tester de Expresiones Regulares[39]: Utiliza el motor de expresiones regulares de JavaScript. No tiene opción de exportar e importar desde librerías. No se pueden diseñar expresiones regulares. Se pueden probar expresiones regulares y remplazar el texto.

RegexPal[40]: Hecho para el motor de expresiones regulares de JavaScript. No implementa la opción de exportar e importar desde librerías. No existe interfaz para diseñar expresiones regulares. Se pueden probar expresiones regulares. No posee interfaz para remplazar y cortar el texto.

reWork[41]: (13) Hecho completamente en JavaScript, por lo que usa el motor de expresiones regulares de dicho lenguaje. No existe la opción de exportar e importar desde librerías. No se pueden diseñar expresiones regulares. Posee una interfaz para probar expresiones regulares. Se pueden remplazar y cortar el texto y además se pueden ver el grafo de la expresión y las coincidencias paso a paso. Adicionalmente tiene una opción para ver el código necesario para usar expresiones regulares en los lenguajes: JavaScript, PHP, Python y Ruby.

A partir de la búsqueda realizada, se puede decir que no se ha podido encontrar evidencia de la existencia de una herramienta que genere expresiones regulares a partir de su representación gráfica, que permita abstraer al usuario de la complejidad de los lenguajes regulares. Se han encontrado otros tipos de representación entre los que se puede destacar el uso de menú para selección de los componentes léxicos de la ER. También el uso de un objeto TreeView para amar la expresión utilizando una estructura jerárquica.

[38] http://www.regextester.com/
[39] http://regexlib.com/RETester.aspx
[40] http://regexpal.com/
[41] http://osteele.com/tools/rework/

En el siguiente epígrafe se comenta sobre una de las tareas intermedias del PLN en la que se ha investigado, en la que se utiliza ampliamente las ER, el Reconocimiento de Entidades Nombradas.

2.8 Reconocimiento de Entidades Nombradas

El reconocimiento de entidades es una tarea intermedia, a su vez, de otras tareas intermedias y de varias tareas finales del PLN. Las entidades que se analizan en esta investigación se clasifican en las llamadas entidades fuertes y también entidades débiles. Un modelo para la detección de las entidades débiles puede ser consultado con más detalle en (Arevalo Rodriguez, Torruella et al., 2004). Las entidades fuertes han sido más estudiadas y existen un mayor número de trabajos dedicados a su reconocimiento y clasificación. En este epígrafe se estará viendo en detalles los principales elementos de este proceso.

2.8.1 Diferentes técnicas utilizadas para el reconocimiento de entidades

De manera general los sistemas que se dedican al reconocimiento de entidades tienen una estructura general como la que se muestra en la Figura 2.10

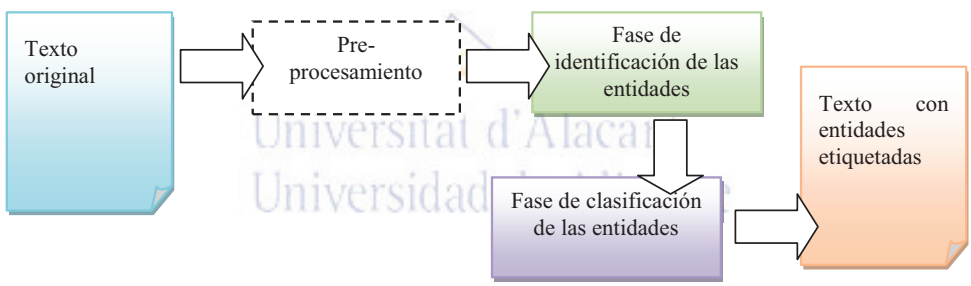


Figura 2.10. Esquema general de un sistema para el REN

El proceso consiste en tomar un texto en su forma original y realizar un pre-procesamiento que incluye varias tareas encaminadas a facilitar el trabajo de las siguientes etapas. Algunos autores (Daelemans, Zavrel et al., 2004; Fernández, Muñoz et al., 2004) han tratado de eliminar o reducir al mínimo este proceso, en aras de disminuir el tiempo computacional de estos sistemas. Posteriormente al pre-procesamiento se pasa a la fase de identificar las posibles entidades. Luego estas entidades, previamente identificadas, pasan por una fase de clasificación y finalmente la salida es un texto con las entidades etiquetadas con su respectiva clase.

Resumiendo, el proceso de reconocimiento de entidades se centro, fundamentalmente en sus dos fases más importantes:

- **Fase de identificación de la entidad**.- Identificación del principio y final de la entidad.
- **Fase de clasificación de la entidad**.- Asignación del tipo de entidad o clase a la que pertenece cada entidad identificada. Las llamadas entidades fuertes (persona, organización, localidad, etc.). Las denominadas entidades débiles, Según (Arevalo Rodriguez, Torruella et al., 2004) consistentes en la unión de en un disparador y opcionalmente una entidad fuerte y un determinante (ej. el presidente de Microsoft). Estos autores plantean la existencia diferentes tipos de entidades débiles según su complejidad sintáctica y semántica.

En los últimos años se han aplicado muchas técnicas y métodos encaminados al perfeccionamiento de todas las fases de este proceso. Sin dudas las más importantes, la identificación y clasificación de las entidades, han sido atacadas con técnicas que se dividen en tres grandes grupos, basadas en conocimiento, las basadas en aprendizaje e híbridas (una mezcla de las dos anteriores):

I. Técnicas basadas en conocimiento:

a) Técnicas que utilizan diccionarios o listas.

b) Técnicas que utilizan reglas.

A continuación se describen brevemente estos dos tipos de técnicas.

a) Técnicas que utilizan diccionarios o listas:

Las aproximaciones basadas en búsquedas en diccionarios o en listas son sistemas que reconocen sólo las entidades que están almacenadas en sus listas (enciclopedias geográficas, bases de datos de nombres, listas de nombres, etc.)

Puntos fuertes: Simples, rápidos, independientes del lenguaje, fáciles de redirigir.

Puntos débiles: Recolección y mantenimiento de las listas, no pueden solucionar variantes de nombres, y no pueden resolver ambigüedades, por ejemplo la palabra África, pudiera estar en un diccionario de países, pero sucede que África también se usa como nombre propio de mujer.

b) Técnicas que utilizan reglas:

Las aproximaciones basadas en reglas son sistemas que reconocen las entidades que se ajustan a las reglas definidas, apoyándose en algunos tipos de listas o diccionarios (abreviaturas, nombres, etc.).

Para la fase de identificación de entidades muchos sistemas han utilizado reglas. Este proceso consiste en encontrar los fragmentos de texto que forman la entidad, o sea, su longitud. Los sistemas de REN que se basan en reglas para la identificación de entidades han obtenido mejores resultados de cobertura que otros, siendo una de las técnicas más usadas en esta fase.

Para que un fragmento de texto determinado sea una entidad debe cumplir con determinados requisitos, por lo que a partir de estos se pueden elaborar reglas R(x) que sean capaces de encontrar las entidades que cumplan sus exigencias.

$R(x) \rightarrow [TEXTO] = \{x1, x2, x3, x4,\ldots\}$

Donde:

R(x)-Regla para detectar todas las entidades x.

→ - Aplicada a.

{x1, x2, x3,…} - Entidades encontradas.

También, para la fase de clasificación se han utilizado ampliamente las reglas. Aprovechando la propia naturaleza de los lenguajes, los que siempre poseen reglas gramaticales, los sistemas para el REN que usan esta técnica, hacen sofisticados paquetes de reglas que se aplicarán a las entidades previamente identificadas.

Puntos débiles: Dependientes del lenguaje, las reglas generalmente son complejas de representar y muchas veces para editarlas hay que ir al código fuente.

Puntos fuertes: Pueden solucionar variantes de nombres, y pueden resolver ambigüedades. No necesitan de corpus anotados.

II. Basadas en aprendizaje (estadísticos)

Actualmente la técnica dominante en el REN es el aprendizaje supervisado. Dentro de esta técnica se incluyen:

1. Las cadenas ocultas de Markov (Bikel, Miller et al., 1997)
2. Árboles de decisión (Sekine, 1998)
3. Modelos de probabilidad de máxima entropía (Borthwick, Sterling et al., 1998)
4. Máquinas de soporte vectorial (Asahara y Matsumoto, 2003)
5. Campos aleatorios condicionales (McCallum y Li, 2003)

III. Técnicas híbridas

Estas técnicas surgen a partir de la combinación de los dos tipos de técnicas anteriormente explicadas. En este caso lo que se pretende es aprovechar lo mejor de cada una de ellas.

Otras técnicas de aprendizaje también han sido aplicadas, aunque en menor cuantía y con resultados que no distan mucho de los anteriores. Debido a que es una de las técnicas fundamentales utilizadas en esta investigación, a continuación se hace una breve descripción de la técnica llamada Modelos de Probabilidad Condicional de Máxima Entropía.

2.8.1.1 Los modelos de probabilidad de máxima entropía

Dentro de las técnicas basadas en aprendizaje se encuentran muchos métodos de clasificación aplicados al PLN. En este caso uno de los modelos utilizados para el REN ha sido el de Probabilidad Condicional de Máxima Entropía. Los que definen funciones de clasificación a partir de un conjunto de ejemplos previamente anotados. Es, por tanto, un método de aprendizaje supervisado basado en ejemplos (Fernández, Muñoz et al., 2004).

Uno de los primeros trabajos que utilizan técnicas de ME en tareas de PLN, (concretamente en la tarea de speech recognition sp. reconocimiento del habla) son los de Lau Raymond descritos en (Lau, Rosenfeld et al., 1993; Lau, 1994). En una revisión histórica interesante (Berger, Pietra et al., 1996), citan a Laplace considerándolo el padre de ME, al haber enunciado su "*principio de la razón insuficiente*" hace más de dos siglos: *"...cuando no tenemos información para distinguir entre la probabilidad de dos eventos, la mejor estrategia es considerarlos a los dos equiprobables"*. Finalmente, en (Jaynes, 1990) se plantea: *"... el hecho de que cierta distribución de probabilidad maximice la entropía sujeta a ciertas restricciones que representan nuestra información incompleta, es la propiedad fundamental que justifica el uso de tal distribución para la inferencia; está de acuerdo con todo aquello que es conocido pero evita cuidadosamente asumir nada que sea desconocido..."*

En realidad, el modelado basado en el principio de ME es un método general que se puede utilizar en cualquier tarea que se aborde desde la perspectiva del aprendizaje estadístico automático.

Los ejemplos se representan de forma codificada mediante funciones que se activan por la presencia de un determinada propiedad (se dice de éste, y de todos los que utilizan esta técnica, que es un método basado en la representación por rasgos o atributos). Los modelos de ME proporcionan un entorno para la integración de, prácticamente, cualquier tipo de información de una forma sencilla. De hecho, la principal preocupación del usuario de esta técnica consiste en la detección de cuáles son las fuentes de información

útiles para el problema que pretende resolver; es el método el que se encarga de componer la función de clasificación a partir de la muestra suministrada (Fernández, Muñoz et al., 2004).

Ratnaparkhi, en su trabajo de Tesis Doctoral (Ratnaparki, Ittycheriah et al., 2001), emplea con éxito ME en varias tareas del PLN: detección de los límites de la frase, etiquetado gramatical (part-of-speech-tagging), análisis sintáctico, dependencia de los sintagmas preposicionales (preposition phrase attachment) y clasificación o categorización de textos.

Así mismo, los sistemas de ME han sido utilizados, entre otros, en reconocimiento de entidades (Borthwick, Sterling et al., 1998.), traducción automática estadística (Sanz, Guillena et al., 2004) clasificación de textos, sistemas de BR (Ratnaparki, Ittycheriah et al., 2001). Como extensión de los modelos de Markov, éstos se han combinado con los de ME dando lugar a un nuevo modelo general, los modelos de Markov de ME (McCallum, Freitag et al., 2000), aplicados en este caso a la EI, concretamente a la segmentación de listas de "preguntas más frecuentes" en preguntas y respuestas. Desarrollos posteriores de este modelo pueden encontrarse en (Lafferty, McCallum et al., 2001), donde es usado en tareas de POS-tagging, y detección de sintagmas nominales.

La implementación de los sistemas de ME utilizada en este trabajo es la desarrollada para un sistema de resolución de la ambigüedad semántica de las palabras (Suárez, 2005).

Según (McCallum, Freitag et al., 2000) los modelos de probabilidad condicional de ME son métodos que se pueden aplicar en cualquier tarea que se trate desde la óptica del aprendizaje estadístico automático. Suárez en su tesis doctoral (Suárez, 2004), plantea que es un método basado en la representación por rasgos o atributos en los que la detección de cuáles son las fuentes de información útiles para dar solución al problema a resolver, son la preocupación fundamental de los usuarios de este método.

Si se habla de que los modelos de probabilidad de ME son un método de aprendizaje supervisado que utiliza cualquier tipo de información, entonces es necesario definir cómo se va a facilitar esta información.

Según McCallum en su artículo titulado "*Maximum Entropy Markov Models for Information Extraction and Segmentation*" (McCallum, Freitag et al., 2000), la potencia de la ME radica en la codificación previa de los datos utilizados para construir el clasificador.

Como se ha planteado anteriormente, los modelos de probabilidad condicional de ME, en lo adelante (MPCME), son un método de aprendizaje supervisado que utiliza cualquier tipo de información, entonces es necesario definir cómo se va a facilitar esta información. Según Suárez en su trabajo de tesis doctoral (Suárez, 2004), la potencia de los MPCME radica en la codificación previa de los datos utilizados para construir el clasificador.

Si se necesita obtener, por ejemplo, un clasificador que pudiera decidir si una palabra que comience con mayúscula (posible entidad) es de tipo persona, es necesario hacer una caracterización de esta entidad basándose en un conjunto de atributos o propiedades que poseerá o no. Se podrían utilizar, en este caso, las evidencias externas o internas (McDonald, 1996) que ofrecen información sobre la entidad analizada. Esto podría ser analizar las palabras cercanas a la entidad en una ventana determinada, más adelante se abundará algo más sobre el uso de las evidencias externas e internas. Se pueden seleccionar muchos otros atributos para esta tarea.

En el siguiente ejemplo se puede analizar cómo se resolvería el caso de la clasificación de una entidad de tipo persona. Para ello se utilizarán tres atributos: si existe una evidencia externa a la entidad del tipo disparador de persona en las dos palabras que anteceden a dicha entidad. Otro atributo será la posible existencia de evidencias internas, por ejemplo, que la entidad esté acompañada de una abreviatura que indica que, indiscutiblemente, se está en presencia de una persona, por ejemplo: `Ing. Antonio Fernández Orquín.`

Partiendo de la siguiente porción de texto se hará el análisis y extracción de los atributos que permitan evaluar si existe una entidad de tipo persona.

`El Ing. Antonio Fernández Orquín está realizando sus estudios de doctorado en la universidad de Alicante, Después de culminar la primera etapa, el señor Dr. Rafael Muñoz Guillena aceptado ser el tutor de este estudiante.`

En este pequeño párrafo se pueden observar varias entidades. La primera sería `Ing. Antonio Fernández Orquín`, en segundo lugar aparece la entidad `Alicante` y por último `Dr. Rafael Muñoz Guillena`. Como se ha explicado anteriormente se está buscando, para este ejemplo, sólo los casos de entidades de tipo persona. `Alicante` sería una localidad por lo que no sería identificada como una posible clase en este caso.

Se tiene entonces, según los atributos que se seleccionaron, las posibilidades siguientes:

a) < “existe evidencia externa”, “existe evidencia interna” > <Persona>

b) <_, “existe evidencia interna”><Persona>

c) <_, _><No Persona>

Si se observa bien, el primer caso, (b), corresponde a la entidad uno encontrada: `Ing. Antonio Fernández Orquín`. En este caso no se ha encontrado evidencia externa que indique la existencia de una persona en las palabras que anteceden a la entidad, pero sí se encontró una evidencia interna del tipo:

Ing. (abreviatura de Ingeniero), que indica que la entidad es una persona. Al final, solo bastaría entrenar un modelo de ME que sea capaz de aprender esta clasificación.

Como se puede ver en (Suárez, 2004), las entidades son representadas por vectores de valores de verdad, cada uno será calculado partiendo de una función o atributo (ver ecuación 2.21).

$$f(x,c) = \begin{Bmatrix} 1 \; si \; c = c' \; y \; cp(x) = cierto \\ 0 \; en \; otro \; caso \end{Bmatrix} \quad 2.1$$

En esta ecuación la condición `c=c'` es la que liga el atributo a una clase concreta, teniendo que `cp` es un predicado cualquiera. Cada instancia del objeto a clasificar se sustituirá en (x), y la clase a que pertenece en `c`.

$$f_1(x,c) = \begin{Bmatrix} 1 \; si \; c = persona \; y \; Existe \; evidencia \; externa = verdadero \\ 0 \; en \; otro \; caso \end{Bmatrix} \quad 2.2$$

$$f_2(x,c) = \begin{Bmatrix} 1 \; si \; c = persona \; y \; Existe \; evidencia \; interna = verdadero \\ 0 \; en \; otro \; caso \end{Bmatrix} \quad 2.3$$

Partiendo de aquí se obtienen los valores de estas funciones para la entidad 1

f_1(entidad$_1$, persona)=0

f_2(entidad$_1$, persona)=1

De esta forma la representación completa del conjunto de ejemplos sería:

Entidad$_1$:01, Entidad$_2$:00, Entidad$_3$:11

Como bien se plantea en (Suárez, 2004) "*se denomina contexto al conjunto de información que se considera útil para caracterizar a un determinado objeto*". En el caso presentado anteriormente el contexto para la entidad$_1$, sería <no hay evidencia externa, existe evidencia interna>.

Suárez plantea que "el contexto es el conjunto de propiedades que tiene un individuo, tal y como las entendemos habitualmente, y un atributo es una de estas propiedades donde, además se verifica su pertenencia a una clase o tipo de objeto".

<_, Existe evidencia Interna><Persona>:01

<_, _><No Persona>:00

<Existe evidencia externa, Existe evidencia Interna><Persona>:11

Aprendizaje y clasificación.

Tal como se expone en (Suárez, 2004), una vez que se puede obtener una función que informe de la probabilidad de que una entidad x pertenece a una determinada clase c, p(c/x); entonces es necesario definir un clasificador, cl(x), que elija la clase con mayor probabilidad, ósea, identificando la presencia de evidencias de un tipo determinado se podría definir de qué tipo pudiera ser una entidad.

$$cl(x) = \arg máx_c \; p(c/x) \qquad 2.4$$

Donde:

$p(x/c) =$ Es la probabilidad de que el alumno x pertenezca a la clase c

A lo que se le llamará aprendizaje es a la definición del modelo de probabilidad por el que se va a obtener los valores de p(x/c). Dicho modelo debe satisfacer una serie de restricciones para adaptarse al modelo de máxima entropía (Suárez, 2004)

Se pudiera decir que el aprendizaje no es más que la asignación los pesos correspondientes a cada uno de atributos definidos. La función de probabilidad quedaría definida con los pesos y atributos de manera que, al encontrar una entidad desconocida, se puede determinar las probabilidades de que pertenezca a cada una de las clases que se analizan y seleccionar aquella que posea el valor máximo.

El proceso de clasificación, según (Suárez, 2004), "*consiste en comprobar el valor de p(x/c) para todas las clases*":

$p(\text{persona} \mid A_x)=p(01)=a$

$p(\text{no\_persona} \mid A_x)=p(00)=b$

Queda claro que si a>b la entidad A_x quedará clasificada como persona.

Como un elemento importante, para ser utilizado tanto en las técnicas basas en conocimiento como en aprendizaje, se encuentra las evidencias. McDonald en 1995 propuso una idea basada en evidencias para la detección de las entidades. Estas podían ser externas o internas (McDonald, 1996).

Las evidencias externas: Son palabras o nombres que acompañan a determinadas entidades en determinados contextos. Ej. `El comandante Fidel Castro` → comandante es una evidencia externa de que la entidad `Fidel Castro`, que le sigue a continuación, es una persona.

Las evidencias internas: Como las entidades suelen tener una determinada estructura interna, algunos componentes de esta estructura pueden estar en determinadas listas o diccionarios y otros pueden estar

supuestos. Ej. El `Ing. Antonio Fernández` → La abreviatura `Ing`. Es una evidencia interna de que la entidad `Ing. Antonio Fernández` es de tipo persona.

Aunque en realidad el análisis de las evidencias internas y externas aporta bastante al esclarecimiento de la clasificación de las entidades, también puede ser un arma de doble filo. Suponga que se tiene una expresión como: `el gerente del Banco Financiero Internacional`, dicha expresión será reconocida por la mayoría de los sistemas como [`Banco Financiero Internacional`] y clasificada como una organización, sin embargo es evidente que se está hablando de una persona. Si se tienen en cuenta los postulados de McDonald al basarse en la evidencia externa `el gerente`, habría una contradicción con la evidencia interna `Banco`. De esta manera un programa que intente clasificar la entidad pudiera errar en su propósito.

En (Arevalo Rodriguez, Torruella et al., 2004) se hace una propuesta interesante a cerca de la definición de dos grupos de entidades fuertes y entidades débiles. El primer grupo comprende los nombres propios, mientras que el segundo contendría las palabras llamadas disparadores y opcionalmente nombres propios. Estos autores enfocan su trabajo en la detección y clasificación de las entidades débiles.

En este trabajo se ha apostado por el método combinado, utilizándose técnicas basadas en el conocimiento (diccionarios y reglas) para la detección-clasificación y técnicas basadas en el aprendizaje (ME), para la clasificación de las entidades fuertes. También se reconocen y clasifican algunas entidades débiles, sobre todo aquellas que no presentan ambigüedad y pueden ser identificadas con absoluta seguridad. De esta forma el proceso de reconocimiento de entidades pasa por dos fases.

2.8.2 Problemas y soluciones en las fases de detección y clasificación.

Para la fase de identificación, el sistema se auxilia de las reglas de colocación de la mayúscula en el idioma español. Para una mejor comprensión se utilizan diferentes abreviaturas:

Tabla 2.3. Abreviaturas para designar los elementos constituyentes de las reglas.

Abreviatura	Significado
(PalMay)	para palabras en mayúscula
(CristNom)	para nombres cristianos
(NomAbr)	para abreviatura de nombres
(LetraMay)	para letra mayúscula
(PREP)	para las preposiciones
(Num)	para los números
(Disp)	para los disparadores
(Adj)	para los adjetivos
(PalGeo)	para lugares geográficos
(NomCiudad)	para nombres de ciudad
(NomPais)	para nombres de países

(VerbPers)	para verbo personal
(Art)	para los artículos
(PntoCard)	para los puntos cardinales
(Prof)	para las profesiones.

A continuación se describen una serie de problemas encontrados y su posible solución.

Problemas:

1. Comienzo de frases.

Debida a las reglas de la colocación de las mayúsculas en los inicios de frase es obligatorio colocar la primera palabra en mayúscula. Al ser esta la forma que tenemos para identificar una entidad, surge el problema de tener, en el inicio de frase, algunas palabras que introducirían cierta ambigüedad, pues no todas las palabras que inician con mayúscula y que se encuentren en esta posición, tienen que ser necesariamente una entidad.

Ejemplo: `La Coca-cola es el refresco más`…, en este caso el artículo `La` no pertenece a la entidad sin embargo está escrito con mayúscula porque aparece al inicio de la frase. También pudiera suceder que la palabra que inicia la frase no sea una entidad, pero sí forme parte de ella, ejemplo: `La Habana`, o `La Coruña`, en este caso artículo `La` no es una entidad, a pesar de estar en mayúscula, pero forma parte de la entidad, por lo tanto habría que etiquetar `<ENT TYPE=>La Coruña</ENT>`, posteriormente al clasificar quedaría `<ENT TYPE=LOC>La Coruña</ENT>`.

Posible solución:

Una posible solución sería comparar con otras apariciones de entidades similares. Normalmente si es un artículo no suele formar parte de la entidad.

2. Aparición de preposiciones.

Artículos definidos y demostrativos entre palabras en mayúsculas, ejemplo: María de los Ángeles, en este caso la preposición `de` y el artículo `los` formarían parte de una entidad compuesta, a pesar de estar en minúscula.

Posible solución:

Verificar las palabras que siguen a las que inicien con mayúscula, en caso de que coincidieran con algunos de los artículos y/o preposiciones, u otras palabras, que pueden formar parte de una entidad (aunque se escriban con minúscula), comprobar que les sigue una posible entidad (porque inicia con mayúscula) y capturar la totalidad del conjunto.

3. Ambigüedad estructural:

Otro problema muy frecuente es la coordinación de entidades a través de conjunciones o preposiciones u otros elementos. Esta ambigüedad estructural crea incertidumbre de si se está en presencia dos entidades diferentes o una sola entidad compuesta, por ejemplo: `Cuervo & Sobrinos`, `Construcciones y Contratas S.A.`, representan una sola entidad. Por el contrario: `IBM y Microsoft`, podría generar duda para un sistema de REN, pues no podría identificar es una o dos entidades las que se están analizando.

Posible solución:

Se puede intentar solucionar estudiando el número del verbo, de esta forma, si es plural se sabe que se trata de dos entidades y no de una compuesta que posee la conjunción `y`. Por ejemplo: `…las compañías IBM y Microsoft`.

Otra posible solución sería el estudio de disparadores a ambos lados de la conjunción. Si ambas subcadenas contienen disparadores, entonces se trata de dos entidades independientes, por ejemplo: `ESPAC S.A. y Construcciones S.A.`

De otra forma, si la subcadena de la izquierda tiene disparador al final, serán dos entidades (`ESPAC S.A. y Microsoft`). En cambio, si la subcadena de la derecha tiene disparador y la de la izquierda una actividad, serán dos entidades (`Clínica San Carlos y Hércules C.F.`).

Por otro lado, si existe acrónimo en alguna de las subcadena, serán dos entidades (`CAM y Hércules`). En cualquier otro caso, sería una sola entidad o se tendría que desambiguar por el verbo, ejemplo: `Construcciones y Contratas S.A. adquiere…`

4. Ambigüedad semántica.

La ambigüedad semántica se ve con relativa frecuencia, ejemplo: `Don Algodón`. Aquí cabría preguntarse si es el nombre de una persona o el de una empresa. O como en el siguiente ejemplo `John F. Kennedy`, pudiera ser una persona, pero también un aeropuerto.

Posible solución:

Incorporar información semántica asociada al verbo y una ontología, ejemplo: `Aterriza en el John F. Kennedy…`, en este caso queda claro que es una localidad, pues el verbo indica que no puede ser una persona, ya que las personas no aterrizan en los aeropuertos.

Otro ejemplo es: `Adolfo Domínguez visitó`..., en este caso, a través del análisis del verbo se puede identificar que se trata de una entidad tipo persona.

La utilización de las evidencias en la clasificación de entidades puede auxiliar mucho en esta tarea. Se destacan claramente dos grupos de evidencia, las externas y las internas. Estas evidencias también se les conocen en la literatura sobre el tema como disparadores.

a) Evidencias internas.

Para representar las reglas de estructuración de las posibles entidades se han tomado las ideas expresadas en (Llopis, 1988). Los paréntesis indican que las palabras son opcionales, es decir, que podrán aparecer o no en el patrón. Los corchetes indican que sólo aparecerá una de las palabras del conjunto. Las llaves indican que habrá una repetición de una o más palabras en la regla.

Para Personas

Existen algunas evidencias internas que ayudan a identificar la entidad como de tipo persona son:

Título (Sr., D.,) + {PalMay} → ejemplo: Sr. Gómez.

Titulo (Sr., D.,) + CristNom + {PalMay} → ejemplo: D. Rafael Muñoz.

CristNom + LetraMay + "." + {PalMay} → ejemplo: Antonio C. Fernández.

CristNom + NomAbr + "." + {PalMay} → ejemplo: Gilberto Sta. Rosa.

Título (Sr., D.,) + PREP + {PalMay} → ejemplo: Sr. De Pedro.

Título (Sr., D.,) + PREP + ART + {PalMay} → ejemplo: Sr. De la Fuente.

Título (Sr., D.,) + PalMay + CONJ + {PalMay} → ejemplo: Sr. Ramón y Cajal.

Para Organizaciones

{PalMay} + DispOrg [S.A., S.L.,] → ejemplo: Coca-Cola S.A.

DispOrg (Sr., D.,) + {PalMay} → ejemplo: Corp. Algodón.

Siglas + DispOrg [S.A., S.L.,] → ejemplo: IBM S.A.

Actividad + {PalMay} → ejemplo: Bar Pedro.

{PalMay}+ PREP + {PalMay} + DispOrg (S.A., S.L.,) → ejemplo: Casa de Cultura.

{PalMay} + NUM + DispOrg [S.A., S.L.,] → ejemplo: Tien 21 S.A.

PalMay+CONJ + PalMay + Disp [S.A., S.L.,] → ejemplo: Construcciones y Contratas S.A.

Para Localidades

PalGeo + {PalMay} → ejemplo: Cabo San Antonio.

DispLoc [C.,Pza.,] + {PalMay} → ejemplo: Pza. Manila.

DispLoc [C., Pza.,] + {PalMay} + "," + NUM +"," + NomCiudad → ejemplo: Pza. Manila, 2, Alicante.

NomCiudad → ejemplo: Alicante.

NomPais → ejemplo: España.

A continuación se verán los ejemplos de reglas para las evidencias externas.

b) Evidencias externas.

Para Personas

Art + Profesión + {PalMay} → ejemplo: el jugador Alfaro

NomCris + "," + Prof + PREP → ejemplo: Ricardo, trabajador de…

NomCris + "," + Prof + texto asociado → ejemplo: Ronaldo, jugador del Real Madrid

{PalMay} + "," + Prof → ejemplo: Alfaro, jugador...

{PalMay} + ... + VerboPers → ejemplo: Alfaro marcó...

Para Organizaciones

Profesión + PREP + {PalMay} → ejemplo: director de Coca-Cola SA.

"Compañía" + {PalMay} → ejemplo: Compañía Dragados.

"Compañía" + PREP + {PalMay} → ejemplo: Compañía de Seguros.

Art + DispOrg + {PalMay} → ejemplo: la empresa Noel Fernández.

Para Localidades

PREP + ART + PNTOCARD + {PalMay} → ejemplo: Al sur de Alicante.

{PalMay} + "es" + ART+ ADJ + [PalGeo] → ejemplo: *Ness*, es un gran lago.

La identificación de las entidades no tiene la gran complejidad de la clasificación, pero no deja de ser un problema el poder detectar correctamente el inicio y fin de una entidad. La dificultad principal del REN radica en poder asignar una correcta clasificación a las entidades. En los últimos años se ha visto una

tendencia generalizada a hacer uso de la combinación de métodos con el objetivo de lograr mejores resultados.

Aunque el uso de reglas para el proceso de identificar entidades puede lograr buenos resultados, estas al estar programadas en el código fuente, provocan que el sistema sea dependiente del lenguaje y modificarlas implicaría transformar los ficheros fuentes del programa.

Una solución a este problema podría ser guardar los fragmentos de códigos que representan cada una de las reglas en ficheros o bases de datos, pero esto no sería de gran utilidad por lo complicado que se hace importar estos textos y transformarlos en código para utilizarlos posteriormente. Además los ficheros resultarían muy extensos y de estar todos los fragmentos de código en el mismo fichero, resultaría difícil delimitar donde termina o empieza cada uno.

2.8.3 Trabajos relacionados en la temática

Uno de los primeros trabajos en la temática del REN es el descrito en el artículo: Extracting Company Name from Text (Rau, 1991), presentado en la sexta Conferencia en Aplicaciones de la Inteligencia Artificial IEEE. Liza F. Rau, hace una descripción detallada en este trabajo de un algoritmo implementado para extraer automáticamente nombres de empresas de noticias financieras. Implementa un algoritmo heurístico que combinan, listas de excepciones y análisis extenso de corpus. El algoritmo genera las variaciones más probables de estos nombres, para su uso en una posible recuperación posterior. Fue probado en más de un millón de palabras de noticias financieras reales, el sistema extrajo miles de nombres de empresas con más de un 95% de precisión.

Después de este primer trabajo, surgen otros de menor importancia en la temática y no es hasta noviembre de 1995 que, por primera vez, se incluye esta tarea como parte del ejercicio de evaluación de la Sixth Message Understanding Conference (MUC-6)[42]. Como las cinco anteriores MUCs, fue organizada por Beth Sundheim del Grupo de Investigación y Desarrollo Naval (NRaD) (Grishman y Sundheim, 1996). Estas conferencias, que han introducido la evaluación de los sistemas de EI aplicados a las tareas comunes, fueron fundadas por DARPA (Defense Advanced Research Projects Agency) para medir y fomentar el progreso en el área de la EI.

Las MUCs anteriores se enfocaron en las tareas simples de la EI: analizar textos libres, identificar eventos de un tipo específico, y rellenar plantillas de bases de datos con información asociada a cada uno de estos eventos. Con el transcurso de las cinco primeras conferencias, las tareas y plantillas se han ido complicando progresivamente.

[42] http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html

En la MUC-5, se define un amplio conjunto de objetivos para las siguientes MUCs: insertar sistemas de EI con una gran portabilidad en nuevos dominios, y alentar a los trabajos más básicos en el análisis de lenguaje natural mediante el suministro de algunas tecnologías básicas del análisis del lenguaje.

Siguiendo esta línea, NYU (New York University) y NRaD trabajan juntos para desarrollar especificaciones para la siguientes cuatro tareas de evaluación:

- Reconocimiento de entidades.
- Correferencia.
- Elementos de plantillas.
- Escenarios de la plantillas (extracción tradicional de la información).

Estas tareas fueron refinadas en 1994 y principios de 1995, a través del procesamiento de corpus anotados y extensas discusiones vía e-mail entre los planificadores del MUC-6 y el comité de etiquetadores. Posteriormente se realizó una evaluación primaria, que tuvo lugar en abril de 1995.

La evaluación formal de la MUC-6 se realizó en septiembre de 1995 y el evento se desarrolla en Columbia, Maryland en noviembre de 1995.

La tarea para el Reconocimiento de Entidades de la MUC-6 incluía reconocimiento (para personas, organizaciones y localidades), ubicar nombres, expresiones de tiempo, y algunos tipos de expresiones numéricas. Esta tarea está orientada a tener un valor práctico (en textos anotados que permitan la búsqueda de nombres, lugares, fechas, etc.) y a ser un componente esencial de muchas tareas de procesamiento del lenguaje, como la EI (Voorhees, 2001).

Después de 1995, los sistemas que implementan el REN han sido desarrollados para algunos idiomas del continente europeo y unos pocos del continente asiático.

En esta etapa, se hace mención a dos importantes estudios que han desarrollado un sistema de REN multilingüe, Palmer y Day usaron el método estadístico para encontrar entidades con nombre en artículos escritos en chino, inglés, francés, japonés, portugués y español (Palmer y Day, 1997). Encontraron que la dificultad de la tarea era diferente para los seis idiomas, pero que gran parte de las tareas podían ser realizadas con métodos simples.

En (Cucerzan y Yarowsky, 1999.), se describe un artículo sobre el reconocimiento de entidades independiente del idioma, Cucerzan y Yarowsky utilizaron pistas morfológicas y contextuales para el REN en inglés, griego, hindú, rumano y turco. Con una mínima supervisión, obtuvieron F-medidas globales entre 40% y 70%, en dependencia del idioma. (Sang y Meulder, 2003).

Parece ser que el año 1995 fue el despegue de los sistemas de REN, pues luego continúa en ascenso en los años subsiguientes. Ejemplo de esto lo constituye la celebración del HUB-4 (Chinchor, Robinson et al., 1998), la MUC-7 y el MET-2 (Chinchor, 1999), el IREX (Sekine y Isahara, 2000).

En este período, como referencia se tomaron también los trabajos presentados en las CoNLL, ya que estas fomentan el reconocimiento de entidades con técnicas basadas en aprendizaje.

En la conferencia CoNLL-2002 (Tjong Kim Sang, 2002), se presentaron 12 sistemas diferentes que implementaban técnicas basadas en aprendizaje y se probaron con datos en español y alemán. En mejor valor de precisión lo obtienen el trabajo (Carreras, Marquez et al., 2002) con un valor de 81.38%

Más adelante, precisamente en el 2003 la tarea del CoNLL-2003 (Tjong Kim Sang y De Meulder, 2003) consistió en el reconocimiento de entidades con independencia del idioma. La competencia se concentró en cuatro tipos fundamentales de entidades: persona, localidad, organización y miscelánea. Los participantes de este evento presentaron datos probados y entrenados para dos idiomas al menos. Estos datos se usaron para elaborar sistemas de reconocimiento de entidades que incluyeran un componente basado en aprendizaje. Para cada idioma se proporcionó información adicional (listas de nombres y datos no anotados).

En CoNLL-2003 participaron 16 sistemas. Utilizaron una amplia variedad de técnicas de aprendizaje y diferentes conjuntos de características. El mejor resultado en la precisión alcanzó un 88.99% para el corpus en inglés. Los resultados para esta competición fueron bastante buenos, excepto para la entidad tipo Organización en la que solo se alcanza un 68.035%.

En siguiente año, la conferencia ACE en el 2004, propone un procedimiento complejo de evaluación. Incluyendo diversas cuestiones a evaluar (coincidencia parcial, tipo incorrecto, etc.). Según se plantea en (Nadeau y Sekine, 2007), la definición de la tarea ACE también está más elaborada que las tareas anteriores a nivel de la entidad con nombre subtipos, clases, así como las entidades mencionadas (correferencias), y más, pero al final estos elementos suplementarios se ignora.

Luego surgieron otros eventos en los que también se trató el tema del REN, estos son los casos del HAREM (Santos, Seco et al., 2006) en la conferencia LREC[43]. Según (Nadeau y Sekine, 2007) esta conferencia ha estado tratando el tema desde el año 2000.

HAREM, fue organizado por lingüistas y diez participantes de seis diferentes países, quienes enviaron 15 variantes de sus sistemas. De manera general, los participantes tenían 48 horas para etiquetar una amplia y

[43] http://www.lrec-conf.org/

variada colección de textos que incluía 1202 documentos de diversos géneros en portugués (Santos, Seco et al., 2006). Según Santos, HAREM introduce varias características innovadoras:

- Separación entre la identificación y la clasificación de los NE. Una propuesta que interesa a esta investigación, pues a partir del 2004 en los trabajo (Fernández, Muñoz et al., 2004; Fernández Orquín, 2005) se hace esta proposición y es aquí que por primera vez se ve una propuesta de oficializar esta práctica;
- Introducción de una tarea morfológica;
- Tienen en cuenta la vaguedad y la indeterminación;
- Permiten la elección de un subconjunto de categorías semánticas;
- Definición de una serie de medidas para reflejar distinciones sutiles, como superposición parcial, sobre generación y asignación distintiva;
- Proporcionan meta-información asociada con los textos, para permitir la investigación sobre género y variedad en portugués;
- Una arquitectura modular completamente documentado, cuyo código fuente está disponible bajo licencia GPL;
- Un primer estado del arte para el REN en portugués.

En el 2006, Toral y Muñoz hacen una propuesta muy interesante en (Toral y Muñoz, 2006), construir y mantener diccionarios de REN utilizando la enciclopedia libre Wikipedia. En este enfoque hacen uso de una jerarquía de sustantivos obtenidos de WordNet (Miller, Beckwith et al., 1990), además de la primera frase de un artículo para reconocer las entidades. Opcionalmente usan un POS-Tagger para obtener las categorías gramaticales, éstas se puede utilizar con el fin de mejorar la eficacia del algoritmo. Plantean que pueden alcanzar un alto nivel de independencia del lenguaje, lo que hace el método útil para idiomas con pocos recursos. Ellos informan que obtienen puntuaciones de F-medida de 78% y de 68% para las clases `Localidades` y `Persona` respectivamente.

Entre el 2007 y 2008 surgen otros trabajos que también utilizan la Wikipedia para el REN, ejemplo de ello son los artículos (Bhole, Fortuna et al., 2007; Watanabe, Asahara et al., 2007; Dakka y Cucerzan, 2008). Watanabe et al., utilizan campos aleatorios condicionales para la clasificación (Conditional Random Fields), obteniendo una F-medida de 79.8% para la clase personas, 72.7% para Localidad y 71.6% para organización. Bhole y otros investigadores, así como Dakka y Cucerzan utilizan aprendizaje supervisado con el clasificador SVM. La diferencia entre ellos radica en que el primero utiliza todo el

texto para construir el vector de características, obteniendo 72.6%, 70.5% y 41.6% de F-medida en las clases `Persona`, `Localidad` y `Organización` respectivamente. Dakka y Cucerzan, para construir el vector de características -además de todo el texto- utilizan el primer párrafo, el resumen los datos en infoboxes, más los hipertextos de enlaces entrantes con las palabras circundantes. Así, Obtienen 95% y 93% de F-medida para las clases `Persona` y `Localidad` respectivamente.

En el año 2009 se realiza el primer taller Named Entities Workshop nombrado NEWS 2009, se realiza en colaboración con ACL-IJCNLP 2009. El segundo taller, NEWS 2010, se lleva a cabo en julio del 2010, se celebra en la Universidad de Uppsala, Suecia. NEWS 2010 se realiza en colaboración con ACL 2010. En el 2011 tiene lugar el tercer taller de sobre REN (NEWS 2011), se realiza junto con la IJCNLP 2011 (The 5th International Joint Conference on Natural Language Processing). Recientemente en el año 2012 se celebra el cuarto taller Fourth Named Entities Workshop, NEWS 2012.

Estos eventos, aunque tratan el tema del REN, introducen una nueva y modificada tarea, la transliteración. La transliteración, como se presenta en (Li, Kumaran et al., 2009) es la conversión de un nombre propio en el idioma de origen (una cadena de texto en el sistema de escritura de origen) a un nombre en el idioma de destino (otra cadena de texto en el sistema de escritura de destino), de manera que el nombre del idioma de destino es: (i) fonéticamente equivalente al nombre de la fuente, (ii) se ajusta a la fonología de la lengua objetivo y (iii) existe coincidencia con la intuición de usuario de equivalencia entre el nombre en el idioma de origen con el idioma de destino, teniendo en cuenta la cultura y el uso de caracteres ortográficos en el idioma de destino.

Otro evento importante, desarrollado en el año 2012 es el COLING 2012, en esta conferencia se destaca el trabajo NEER: An Unsupervised Method for Named Entity Evolution Recognition (Gossen, Kanhabua et al., 2012). En él se propone un método no supervisado para el reconocimiento de la evolución de las entidades nombradas, independiente de fuentes conocimiento externas. Mediante el análisis de períodos de tiempo utilizando un método de corrimiento de ventanas de co-ocurrencia y capturan evolución en términos en el mismo contexto. Así evitan comparar términos de épocas muy diferentes y superar una grave limitación de los métodos existentes para la llamada evolución de entidad nombradas. Obtienen una cobertura de 90% en el corpus del New York Times. Usan aprendizaje automático con supervisión mínima, mejorando la precisión en un 94%.

También en el 2012, en Estados Unidos, específicamente en la ciudad de Washington, se celebra en junio la conferencia JCDL 2012 (Joint Conference on Digital Libraries) auspiciada por ACM/IEEE. Esta conferencia es el mayor fórum enfocado en bibliotecas digitales y técnicas asociadas. Aquí se presenta el trabajo: An Analysis of the Named Entity Recognition Problem in Digital Library Metadata (Freire, Borbinha et al., 2012). Este trabajo presenta los resultados de un estudio de cómo el problema del REN se

manifiesta en los metadatos de biblioteca digital. En particular, se presentan las principales diferencias entre la realización de REN en lenguaje natural y en el texto dentro de los metadatos. El documento finaliza con un nuevo enfoque para el REN en los metadatos.

La mayoría de los trabajos consultados, tanto los presentados en las MUCs como en las CoNLL y las demás conferencias, reportan resultados en el reconocimientos, solamente, de las entidades fuertes. Algunos son capaces de detectar entidades que están en los tres grandes grupos definidos para esta tarea en la MUC-7 en 1997: Name Extraction (ENAMEX), Time Extraction (TIMEX) y Number Extraction (NUMEX), el primero comprende entidades del tipo nombres propios, organizaciones y localidades. El segundo grupo dedicado a la extracción de expresiones de tiempo y el último en el que se reconocen expresiones numéricas que involucran valores monetarios y porcentajes (Tjong Kim Sang y De Meulder, 2003).

Aún cuando se ha investigado bastante en esta dirección, en realidad, se hace difícil definir qué es una entidad, pues muchos especialistas tratan este término de forma diferente. Donde sí parece existir consenso es en la definición de las entidades fuertes (pertenecientes a las ENAMEX), pero existen algunas discrepancias en las que comúnmente han sido nombradas como entidades débiles. Sobre este tema se abundará más adelante.

Como elemento curioso, se puede plantear que la técnica más utilizada en el CoNLL-2003 fue los Modelos de Probabilidad Condicional de Máxima Entropía (MPCME) (Bender, Och et al., 2003). Esta técnica de aprendizaje automático ha reportado buenos resultados en varias tareas del PLN, por ejemplo la desambigüación (Suárez, 2005), en la detección del límite de oraciones (Beltrán, 2007), etc.

Un elemento que frecuentemente aparece en estos eventos son los indicadores para medir la Precisión (en inglés Precision), expresado como la capacidad del sistema de no presentar entidades incorrectas. También, la Cobertura (en inglés Recall) es utilizada para expresar la capacidad del sistema de identificar el mayor número de entidades. Por otro lado, se ha utilizado la F-medida (en inglés F-measure), como la función que regula el balance entre precisión y cobertura a través del parámetro β. Para β=1, precisión y cobertura tienen la misma importancia (Troyano, 2004). En la mayoría de los trabajos consultados se toma β=1, para más detalles ver el epígrafe 2.5.2.

De todos los trabajos relacionados anteriormente, no se ha podido encontrar información sobre sistemas que realicen el reconocimiento de entidades sin pre-procesamiento de los corpus con que trabajan, pues en su gran mayoría utilizan alguna técnica de PLN como pre-procesamiento antes de iniciar el reconocimiento de entidades. La mayoría de los investigadores en esta temática realizan algún tratamiento

previo, entre los que con frecuencia se encuentran el análisis léxico-sintáctico, semántico y en mucha menor cuantía otros más profundos como análisis del discurso, etc.

A diferencia de los trabajos mencionados, esta investigación se ha propuesto prescindir de todo tipo de procesamiento previo del corpus a analizar. Esto evitaría en gran medida la pérdida de tiempo en estas tareas que resultan bastante consumidoras, pues conllevan una alta carga computacional. La interrogante entonces está en saber si se podrán lograr buenos resultados, o en cierta medida, similares a los que obtienen otros investigadores.

Por otra parte, tampoco se ve con mucha frecuencia en la literatura, anterior al 2004, una propuesta de evaluación separada de las fases que componen el proceso. No es hasta el 2005, un tiempo después que este investigador lo propusiera en (Fernández, Muñoz et al., 2004; Fernández, 2005) que en (Ferrández, Kozareva et al., 2005) se puede ver un trabajo de este tipo. No es hasta HAREM 2006, que se comienza a ver la importancia de dividir la evaluación de ambas fases. Téngase en cuenta que esto permitiría determinar cuán bueno o malo es el método utilizado en la identificación y en la clasificación, de forma independiente. Es necesario tener en cuenta que para la fase de identificación se utilizan muchos recursos que aún poseen márgenes de errores en su funcionamiento; haciendo que en ocasiones la clasificación falle debido a una mala identificación. Obtener una medida que evalúe por separado ambas fases, permitiría rectificar el método utilizado con mayor facilidad, detectando con claridad en qué fase se han producido los errores.

2.8.3.1 Sistemas para el REN basados en reglas

Entre los sistemas para el REN basados en reglas se encuentra:

FASTUS, un sistema desarrollado por el año 1993. Según (Appelt, Hobbs et al., 1993), los sistemas que basan su enfoque para el procesamiento de texto en el análisis con una gramática libre de contexto, tienden a ser lento y propensos a errores debido a la ambigüedad masiva de las frases largas. Por el contrario, FASTUS emplea un modelo de lenguaje de estados finitos no determinista que produce una descomposición de una frase en grupos nominales, grupos de verbos y partículas. Otra máquina de estado finito reconoce las frases específicas del dominio basados en combinaciones de los encabezados de los componentes que se encuentran en la primera pasada.

FASTUS utiliza reglas que son manualmente creadas, las cuales se dividen en reglas de grano grueso, independientes del dominio (macros), así como reglas dependientes del dominio, las que utilizan las macros para construir las reglas finales. Dichas reglas utilizan elementos morfo-léxicos, sintácticos y semánticos.

Este sistema fue evaluado en la MUC-4 mostrando un 55% de precisión y un 44% de cobertura los que constituyeron uno de los mejores resultados en esta evaluación (Sundheim, 1992). En la MUC-6 fue utilizado para la EI en dominios que abordaban el tema del terrorismo y textos militares. Los mayores errores de este sistema se producen en la fase de identificación, sobre todo en la detección de las entidades de tipo organización y localidades.

Otro sistema desarrollado unos años más tarde es **LaSIE**. Es un sistema constituido por nueve módulos que trabajan secuencialmente (Gaizauskas, Humphreys et al., 1995). El orden es: un tokenizador, la búsqueda en diccionarios, el segmentado de frases, el etiquetador morfosintáctico, POS-tagger, macheo de nombres, análisis del discurso y generador de plantillas. Este sistema fue presentado en la MUC-6. LaSIE utiliza diccionarios de entidades y disparadores, así como un gran grupo de reglas contenidas en diez gramáticas para la detección de entidades. Estos recursos son generados manualmente. Las reglas hacen uso de etiquetas morfosintácticas, semánticas y elementos léxicos. Las reglas fueron extraídas a partir del análisis del corpus Penn Tree Bank.

Este sistema fue capaz de detectar las entidades en los títulos de los documentos de la MUC-6 buscando entre las que ya habían sido extraídas del resto del documento. Alcanzo una precisión del 94% y una medida F-medida del 90.4%, con una cobertura del 87% en la competición del MUC-7 (Chinchor, 1999). Los resultados de este sistema fueron bastante bueno, sin embargo su cobertura se vio afectada debido a la incapacidad de identificar entidades que no estaban consideradas en sus diccionarios. La mayor cantidad de errores se producen por la mala clasificación de las entidades de tipo persona y organización.

En una segunda versión nombrada LaSIE II (Humphreys, Gaizauskas et al., 1998), presentada en la MUC-7, logran superar los errores de la anterior incrementando la cantidad de reglas gramaticales del sistema.

Otro sistema dentro de esta clasificación es FACILE. Realizado para acometer la tarea de REN en MUC-7. Como se puede ver en (Black, Rinaldi et al., 1997): Es basado en reglas, realiza análisis gramatical del contexto, las reglas usan operadores de iteración con patrones de congruencia y no se emplean técnicas basadas en el aprendizaje. Este sistema obtuvo resultados con una cobertura de 78% y una precisión de 87%. La categoría que más dificultades presentó fue la de organización, los autores alegan que esto se debe a la cantidad de dominios de los que deben depender las reglas y a una inadecuada base de datos con disparadores para identificar las entidades de tipo organización.

Teniendo en cuenta que este sistema constituyó un referente para esta investigación, es necesario mencionar a EXIT. Según se puede ver en (Llopis, 1988) es un sistema de EI en el dominio de las escrituras notariales de compra-venta. Las entidades que son objeto de extracción fundamentalmente son

las organizaciones y/o personas que intervienen en las operaciones de compra-venta, así como los inmuebles objetos de la transacción. Para realizar el trabajo los autores se basaron en:

- Una serie de disparadores (palabras específicas que introducen o forman parte de las entidades)
- Un conjunto de reglas específicas que determina la estructura de una entidad
- Heurísticas para la desambigüación.
- Arquitectura del Sistema EXIT.
- Tokenización y etiquetado (estos conceptos se explicarán en próximos capítulos).
- Tokenizador específico: Uso de diccionarios de nombres, apellidos y localidades.
- Reconocedor de entidades: Esta etapa o módulo se apoya en una gramática específica para identificar entidades o partes de entidades.
- Analizador: En esta fase se realiza un análisis sintáctico de las frases apoyándose en una gramática y en una ontología de rasgos, obteniéndose una serie de estructuras sintácticas.
- Resolución de correferencia.
- Interpretación semántica.
- Interpretación del discurso: se realiza una interpretación del discurso basándose en un modelo del mundo que se obtiene de una base de hechos y una serie de reglas de inferencia.

Ya en el año 2005, surge el sistema **DRAMNERI**. Un programa que utiliza un método basado en reglas y gazetteers o diccionarios para el reconocimiento de entidades. El sistema es adaptable a cualquier dominio en específico, además es multilingüe. Fue aplicado con éxito en las tareas de Extracción de Información y Búsqueda de Respuestas (Toral, 2005).

Su estructura se organiza como un conjunto secuencial de módulos con un alto grado de flexibilidad, lo que significa que algunos módulos se pueden usar o no dependiendo de la entrada. Además, la mayoría de las acciones que realiza, y los diccionarios y reglas que utiliza son configurables mediante el uso de archivos de parámetros. Los módulos principales del sistema son: Toquenizador, segmentador de frases, identificador de entidades, reconocedor de entidades. El reconocedor puede configurarse para que utilice o no las evidencias externas e internas.

Según su autor, "*los resultados obtenidos para la clase organización son muy bajos. Esto se debe principalmente a la alta interacción entre esta categoría y localidad y a la escasez de entidades del tipo organización*" (Toral, 2005).

2.8.3.2 Sistemas para el REN basados en aprendizaje automático

Dentro de la rama del reconocimiento de entidad con la utilización de ME existen en la actualidad un gran número de investigadores como Borthwick con su trabajo Description of MENE Named Entity System (Borthwick, Sterling et al., 1998) presentado en MUC-7. Estos investigadores proponen una teoría de utilización framework de ME y una arquitectura flexible basada en objetos. El sistema es capaz de utilizar un rango bastante diverso fuentes de conocimiento para la toma de decisión en el etiquetado. Estas variadas fuentes de conocimiento incluyen atributos tales como la capitalización de las palabras, atributos léxicos y otros que indican el tipo de texto (por ejemplo: titular o cuerpo principal del documento). También, hacen uso de grandes diccionarios de términos simples y multi-palabras, así como nombres de propios de personas, compañías, corporaciones, etc.

Este sistema, construido a partir de las fuentes de conocimiento, no contenía patrones generados manualmente y logró resultado comparable con los mejores sistemas estadísticos. Otros experimentos mostraron que cuando al combinarlo con sistemas de codificación manual fue capaz de generar resultados que superaron las calificaciones más altas reportadas por los sistemas en la evaluación del MUC-7. Este sistema fue capaz de alcanzar una F-medida de 88.80.

También están los trabajos de James R. Curran y Stephen Clark de la Escuela de Informática de la Universidad de Edimburgo con su sistema Language Independent NER using a Maximum Entropy Tagger (Curran y Clark, 2003), presentado en CoNLL-2003. Este trabajo demostró que los modelos de máxima entropía pueden identificar efectivamente las entidades en un texto gran exactitud. Esta investigación se enfocó no solo en el inglés, sino también en el alemán y el holandés. En este trabajo se demuestra que los modelos de ME son efectivos cuando se adicionan atributos diversos y superpuesto. A continuación se muestran los resultados obtenidos para las pruebas con datos en inglés. Obtienen valores de F-Medida de 87.66%, 76.27%, 79,51%, 91.21%, para las entidades localidad, misceláneas, organización y persona respectivamente para la prueba de inglés.

Otro ejemplo lo constituye Bender, con su sistema Maximum Entropy Models for Named Entity Recognition (Bender, Och et al., 2003). En este trabajo los investigadores parten de un corpus anotado y un conjunto de atributos o características obtenidas de algún lenguaje. Ellos parten de un sistema base para reconocer las entidades y la información del contexto de datos adicionales no anotados. Finalmente, la lista de entidades obtenidas es incorporada en un reconocedor para con esto aumentar la exactitud en el reconocimiento. Finalmente demuestran que utilizar ME y adicionar información no anotada puede mejorar la exactitud de un reconocedor de entidades. Obtienen valores de F-Medida de 88.09%, 75.70%, 78.16% y 88.81%, para las entidades localidad, misceláneas, organización y persona respectivamente para la prueba del inglés.

Desarrollado por Hai Leong Chieu[44] y Hwee Tou Ng[45], está el sistema Named Entity Recognition with a Maximum Entropy Approach using global information (Chieu y Ng, 2002). Este trabajo difiere de los anteriores en el hecho de que usa información de todo el documento para clasificar cada palabra, con solo un clasificador, ya que los anteriores trabajos que usa esta técnica utilizan un clasificador secundario, el que corrige los errores del primero. Ellos demuestran que los modelos de ME son capaces de usar información global directamente y obtener resultados comparables a los mejores sistemas basados en aprendizaje que le precedieron en las MUC-6 y MUC-7. Obtienen valores de F-Medida de 88.79%, 79.28%, 82.30% y 91.67%, para las entidades localidad, misceláneas, organización y persona respectivamente para la prueba del inglés.

Utilizando otras técnicas de aprendizaje, surge Memory-Based Named Entity Recognition using Unannotated Data. En este trabajo se utiliza TiMBL (Daelemans, Zavrel et al., 2004) para encontrar nombres en inglés y alemán en texto de periodísticos. El primer sistema utiliza solo los datos de entrenamiento y un número de diccionarios. Los resultados muestran que los diccionarios no son beneficiosos en el caso del inglés, mientras que sí lo son para los datos en alemán. Se aplicó la generalización de tipo simbólico, pero también reduce el rendimiento. El segundo sistema utiliza diccionarios derivados de corpus no anotados, así como la relación de uso de capitalización frente a no capitalización de cada palabra. Estas estrategias le dieron un aumento en el rendimiento. Al igual que esta investigación, éste uno de los pocos trabajos encontrado que se desarrolla utilizando corpus no anotados.

Otro sistema que usa varios clasificadores, entre ellos ME, es NERUA (Ferrández, Kozareva et al., 2005): Sistema de reconocimiento de entidades para el idioma español combinando diferentes algoritmos de aprendizaje Se propone una detección de entidades independiente del lenguaje y se estudia la influencia del tamaño del corpus de entrenamiento en los resultados. Es de destacar que es uno de los pocos trabajos que evalúan, por separado, las fases de identificación y clasificación.

Este sistema aborda la tarea de reconocimiento de entidades en varios módulos. Los módulos más importantes son: el módulo NED encargado de la detección de las entidades y el módulo que realiza la clasificación (NEC). Estos módulos necesitan alimentarse de las características obtenidas del texto, dicha labor la realizan los módulos MEC para la detección y para la clasificación. Como salida el sistema produce los textos de entrada anotados con las entidades detectadas y clasificadas. El módulo de detección de entidades utiliza dos modelos de aprendizaje automático, concretamente Memory-based learner y Hidden Markov Model, mientras que para la implementación del módulo de clasificación se emplea, además de los modelos anteriores, un modelo basado en el principio de Máxima Entropía.

[44]Defense Science Organization National Laboratories of Singapore.

[45] Department of Computer Science National University of Singapore.

En ambos casos, detección y clasificación de entidades, se desarrolla a través de una estrategia de voting (en español votación) que permite aumentar los resultados mediante una cooperación adecuada de los clasificadores. En la detección, al disponer de solo dos clasificadores, el voting se realiza mediante combinaciones de estos con diferentes conjuntos de características. NERUA obtuvo 92.96% de F-medida en la detección y 78.59% en la clasificación de entidades. Todas las pruebas, experimentos y resultados obtenidos, utilizan los recursos proporcionados para el español en CoNLL-2002 (Tjong Kim Sang, 2002).

El sistema obtiene una cobertura alta, si se tiene en cuenta que logra un resultado importante en la fase de detección. Sin embargo, la precisión es bastante modesta a pesar de usar tres modelos de aprendizaje, pues obtiene un discreto 78.59% en de F-medida en la fase de clasificación. No se hace referencia a esto, pero tanto la etapa de entrenamiento como la de ejecución deben consumir gran tiempo computacional, si te tiene en cuenta que usa más de un modelo de aprendizaje para su funcionamiento.

Otro sistema desarrollado en el 2005 es: Towards large-scale, open-domain and ontology-based named entity classification. En este trabajo se aborda la clasificación de entidades nombradas en relación con grandes conjuntos de clases que se especifican en una ontología dada (Cimiano y Völker, 2005). Este enfoque es no supervisado, ya que se no apoya en ningún corpus de entrenamiento etiquetado. Es considerado de dominio abierto, pues la ontología se podría intercambiar.

El enfoque se basa en la hipótesis de distribución de Harris (Harris, 1954), la que se sustenta en el modelo de espacio vectorial; una entidad nombrada se asigna al concepto contextualmente más similar a partir de la ontología. La principal contribución de este trabajo es el análisis sistemático del impacto de la variación de ciertos parámetros en un enfoque basado en el contexto, explotando las similitudes en un espacio vectorial para la desambigüación de entidades nombradas.

Aunque sería difícil de clasificar en algunos de los grupos antes mencionados, pues utiliza diversas técnicas y algo más general que los ejemplos anteriores, es necesario también mencionar en este punto el proyecto GALE, pues es el más grande, en el que incluyen varios elementos del PLN, entre ellos necesariamente el REN.

GALE: Global Autonomous Language Exploitation. Implementado con el objetivo del DARPA GALE PROGRAM de desarrollar y aplicar la tecnología de la computación para absorber, analizar e interpretar grandes volúmenes de discursos y textos en múltiples lenguajes. Los motores de auto-procesamiento automático convertirá y destilara los datos, deliberándolos pertinentemente, información consolidada en un formato fácil de entender por los militares y analistas monolingües de habla Inglesa en respuesta a las peticiones directas o implícitas.

GALE constara de tres motores principales: Transcripción, Traducción y Destilación. La salida de cada motor es un texto en inglés. La entrada al motor de transcripción es un discurso y al motor de traducción es un texto. Los motores pasarán los punteros a datos en el idioma de origen pertinentes que estarán disponibles a los humanos y los procesos subsiguientes. Los motores de destilación integran la información de interés a sus usuarios desde múltiples fuentes y documentos. El personal militar podrá interactuar con el motor de destilación vía una interface que podría incluir varios formularios de diálogos hombre-máquina (no necesariamente en lenguaje natural).

El Linguistic Data Consortium soporta el programa para el proyecto GALE, facilitando recursos lingüísticos, datos, tienen en común la mayoría de los sistemas de REN que se han realizado hasta la fecha, es el hecho de que constan de dos fases bien definidas, la fase de identificación de las entidades y la fase de la clasificación de las entidades. Aunque en algunos casos, como en este trabajo, puede herramienta para anotaciones y estándares de buenas prácticas para el entrenamiento, desarrollo y evaluación de sistemas.

Otro trabajo interesante es el titulado: Towards large-scale, open-domain and ontology-based named entity classification (Cimiano y Völker, 2005). Aquí se aborda la clasificación de entidades nombradas en relación con grandes conjuntos de clases que se especifican en una ontología dada. Este enfoque es no supervisado, ya que se no apoya en ningún corpus de entrenamiento etiquetado. Es considerado de dominio abierto, pues la ontología se podría intercambiar. El enfoque se basa en la hipótesis de distribución de Harris (Harris, 1954), la que se sustenta en el modelo de espacio vectorial; una entidad nombrada se asigna al concepto contextualmente más similar a partir de la ontología. La principal contribución de este trabajo es el análisis sistemático del impacto de la variación de ciertos parámetros en un enfoque basado en el contexto, explotando las similitudes en un espacio vectorial para la desambigüación de entidades nombradas.

Cambiando a otro tema, es conocida por los investigadores en el área del PLN la importancia que han cobrado los llamados stemmers. Son ampliamente utilizados en la Recuperación de Información, así como en otras tareas. En el próximo epígrafe se abundará en esta temática, ya que constituye un objetivo de esta investigación.

2.9 El Stemming

El Stemming, traducido en muchos trabajos en español como estemizado (en lo adelante se usará el término en inglés), es un método para reducir una palabra a su raíz (stem). A los sistemas que realizan esta tarea se les conoce como stemmers (estemizadores). En la mayoría de los casos, las variantes morfológicas de un término puede converger a un único representante de forma y la interpretación semántica puede ser considerada como equivalente a los efectos de las solicitudes de los recuperadores de información.

Varias aplicaciones de PLN miden la similitud o las distancias de edición entre palabras basadas en la morfología. Sabe que esto es debido a que las palabras están compuestas de morfemas y lexemas. Se vio en la explicación de epígrafe 2.5.1 que el lexema o raíz es la parte fija que aporta el significado principal y los morfemas o afijos que lo modifican, aportando información complementaria. La contribución puede ser léxica, expresada a través del sufijo, sintáctica o morfológica, expresada a través de la desinencia[46]. Por ejemplo, en el idioma inglés, run, runs, ran y running son formas del mismo lexema, convencionalmente escrito como RUN.

Otro concepto muy utilizado en el PLN es el lema, no es exactamente un lexema, es una forma particular de este; que es elegido por convención para representar una forma canónica de un lexema. Por ejemplo, para el caso de los verbos sería su forma simple, el infinitivo.

Los lexemas constituyen un punto clave de la presente investigación, representan el segmento inmutable de la palabra, forman la mayor parte del léxico de una lengua y su número es siempre superior al de los gramemas. El lexema de una palabra suele llamársele la raíz o tallo por el que se comienza la construcción de su familia. En la lingüística, un tallo es la parte de una palabra que es común para todas sus variantes conjugadas. Un tallo es a menudo raíz, pero no siempre tiene que coincidir con ella. Un ejemplo donde se aprecia este proceder, es en la familia de la palabra `gato`, cuyo lexema es `gat`. La familia puede estar compuesta por:

`gat-o, gat-a, gat-os(plural masculino), gat-as, gat-os(neutro), gat-icos, gat-icas`

Teniendo en cuenta estos aspectos, la meta del Stemming es reducir a la formas inflexional y a veces las formas derivativas relacionadas de una palabra a una base derivativa común. Una aproximación muy

[46]Se denomina desinencia a cualquier morfema flexivo. Parte variable de la terminación de las palabras (por la oposición a la raíz), tiene una función gramatical o léxica. En español, como en otras lenguas, las desinencias sirven para conjugar un verbo expresando los diferentes tiempos, personas y modos añadiéndose tras la raíz o lexema.

popular entre los stemmers es remover los afijos de la palabra analizada, así la reducen a un supuesto stem, si esto se hace correctamente, todas las formas derivadas de la palabra serán aglutinadas en una forma estándar.

No obstante, el beneficio de los stemmers es evidente al darle tratamiento a las diferentes variantes morfológicas e indexar solo el stem de las palabras. De esta forma los usuarios de un buscador no están obligados a incluir varias formas de la misma palabra en una pregunta. Por ejemplo, las palabras del inglés -antes mencionadas- run, runs, ran y running, pueden ser manipuladas en un solo stem, RUN.

De esta forma, una consulta que incluya los términos run o runs retornará los mismos resultados. Además, debido a que los términos indexados son almacenados en un lexicón, aplicar Stemming a un índice significa que serán almacenadas las raíces lingüísticas en lugar de toda la palabra, reduciendo así el tamaño del índice.

Como se menciona en párrafos anteriores, el Stemming es ampliamente utilizado para mejorar la eficiencia de los sistemas de Recuperación de Información. Estos comienzan a ser estudiados en la década del 60, con el propósito de reducir los índices. También como una forma de normalizar los términos. Otra manera de verlos es como una forma de expandir la pregunta, uniendo las formas flexivas o derivativas de las palabras.

Como es sabido, los lenguajes son altamente irregulares, por consiguiente los métodos de Stemming desarrollados hasta el momento no son cien por ciento exactos. En la mayoría de los casos sucede que palabras semánticamente distintas se agrupan bajo el mismo stem. Por ejemplo: campo, campamento, campesino, son agrupados bajo el stem camp; este hecho permitiría que la palabra campana, semánticamente distante de las anteriores, se agrupe baja el mismo stem.

Existen muchos criterios para intentar evaluar los resultados del Stemming:

- Exactitud (en inglés el término más usado es correctness). La exactitud es medida a través de los valores de overstemming y understemming. (estos términos serán explicados más adelante).
- Efectividad en la recuperación: se mide su funcionamiento los sistemas de RI, a través de cobertura y precisión, su velocidad, etc.
- Compresión: se mide a través de la capacidad del stemmer de comprimir el tamaño de los indexados.

Con relación a esto, en (Paice, 1990) se describen los dos términos asociados al error del cálculo de los límites del stem, son los términos mencionados anteriormente: overstemming y understemming, que en español sería algo como sobre-esteimizado y sub-estemizado.

En ambos casos estos errores afectarían el buen funcionamiento de los sistemas de RI. Un error del tipo overstemming implicaría que se está reduciendo demasiado el tamaño del stem, lo que provocaría que se agrupen mayor cantidad de palabras bajo este concepto. Esto provocaría, en términos de un sistema de RI, que sean recuperados documentos no relevantes.

Por el contrario, un error del tipo understemming sería la inclusión de más letras de las que debe tener en realidad el stem, por lo que se agruparían menos cantidad de palabras bajo este concepto. Esto provocaría que el sistema de RI que use este tipo de stem en si índice, que no sean recuperados los documentos que son relevante. El understemming puede disminuir la cobertura (recall) en los sistemas de recuperación de información.

Según la clasificación hecha en (Frakes, 1992) (ver Figura 2.11), aunque han existido muchas otras, lo stemmers se dividen según el método de agrupamiento en:

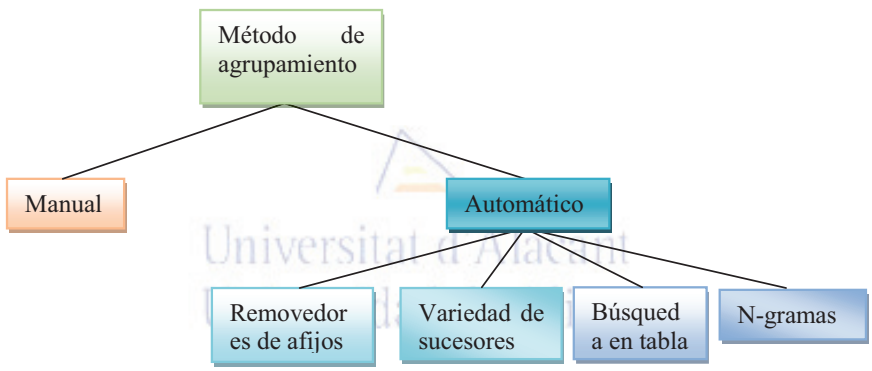


Figura 2.11. Topología de los stemmers

No se comentará nada sobre el agrupamiento manual porque el proceder se desprende de su propio nombre, además es un método que en la actualidad solo se utiliza para las comparaciones con los métodos automáticos.

Por su parte, los removedores de afijos, como su nombre lo indica eliminan sufijo y/o prefijos para lograr en stem. Generalmente basan su trabajo en un grupo de reglas que se aplican según el lenguaje. Queda claro que este método es dependiente del lenguaje y necesita de gran participación de los humanos para poder escribir las reglas.

En otro extremo, los que utilizan el método de los N-gramas, agrupan los términos basados en el número de N-gramas que comparten. Los términos y sus stem correspondientes son almacenados en una tabla. Luego el proceso de Stemming se realiza mediante una búsqueda en dicha tabla. Una vez que encuentran los N-gramas que no se repiten en la comparación de dos palabras, aplican una medida de similitud entre

estas palabras (el coeficiente de Dice es frecuentemente utilizado para este objetivo (Dice, 1945)). Luego se crea una matriz de similitud con estos valores. Los términos son agrupados utilizando el método de agrupación de un solo enlace (Frakes, 1992).

Así también, los que usan el método de búsqueda en tabla, almacenan todos los términos indexándolos con su respectivo stem, luego pueden ser extraídos usando un B-tree o tablas hash. Estos métodos suelen ser bastante rápidos.

Se ha dejado para el final, no porque sea el peor o menos importante, sino porque guarda cierta relación con el método utilizado en este trabajo y será descrito con más detalles. Aquí se recogen los que usan la variedad de sucesores para el agrupamiento (Hafer y Weiss, 1974). Este método se basa en el cálculo de la frecuencia de la secuencia de caracteres en un grupo de palabras.

En este sentido, Hafer y Weiss formalizaron esta técnica de la forma siguiente:

> Sea α una palabra de longitud n; α_i es el prefijo de longitud i de α. Sea D un corpus de palabras. $D_{\alpha i}$ es definido como el subconjunto de D que contiene los términos los cuales la primera letra i coincide exactamente con α_i. La variación de sucesor de α_i, denotada como $S_{\alpha i}$, es entonces definido como el número de letras distintas que ocupan la posición $i+1^{era}$ posición de las palabras en $D_{\alpha i}$. Una palabra de muestra de longitud n tiene n variaciones de sucesores $S_{\alpha 1}$, $S_{\alpha 2}$, ..., $S_{\alpha n}$

La variación de sucesor de una cadena es el número de caracteres diferentes que la siguen en un cuerpo de texto. Por ejemplo, supóngase el conjunto de textos siguiente: `educador, enseñar, elevador, escaso, educando, edición, educa, educar, eduqué, educable.`

Entonces, para determinar la variación de sucesor para la palabra `educable`, se puede seguir el siguiente procedimiento. La primera letra de `educable` es la `e`. La `e` está seguida, en el conjunto de texto anterior, por cuatro caracteres: `d`, `n`, `l` y la `s`. De esta forma la variación de sucesor de la letra `e` es cuatro. La siguiente variación de sucesor para `educable` es dos, debido a que las letra `u` y la `i` están seguidas del bi-grama `ed` en el conjunto de textos. Este proceso se repite variando todos los caracteres de la palabra `educable`. A continuación se muestra en la Tabla 2.4 con el proceso completo:

Tabla 2.4. Proceso de Stemming con variedad de sucesor

Prefijo	Variación de sucesor	Letras	$I D_{\alpha i} I$
E	4	D, N, L, S	10
ED	2	U, I	7
EDU	1	C	6
EDUC	1	A	5
EDUCA	3	D, N, R	5
EDUCAB	1	l	1
EDUCABL	1	E	1
EDUCABLE	1	BLANCO	1

Para decidir cuál será el prefijo que se seleccionará existen varios métodos:

1. Usar el método de corte (en inglés cutoff), se selecciona un valor de variación de sucesor y los límites del prefijo se obtiene cuando se alcanza ese valor.
2. El método de peak y plateau, aquí se obtiene el punto de corte cuando se encuentra un valor de variación de sucesor que excede tanto a su predecesor como a su sucesor.
3. El método de la palabra completa, el punto de ruptura se obtiene cuando algún prefijo sea igual a una palabra del corpus.
4. El método de la entropía, este método saca partido de la distribución de variación de letras sucesora. Sea $I D_{\alpha i} I$ el número de palabras en un conjunto de texto que comienza con una longitud de secuencia i de letras α. Sea $I D_{\alpha i j} I$ el número de palabras en $D_{\alpha i}$ con sucesor j. La probabilidad de que un miembro de $D_{\alpha i}$ tenga el sucesor j es dado por $\frac{|D_{\alpha i j}|}{|D_{\alpha i}|}$. La entropía de $I D_{\alpha i} I$ es:

$$H_{\alpha i} = \sum_{p=1}^{n} -\frac{|D_{\alpha i j}|}{|D_{\alpha i}|} \cdot \log_2 \frac{|D_{\alpha i j}|}{|D_{\alpha i}|} \qquad 2.22$$

Mediante esta ecuación, se puede obtener un valor de entropía para un prefijo. De igual forma, para los sucesores también podrían obtenerse esta medidas. Se selecciona un punto de corte y los límites de los prefijos serían identificados una vez que se alcanzara una entropía significativamente mayor a la del predecesor inmediato.

En el ejemplo en la, tanto aplicando en método (2) (peak y plateau) o el (3) se puede ver fácilmente que se obtiene el prefijo EDUCA como stem de la palabra EDUCABLE.

Como bien se plantea en (Hafer y Weiss, 1974), ninguno de estos métodos es la perfección, Hafer y Weiss expresan que la combinación de algunas de estas técnica funcionan bastante bien. Sin embargo, más tarde se hace un estudio en (Al-Shalabi, Kannan et al., 2005), en el que se llega a la conclusión que el

algoritmo basado en variedad de sucesores con punto de corte, funciona mejor que el basado en la entropía. Los autores plantean que logran una diferencia de un 15% entre ambos métodos.

No obstante, a pesar de que en la mayoría de la literatura consultad se ofrecen buenos criterios sobre el método de variedad de sucesor, es bueno destacar que la técnica tiene también sus puntos débiles. Por ejemplo:

Supóngase que se tiene el siguiente conjunto de palabras del inglés: box, boxer, bottle, both, bring, barn, bag. Y se desea obtener la variedad de sucesor para la palabra boxer, se tendría:

Tabla 2.5. Variedad de sucesor para la palabra boxer.

Prefijo	Variedad de sucesor	letras
B	3	A, R, O
BO	2	T, R
BOX	1	E
BOXE	1	R
BOXER	1	blanco

Si se selecciona el método de punto de corte con valor cuatro, el stem sería boxe. Con el método de peak y plateau no se podría seleccionar, pues la variedad de sucesor monótonamente decrece. Aplicando el método de palabra completa se elegiría el prefijo box. En este ejemplo tampoco habría suficientes valores para aplicar el método de la entropía.

Como se puede apreciar, aplicando métodos diferentes se obtienen resultados diferentes para el mismo procedimiento de variedad de sucesor.

Analizando otros elementos, es necesario aclarar que la medición de la eficiencia de los métodos de stemming, con frecuencia ha sido evaluarlos por su efecto en los sistemas de RI (Smirnov, 2008). Esta idea de evaluarlos a partir de una aplicación concreta, puede ser que permita decidir cuál de un grupo de stemmer es mejor que otro. Sin embargo, esta idea no permite determinar dónde, ni qué elementos del stemmer están fallando o que partes del método se podrían mejorar. En este sentido, a sido creada otra forma de evaluación, a continuación se hace una breve explicación de este nuevo método de evaluación propuesto para comparar los stemmers.

Método de evaluación de Paice

El método de evaluación fue presentado por Chris D. Paice en el artículo (Paice, 1990). Aquí el autor persigue la evaluación de un algoritmo mediante la contrastación de los valores de understemming y overstemming. Para ello se definen los cuatro parámetros siguientes:

- DMT (Desired Merge Total): Agrupado total deseado, cuantifica la cantidad total de familias en las que se dividió el cuerpo original.

- DNT (Desired Non-Merge Total): Agrupado total no deseado, contiene la cantidad de palabras mal agrupadas.
- UMT (Unachieved Merge Total): Total de palabras no alcanzadas, palabras no procesadas por el algoritmo debido a insuficiencia de reglas u otros errores.
- WMT (Wrongly Merge Total): Total de palabras mal agrupadas por el algoritmo, descartados errores ortográficos, debido a insuficiencia de reglas.

Para cada familia de palabra, entiéndase por familia un grupo g que comparte un lexema similar. El DMT es equivalente a la cantidad de palabras agrupadas en dicha familia, que comparten exactamente el mismo lexema.

De la misma forma, el DNT representa a las palabras mal agrupadas, pues su lexema está compuesto por el general de la familia y alguna partícula más que lo hace diferente. Esta situación puede ser expresada por las siguientes definiciones, donde n_g representa el número de palabras por subfamilia y W es el total de palabras en la familia.

$$DMT_g = \frac{1}{2} n_g (n_g - 1) \tag{2.23}$$

$$DNT_g = \frac{1}{2} n_g (W - n_g) \tag{2.24}$$

Para cada familia, el UMT puede ser calculado contando las posibles ubicaciones dentro de sus subfamilias, donde pueden ser colocadas las palabras mal ubicadas o no procesadas.

Supóngase que un grupo de tamaño n_g contiene s distintos lexemas, representando a cada subfamilia luego de procesados; el número de instancias de los lexemas es $u_1, u_2, \dots, u_n$ respectivamente.

El número de errores por understemming en el grupo está dado por:

$$UMT_g = \frac{1}{2} \sum_{i=1}^{s} u_i (n_g - u_i) \tag{2.25}$$

Se necesita una operación adicional para calcular el WMT. Se crea un grupo paralelo mediante la agrupación de todas las palabras que compartan idénticos lexemas y es etiquetado con un número, según el orden de a parición en el grupo original.

Luego, se puede calcular el WMT comparando los números del nuevo grupo con los del original. Si el nuevo grupo contiene lexemas con números idénticos, no se cometieron errores de overstemming y el valor de WMT para ese grupo es cero.

De lo contrario, si en el nuevo grupo aparecen lexemas con diferentes etiquetas numéricas, significa que existen palabras semánticamente diferentes agrupadas en el grupo original.

Supóngase una familia de tamaño n_s que contiene t diferentes etiquetas numéricas, lo que significa que las palabras provienen de t diferentes grupos iniciales. Las etiquetas de cada grupo original están representadas por $v_1, v_2, \ldots, v_t$. El número de errores de overstemming o WMT para este grupo se calcula por:

$$WMT_s = \frac{1}{2}\sum_{i=1}^{t} v_i\,(n_s - v_i) \qquad 2.26$$

Los valores globales de DMT, DNT, UMT y WMT son simplemente sumatorias de los resultados por grupos expresados como GDMT, GDNT, GUMT y GWMT. Los valores de GDMT y GDNT son utilizados para normalizar los valores de errores de understemming y overstemming a una fracción:

$$UI \text{ (Under-Stemming index)} = GUMT / GDMT \qquad 2.27$$

$$OI \text{ (Over-Stemming index)} = GWMT / GDNT \qquad 2.28$$

Finalmente, se define una última variable, Stemming Weight (SW) o peso del algoritmo, que brinda una idea de la solidez del procedimiento, calculada a través de los porcentajes de sus errores.

$$SW \text{ (Stemming Weight)} = OI / UI \qquad 2.29$$

Una vez vistos los referentes teóricos sobre los métodos de stemming, en el próximo epígrafe se relacionan los trabajos vinculados a esta temática, con el objetivo de ganar en claridad en el estado de esta cuestión.

Una medida que trata de obtener el rango de error relativo al truncamiento es la llamada *ERRT*, esta medida puede ser obtenida extendiendo una línea desde el origen O a través del punto (*UI*, *OI*) hasta que se intersecte con la línea T con se ilustra en la .De esta forma *ERRT* se define como:

$$ERRT = \text{longitud (OP)} / \text{longitud (OT)} \qquad 2.30$$

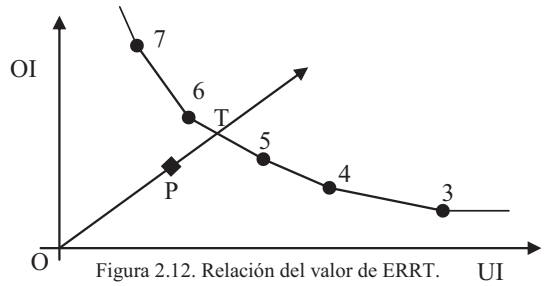


Figura 2.12. Relación del valor de ERRT.

2.9.1 Trabajos relacionados en la temática del Stemming

El Stemming ha sido ampliamente utilizado para mejorar la eficiencia de los sistemas de RI. Comenzó a ser estudiado en la década del 60, con el objetivo de acortar los indexados, así como también como una forma de normalizar los términos a indexar.

Es también visto por muchos como una forma de expansión de la pregunta en los sistemas de RI, adicionando formas flexivas o derivadas de las palabras. Los stemmers pueden ser lingüísticos (removedores de afijos), automáticos (estadísticos) o mixtos. A juicio de este autor, tal vez fue Julie Beth Lovins el creador en 1968 del primer algoritmo para el Stemming lingüístico (Lovins, 1968).

Lovins usó un solo algoritmo y varias listas de excepciones para remover diferentes terminaciones de las palabras, removiendo una cada vez usando algunas tablas de sufijos y reglas. Posteriormente ya en 1974 existieron otros stemmers lingüísticos como el descrito en (Dawson, 1974).

Uno de los stemmers más usado para el inglés es el stemmer de Porter, fue presentado por primera vez en 1980 por Martin Porter en la Universidad de Cambridge (Porter, 1980). Ha sido utilizado, fundamentalmente, como parte del proceso de normalización que comúnmente se realiza en los sistemas de Recuperación de Información. Se le considera dependiente del lenguaje.

Este algoritmo remueve alrededor de 60 finales de palabras en cinco pasos, para su funcionamiento necesita una completa definición de los sufijos del lenguaje, que luego serán truncados. Este algoritmo fue desarrollado utilizando el framework Snowball[47], un grupo de desarrollo dirigido por M.F. Porter, que ha presentado un lenguaje de programación del mismo nombre (Porter, 1980).

Al respecto se puede plantear que, el lenguaje es simple, pero riguroso y permite expresar de manera natural las reglas para los algoritmos de Stemming. Puede ser aprendido por programadores con experiencia en una o dos horas.

[47] http://snowball.sourceforge.net

Por otro lado, las reglas para el Stemming definidas en Snowball son traducidas por un compilador brindado por el mismo lenguaje a un equivalente en ANSI C o en Java, ambos lenguajes de programación. Finalmente provee al usuario de un vocabulario de palabras acompañadas de su stem o lexema dependiendo del algoritmo de Stemming implementado.

Debido al aumento de la necesidad de recuperación de información, los algoritmos de Stemming han adquirido gran utilidad, a pesar de que necesiten diccionarios para trabajar, y de su frecuente malinterpretación. De ahí y con la idea de estandarizar el Stemming, surge el lenguaje Snowball, ideado expresamente para este tipo de algoritmos. La aproximación de Porter ha sido extendida a otros lenguajes. La principal deficiencia de los stemmers lingüísticos es su naturaleza dependiente del lenguaje.

Luego en 1990, surge el stemmer de Paice-Husk, otro algoritmo basado en reglas (Paice, 1990). Cada regla era agrupada en secciones correspondientes a la letra final del sufijo de la palabra analizada.

Más tarde, en el 2008 fue creado otro considerado una mejora a la variante de Lovins, descrito en (Smirnov, 2008).

Antes de Porter, en la década del 70 -precisamente en 1974- fue publicado el trabajo titulado Word segmentation by letter successor varieties (en español, segmentación de palabras por variación de letras sucesoras) (Hafer y Weiss, 1974). Éste se basó en trabajos sobre lingüística estructural. En él los autores describen un método para separar automáticamente las palabras en sus tallos y afijos. El proceso utiliza ciertas propiedades estadísticas de un corpus (conteo de variedad de letras sucesoras y predecesoras) para indicar que las palabras deben dividirse. En consecuencia, este proceso es menos dependiente de la intervención humana que otros métodos de Stemming más tradicionales. A juicio de este auto este es el primer trabajo que trata la temática de successor variety aplicada al Stemming.

En este trabajo, el sistema de segmentación se utiliza para construir los diccionarios de stem para la clasificación de documentos. Luego, realizan experimentos de RI con los documentos y las consultas así clasificadas. Los resultados muestran no sólo que este método es capaz de obtener una alta calidad de la segmentación de palabra, sino también que su uso en la RI produce resultados que son al menos tan buenos como los obtenidos usando los procesos más tradicionales.

Según lo antes expuesto, el principal problema con este algoritmo es el método de corte utilizado para encontrar la variación de sucesores y los límites.

Ya en 1993, Krovestz usa la frecuencia de los finales derivacionales en el idioma inglés como base para su stemmer. Este tipo de stemmer es considerado mixto.

Como elemento importante, se puede mencionar que los stemmers estadísticos han venido a mejor las deficiencias de los stemmers lingüísticos (dependientes del lenguaje). En los últimos años se han desarrollado varios stemmers, muchos de ellos basados en propiedades estadísticas extraídas de corpus.

En el 2005, es publicado el artículo: Experiment with successor variety algorithm using the cutoff and Entropy Methods (Al-Shalabi, Kannan et al., 2005), en este trabajo se presenta el desarrollo de un sistema de Stemming para el Árabe, vasado en la técnica de variedad de sucesores. Utiliza como técnica para determinar el stem el método punto de corte, el que comparan con el método de la entropía. Estos autores reportan que obtienen mejores resultados con el método de corte, obteniendo una diferencias de un 15%. Su stemmer funciona correctamente solo para un 80% cuando usan el método de corte y un discreto 75% para el método de la entropía.

Unos años después, en 2007, Benno publica su trabajo titulado: Putting successor variety Stemming to work, en español sería: poniendo a los estemizadores de variedad de sucesores a trabajar (Stein y Potthast, 2007). Este trabajo, aunque no se reconoce así en su artículo, retoma las ideas expuestas en (Hafer y Weiss, 1974). Este trabajo desarrolla un stemmer estadístico basado en la distribución de los afijos de las palabras en una colección de documentos. Aunque no se puede ver claramente en este artículo los resultados de haber utilizado el método de la entropía para seleccionar la variación de sucesores y los límites del stem. Este trabajo hace un aporte interesante a modificar el método de peak y plateau.

Intentando clasificar los stemmers, Smirnov menciona en (Smirnov, 2008) que -entre otros- existen: stemmers basados en corpus (Jinxi y Bruce, 1998), stemmers sensitivos al contexto (Peng, Ahmed et al., 2007) y stemmers basados en N-gramas (James y Paul, 2003). Estos no necesitan conocimiento lingüístico previo, en su lugar usan un entrenamiento supervisado. Muchos de estos trabajos han reportado mejoras sustanciales a la problemática del Stemming.

Desafortunadamente, a pesar del reciente incremento del estudio del Stemming para lenguajes diferentes del inglés (Popovic y Willet, 1992), (Braschler y Schäuble, 2000), (Kraaij y Pohlmann, 1996), (Moulinier, McCulloh et al., 2001), (Savoy, 1999), (Sheridan y Ballerini, 1996), sin embargo es bien conocido que la mayoría de los stemmers han sido realizados principalmente para el idioma inglés. Así que se podría decir que, hay una potencial área de investigación para otros lenguajes.

Se coincide totalmente con Smirnov cuando plantea en (Smirnov, 2008): "…*so the question of finding* "*the best" conflation algorithm is still open, as well as the problem of finding the best* "*tuning*" *parameters for existing algorithm",* que traducido al español sería: así que la pregunta de encontrar el mejor algoritmo está aún abierta, así como el problema de encontrar el mejor ajuste de parámetros para los algoritmos existentes.

Existen algunos trabajos (Goldsmith, 2001), (Wilbur, 2007), (Kazarov y Manandhar, 2001), (Hammarström, 2007), que tienen puntos en común con esta investigación. También otros como los que utilizan la técnica de variación de sucesores, son muy cercanos al espíritu y el objetivo de este trabajo, pero difieren en algunos aspectos.

Con respecto a los anteriores, la primera diferencia radica en la forma en que se obtienen las características morfológicas, el método de procesamiento del corpus de entrada, el que se adapta para trabajar con los acentos en el español y otros caracteres especiales del lenguaje. En segundo lugar, se utiliza una distancia de edición para el agrupamiento en familias de palabras, en un proceso muy parecido al método de variedad de sucesores, pero con marcadas diferencias. También, se toman en consideración algunas características de la etimología de las palabras para el agrupamiento.

A diferencia de otras, la distancia que se utiliza introduce la penalización por diferencias en la raíz de las palabras comparadas, lo que mejora la formación de familias, además se puede configurar la flexibilización ante los errores tipográficos y ortográficos, lo que le confiere una posibilidad de tratamiento de corpus con ruido.

Una vez culminado el recorrido por los conceptos fundamentales y los referentes teóricos que sirven de base a esta investigación, se pasará a hace las conclusiones parciales de este capítulo.

2.10 Conclusiones parciales

Una vez analizados todos los elementos expuestos en este capítulo, se han podido demostrar los problemas existentes en las diferentes temáticas analizadas. Empezando por las distancias de edición, se puede ver que la mayoría de los autores no hacen tratamiento diferenciado de la posición en la palabra donde se realizan las transformaciones. Algunos consideran penalizaciones y permiten secuencias de caracteres erróneos en el alineamiento, pero no tiene en cuenta la importancia del tipo de transformación que se realiza. Los que penalizan las transformaciones en la raíz de la palabra, utilizan un prefijo estático para su análisis. Los resultados de similitud obtenidos por estas distancias, en especial para el español y otras lenguas que utilizan la acentuación gráfica, no son los mejores; ya que no tienen en cuenta, en la comparación, los caracteres acentuados.

En la búsqueda por resolver el problema de la detección de entidades, se puede ver que este tema ha sido abordado desde diferentes perspectivas y utilizando distintas técnicas. Pero, a ciencia cierta, no se puede decir que una sea mejor que otra, finalmente los resultados varían indistintamente.

La gran mayoría de los trabajo en la temática del REN, utilizan algún tipo de análisis lingüístico previo para auxiliarse en la tareas tanto de identificación, como de clasificación de las entidades.

Es de destacar que muchos sistemas reportan que las entidades con peores resultados en su reconocimiento son las categorías Miscelánea y Organización. Esto sucede así tanto para los sistemas supervisados como para los no supervisados.

Por otra parte, se ha podido constatar que los resultados alcanzados hasta el momento en temáticas como la Implicación Textual, la Paráfrasis y el Reconocimiento de Entidades, indican que hay que seguir buscando nuevas vías y recurso para mejorarlos. En el caso de la implicación se ha podido apreciar en el estudio de las últimas competiciones, que al intentar acercar a la realidad los corpus de prueba, se ha producido un descenso en los resultados alcanzados.

Ya se comentó anteriormente, la gran utilidad que poseen las expresiones regulares en varias tareas del PLN. Su uso está bien difundido entre los sistemas de esta temática. Uno de sus usos más frecuentes es la escritura de reglas, así como patrones de emparejamiento y extracción. La vía más utilizada para generarlas ha sido la escritura directa en el código de los programas, utilizando para ellos un lenguaje regular. Precisamente estos dos aspectos constituyen un problema que puede ser mejorado. Primero, sacarlas del dominio absoluto del programador es una forma de facilitar su mantenimiento y edición en todo momento. En segundo lugar, obtener una vía más simple para su generación, evitaría a los usuarios la dificultad de aprender -en detalles- la complicada sintaxis de los lenguajes regulares. La mayoría de las aplicaciones desarrolladas para satisfacer este objetivo, presentan algunos problemas, los que fueron debidamente detallados en este capítulo.

En los nueve epígrafes anteriores, se ha podido apreciar que en su totalidad las grandes tareas del PLN necesitan -inevitablemente- de varias tareas intermedias las que, al final, permiten obtener el objetivo buscado, una mejor comprensión del lenguaje humano. Muchos de los sistemas de PLN ven afectados sus resultados debido a la suma de los errores cometidos por estas tareas, mejorarlas implicaría perfeccionar también la forma en que las máquinas intentan entender el lenguaje.

Si bien es cierto que han sido detectados diferentes problemas asociados a las temáticas en estudio, también se puede plantear que existen tanto los conceptos, los recursos, como las vías para intentar solucionarlos. Por esta razón, se trabaja en una propuesta encaminada a transformar la problemática identificada, la que se origina a partir del estudio del marco teórico.

Existen varias características presentes en el estudio computacional del lenguaje que son fruto del análisis de las tareas intermedias antes mencionadas, su integración con el fin de resolver ciertas tareas finales supondrían un gran aporte al estado de la cuestión actual.

Capítulo 3 La metodología desarrollada

En este capítulo se presentan los métodos obtenidos y los resultados alcanzados en esta investigación. En él, se hace un recorrido por las temáticas: Distancia de Edición, Reconocimiento de la Implicación Textual y la Paráfrasis, el Stemming, el Reconocimiento de Entidades Nombradas y por último la Generación de Expresiones Regulares en el PLN.

3.1 Descripción de la propuesta de Distancia Extendida

La búsqueda del grado de parecido entre dos palabras, dos cadenas de caracteres o incluso entre dos documentos, es uno de los temas que más preocupa a los investigadores del PLN. Como se vio en el epígrafe 1.1 y 2.6, existe una amplia investigación sobre esta temática, en la que aún hoy se busca una mejoría de los resultados obtenidos.

En el epígrafe 2.6.1.1, se puede ver que dentro de las distancias léxicas más utilizadas en el PLN está la distancias de Levenshtein, también conocida como distancia de edición (Levenshtein, 1966). Otras, que también se han utilizado con mucha frecuencia son la de Jaro (Jaro, junio 1989), la extensión hecha a ésta publicada en (Winkler, 1999) y la de Needleman y Whunch descrita en (Neeedleman y Wunsch, 1970).

Estas, como otras distancias y medidas de similitud, han sido comparadas con la Distancia Extendida (DEx) descrita en (Fernández, Díaz et al., 2009). Esta propuesta se basa en la idea de introducir algunos elementos como el análisis de la etimología de la palabra, el estudio de la alternancia prosódica y gráfica, así como la flexibilidad ante errores tipográficos y ortográficos.

Todos estos elementos se han intentado poner en función de obtener un mejor cálculo de la similitud entre cadenas de caracteres. A continuación se explican los elementos que se han tenido en cuenta para el desarrollo de esta distancia.

En principio, esta distancia es considerada una extensión del algoritmo de la distancia de edición (DE) o distancia de Levenshtein, pues parte del análisis de la matriz de costos usada por este algoritmo.

A diferencia de la DE, se ofrece la posibilidad de obtener la subsecuancia común más larga (Longest Common Subsequence (LCS)) (Hirschberg, 1977) y diferentes atributos que ayudarían en la determinación de la similitud entre cadenas de caracteres. Además, se ofrecen otros elementos como la posición en la que se han realizado las transformaciones, así con su tipo (inserción, substitución, o borrado), todo esto en una sola iteración.

La nueva propuesta de la Distancia de Edición Extendida (DEx) tiene su primera versión en (Fernández, Díaz et al., 2009). Conociendo la posición en la que se han hecho las transformaciones, se aporta una importante información; pues se podría saber si estas han ocurrido en la raíz de la palabra, penalizándose aquellas que ocurran más a la izquierda, por encima de las que ocurren más a la derecha.

También, se ofrece la posibilidad de colocar un costo de penalización que depende de cuál es el carácter involucrado en el tipo de transformación. Esta distancia se normaliza en el intervalo (0,1).

Para una mejor comprensión de esta transformación, a continuación se detallan todos los pasos del algoritmo. Generar la matriz de la misma manera que se hace en el algoritmo creado por Wagner y Fisher, para más detalles ver en (Wagner y Fisher, 1974). Por ejemplo, para calcular la distancia que existe entre las palabras `educar` y `educadores`, la matriz, según la idea en (Levenshtein, 1965), implementada por (Wagner y Fisher, 1974) quedaría de la siguiente forma:

Tabla 3.1. Matriz resultado de la comparación de las palabras.

		e	d	u	c	a	r
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
u	3	2	1	0	1	2	3
c	4	3	2	1	0	1	2
a	5	4	3	2	1	0	1
d	6	5	4	3	2	1	1
o	7	6	5	4	3	2	2
r	8	7	6	5	4	3	2
e	9	8	7	6	5	4	3
s	10	9	8	7	6	5	4

Como se puede apreciar, la celda extrema inferior derecha tiene un número cuatro, lo que indica que son necesarias cuatro transformaciones para que la palabra `educar` se pueda convertir en `educadores`. El siguiente paso es determinar el camino que corresponda con la subsecuencia común máxima (SCM) (Hirschberg, 1975). Posicionándose en la casilla inferior derecha y recorriendo hacia atrás, pasando a la celda con el valor mínimo, priorizando la diagonal en caso de ser iguales las casillas aledañas, se obtiene el camino buscado.

Tabla 3.2. Matriz después de seleccionada la SCM.

		e	d	u	c	a	r
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
u	3	2	1	0	1	2	3
c	4	3	2	1	0	1	2
a	5	4	3	2	1	0	1
d	6	5	4	3	2	1	1
o	7	6	5	4	3	2	2
r	8	7	6	5	4	3	2
e	9	8	7	6	5	4	3
s	10	9	8	7	6	5	4

De esta forma, partiendo del camino encontrado se genera la cadena de operaciones, interpretando como borrado un movimiento por la vertical, como inserción un movimiento por la horizontal y como sustitución un movimiento en la diagonal. Si la casilla de origen y la de destino tienen el mismo valor, se interpreta como una NO operación, asignándole las letras D, I, S, O, a cada operación respectivamente. Para el caso anterior la cadena de operaciones sería: OOOOODDODD.

Tabla 3.3. Matriz después de seleccionada la SCM.

		e	d	u	c	a	r	
	0	1	2	3	4	5	6	
e	1	0	1	2	3	4	5	O
d	2	1	0	1	2	3	4	O
u	3	2	1	0	1	2	3	O
c	4	3	2	1	0	1	2	O
a	5	4	3	2	1	0	1	O
d	6	5	4	3	2	1	1	D
o	7	6	5	4	3	2	2	D
r	8	7	6	5	4	3	2	O
e	9	8	7	6	5	4	3	D
s	10	9	8	7	6	5	4	D

- Evaluar la ecuación 3.1 con la cadena de operaciones encontrada y los caracteres implicados en cada operación.

$$\mathbf{EDx} = \sqrt[8]{\frac{\sum_{\mathbf{i=0}}^{\mathbf{l-1}} \mathbf{V_{(O_i)}} * \left(\mathbf{P_{(c1_j)}}, \mathbf{P_{(c2_k)}}\right) (\mathbf{2R_{max}} + \mathbf{1})^{\mathbf{L-i}}}{\mathbf{N}}} \qquad 3.1$$

Donde:

O – es la cadena de operaciones necesarias para transformar una palabra en otra.

O_i – Operación en la posición i-ésima.

(O – no operación, I – inserción, D – borrado, s – sustitución)

V – será formalizado como el vector que aparece a continuación:

$$V = \begin{Bmatrix} (0,0) : o \\ (1,0) : i \\ (0,1) : d \\ (1,1) : s \end{Bmatrix}$$

c1 y c2 – son las palabras examinadas.

$c1_j$ – el j-ésimo carácter de la palabra.

c1 $c2_k$ – el k-ésimo carácter de la palabra c2.

P – Es el peso asignado a cada carácter.

El valor del peso de cada carácter, se obtiene a partir del cálculo de la frecuencia de aparición del carácter en un diccionario del lenguaje al que pertenezcan las palabras analizadas. Luego, se ordenan los caracteres en orden descendente según la frecuencia de aparición. Finalmente, se le coloca un número empezando por uno hasta la cantidad de caracteres y en orden inverso. El diccionario completo se muestra a continuación. $P_{(c1_j)}$ – Es el peso del carácter en $c1_j$

P=

a	:52	l	:43	y	:34	)	:25	0	:17	5	:8	è	:50				
i	:51	t	:42	f	:33	(	:25	2	:16	8	:7	\	:1				
e	:50	u	:41	v	:32	q	:24	-	:15	,	:6						
o	:49	d	:40	ó	:49	k	:23	3	:14	/	:5						
s	:48	p	:39	x	:30	é	:50	7	:13	ü	:41						
r	:47	m	:38	z	:29	ú	:41	9	:12	tab	:3						
n	:46	h	:37	í	:51	w	:20	6	:11	=	:3						
esp	:45	g	:36	j	:27	1	:19	.	:10	_	:3						
c	:44	b	:35	á	:52	ñ	:18	4	:9	'	:2						

Nota: esp y tab, se refiere a los caractes espacio y tabulación respectivamente.

Como se muestra, la cantidad de elementos en el conjunto P es 56, pero como se le ha asignado el mismo valor de peso a algunos caracteres, el valor máximo de P es 52.

$$j \text{ es igual a } \begin{Bmatrix} j+1 & si\ O_i \neq I \\ j & si\ O_i = I \end{Bmatrix}$$

$P_{(c2_k)}$ - El peso del carácter en $c2_k$, donde $k = \begin{Bmatrix} k+1 & si\ O_i \neq D \\ k & si\ O_i = D \end{Bmatrix}$

L – Longitud de la palabra más larga del lenguaje que se utiliza.

Para los experimentos en esta investigación se ha tomado 25 como la longitud de la palabra más larga del español.

l – Longitud de la cadena de operaciones de edición.

R_{max} – Cantidad de caracteres en P.

λ – Es el producto de las componentes del vector resultante de $V_{(O_i)}$ $N = \sum_{i=0}^{L-1} 2R_{max}(2R_{max} + 1)^i$. Como se puede observar en la ecuación 3.1, el término $V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)$ es el producto cartesiano que analiza la importancia de efectuar la operación $V_{(O_i)}$ entre los caracteres $P_{(c1_j)}$ y $P_{(c2_k)}$.

El término $(2R_{max} + 1)^{L-i}$ en la ecuación 3.1, penaliza la posición de la operación, de forma tal que mientras más próximo a la izquierda se realice la operación, mayor será la penalización. N es el término que normaliza la distancia en el intervalo [0,1]; con el peor caso posible, que es una cadena de operaciones de longitud L llena de operaciones de sustitución entre los caracteres más costosos del alfabeto.

La raíz octava se aplica buscando que los valores no resulten tan pequeños y que no quede afectada la relación de orden. A continuación se calculan todos los valores para el ejemplo de `educar` y `educadores`.

Tabla 3.4. Calculo de la distancia entre las palabras “educar” y educadores”.

Oi	v(Oi)		C1	PC1	C2	PC2	$\mathbf{V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)}$	i	L-i	$\mathbf{(2R_{max} + 1)^{L-i}}$	$\mathbf{V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max} + 1)^{L-i}}$
O	0	0	e	50	e	50	0	0	24	1.87880905060953E+49	0
O	0	0	d	40	d	40	0	1	23	1.66266287664561E+47	0
O	0	0	u	41	u	41	0	2	22	1.47138307667753E+45	0
O	0	0	c	44	c	44	0	3	21	1.30210891741374E+43	0
O	0	0	a	52	a	52	0	4	20	1.15230877647234E+41	0
D	0	1	r	47	d	40	47	5	19	1.01974228006402E+39	4.79278871630087E+40
D	0	1		49	o	49	49	6	18	9.02426796516828E+36	4.42189130293246E+38 0
O	0	0		43	r	47	0	7	17	7.98607784528166E+34	0
D	0	1			e	50	0	8	16	7.06732552679793E+32	0
D	0	1		23	s	48	23	9	15	6.25427037769728E+30	1.43848218687038E+32
Datos							$\sum_{i=0}^{l-1} \mathbf{V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max} + 1)^{L-i}}$				4.83700764371502E+40
l	L	Rmax									
10	24	56									
$\mathbf{N = \sum_{i=0}^{L-1}(2R_{max} + 1)^i}$							$\mathbf{\frac{\sum_{i=0}^{l-1} V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max} + 1)^{L-i}}{N}}$				2.27832505725485E-11
N=	2,12305422718877E+51						$\mathbf{EDx = \sqrt[8]{\frac{\sum_{i=0}^{l-1} V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max} + 1)^{L-i}}{N}}}$				0.046741426

Como la distancia se evalúa a partir de la cadena de operaciones mínimas y ésta es generada por la aplicación del algoritmo de cálculo de la SCM, en la matriz de programación dinámica para la DEx, el orden del algoritmo queda igual que en la DE ($O(m,n)$), donde `m` y `n` son las longitudes de las cadenas comparadas.

3.1.1 Experimento con la Distancia Extendida

Para poder comprobar que el método creado obtiene los resultados esperados, se parte la idea de hacer una comparación con otras métricas. El experimento intenta comprobar la capacidad que tienen los diferentes algoritmos para el cálculo de la distancia entre una palabra seleccionada como pivote y su respectiva familia de palabras.

3.1.1.1 Medición de distancia entre palabras de misma familias

La hipótesis de partida es que la mayoría de las palabras pertenecientes a una familia deben compartir la misma raíz o tallo. Esto presupone que la distancia desde todas ellas al pivote seleccionado, debe presentar variaciones muy pequeñas, pues solo cambiarían las terminaciones de las palabras. Si se grafica esta información la curva resultante debería tender a una recta, ya que las distancias deben ser relativamente pequeñas.

Se tomaron para el experimento, aleatoriamente, cuatro familias de palabras para los pivotes `empaña`, `baila`, `enseña` y por último la palabra `familia`. Los conjuntos están compuestos por 50, 57, 100 y 73 palabras respectivamente.

3.1.1.2 Resultados

Los resultados que se muestran en las gráficas que ilustran el experimento (Figura 3.1, Figura 3.2, Figura 3.3 y la Figura 3.4), fueron obtenidos al evaluar la distancia desde los pivotes antes mencionados, hasta cada una de las palabras de sus respectivas familias. A continuación se muestra una leyenda con la codificación utilizada para designar cada una de las distancias:

Leyenda	**-Distancia utilizada**
A	-Distancia Extendida
B	-ChapmanLengthDeviation
C	-ChapmanMeanLength
D	-Jaro
E	-JaroWinkler
F	-Levenshtein
G	-NeedlemanWunch
H	-QGramsDistance

Es válido aclarar que solo se utilizan estas distancias de todas las que aparecen en la API de SimMetric Library, pues las demás distancias contenidas en esta API no arrojan buenos resultados. Solo se toman para el experimento las de mejor comportamiento.

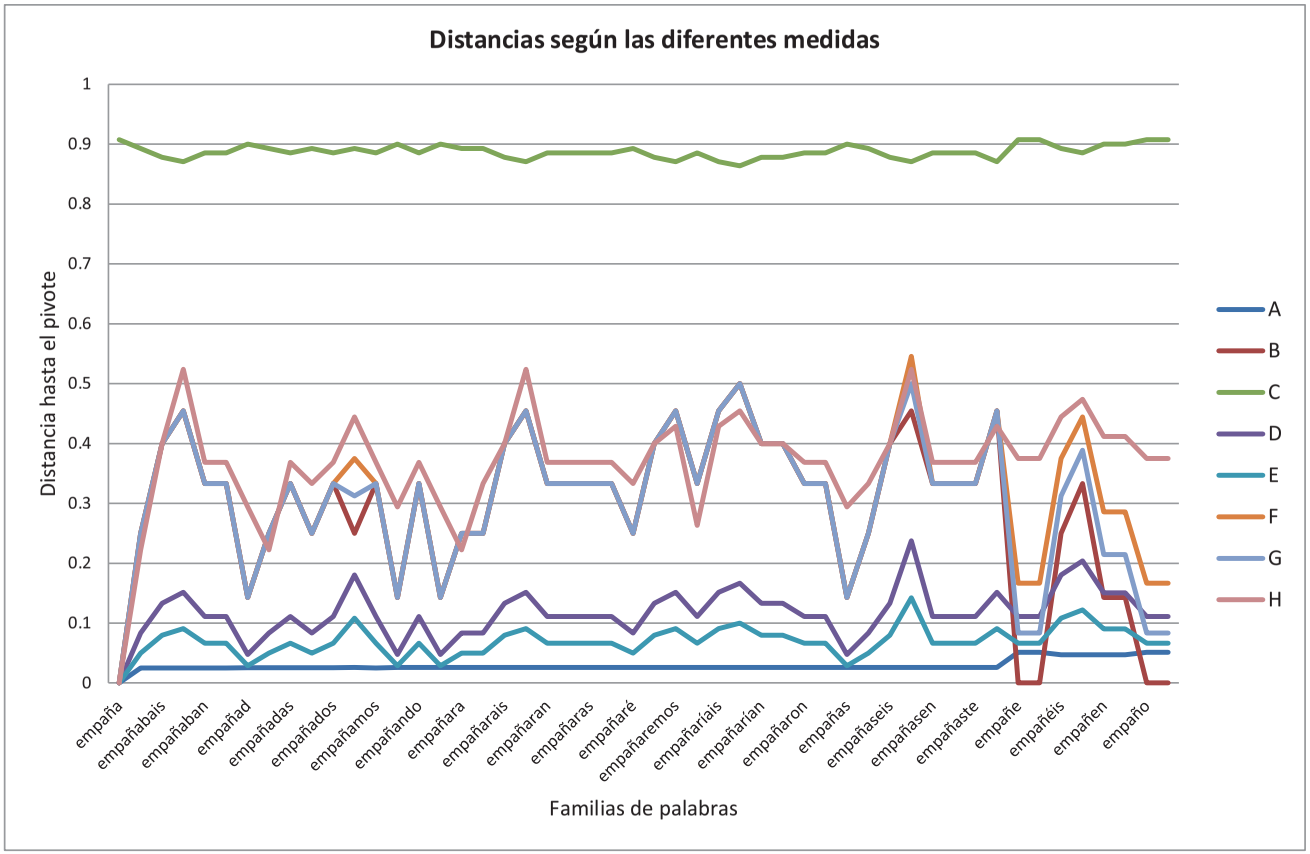


Figura 3.1. Distancia entre el pivote EMPAÑA y su familia de palabras.

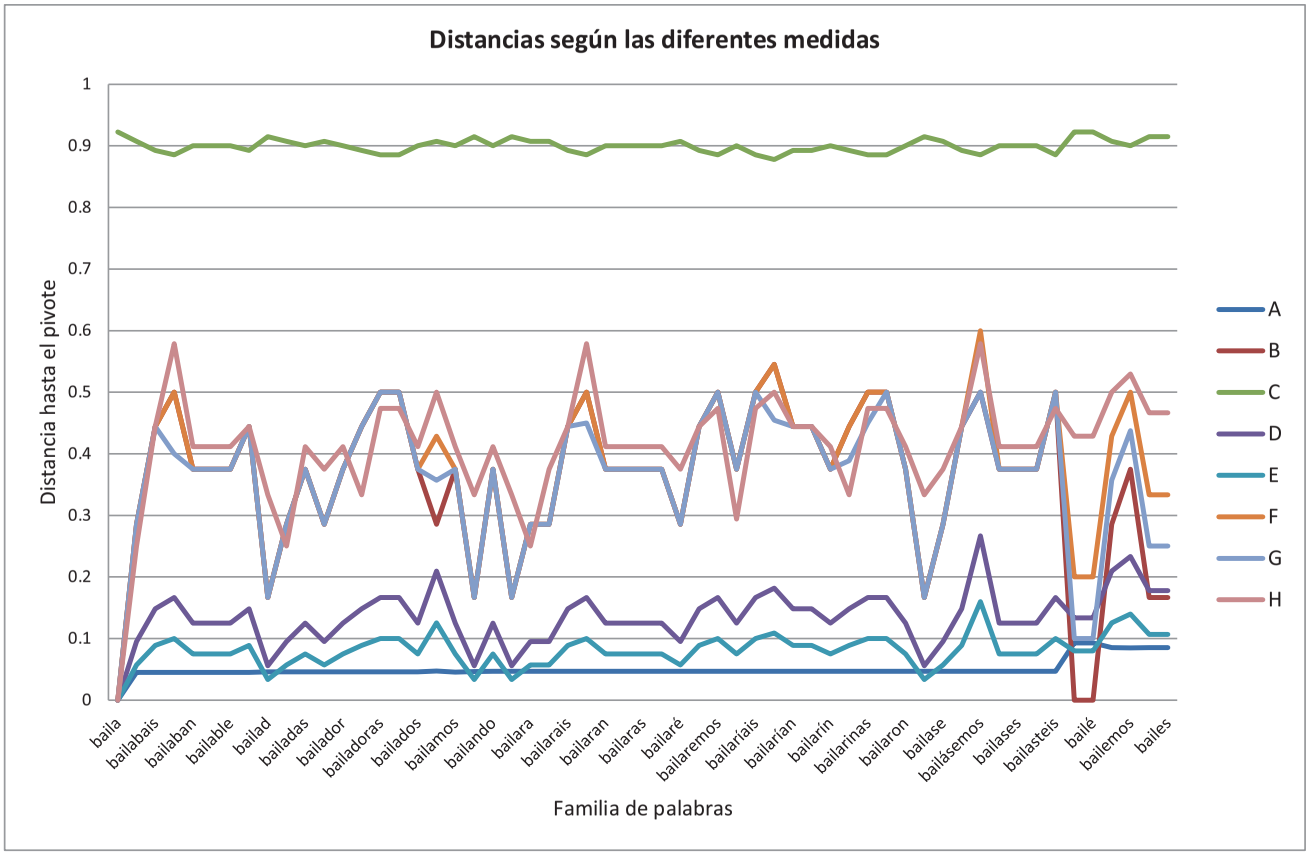


Figura 3.2. Distancia entre el pivote BAILA y su familia de palabras.

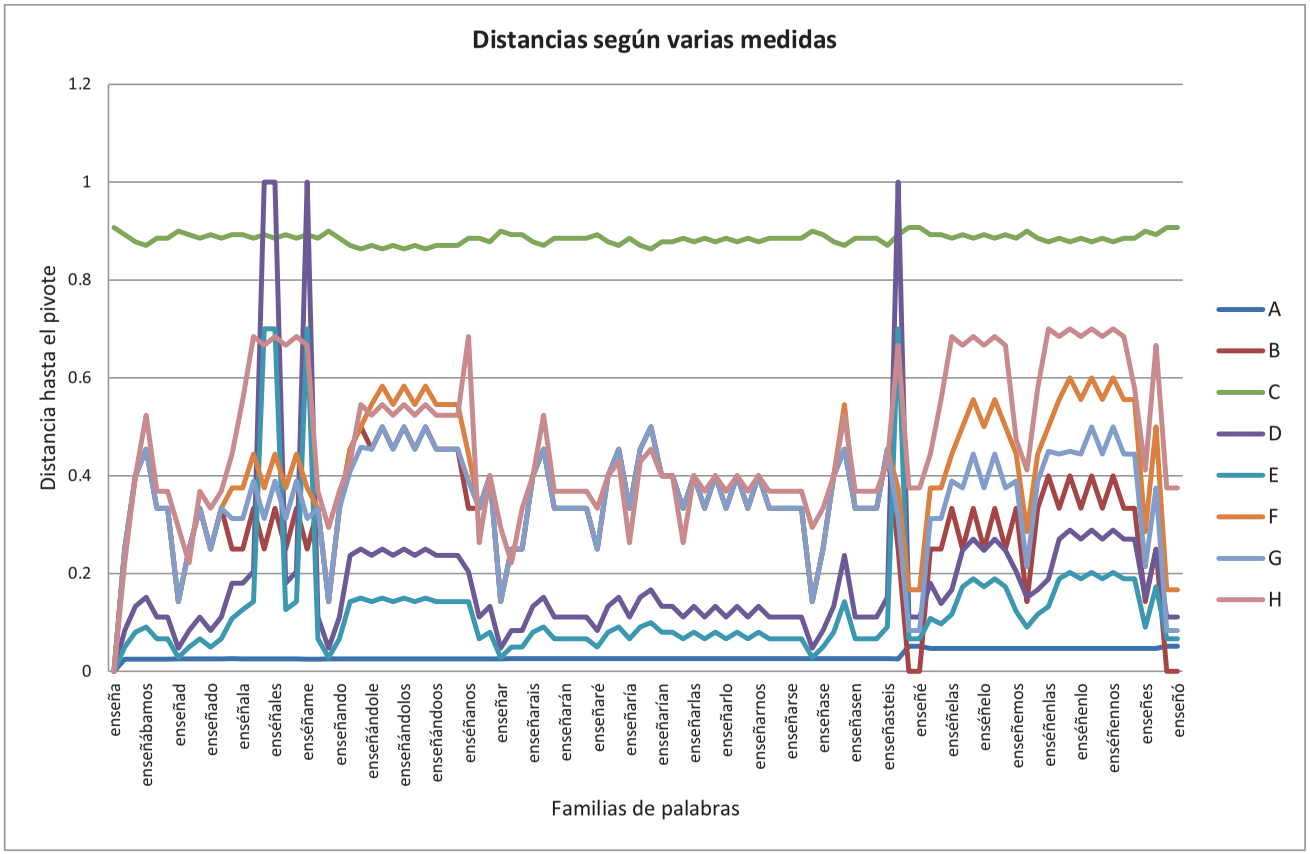


Figura 3.3. Distancia entre el pivote ENSEÑA y su familia de palabras.

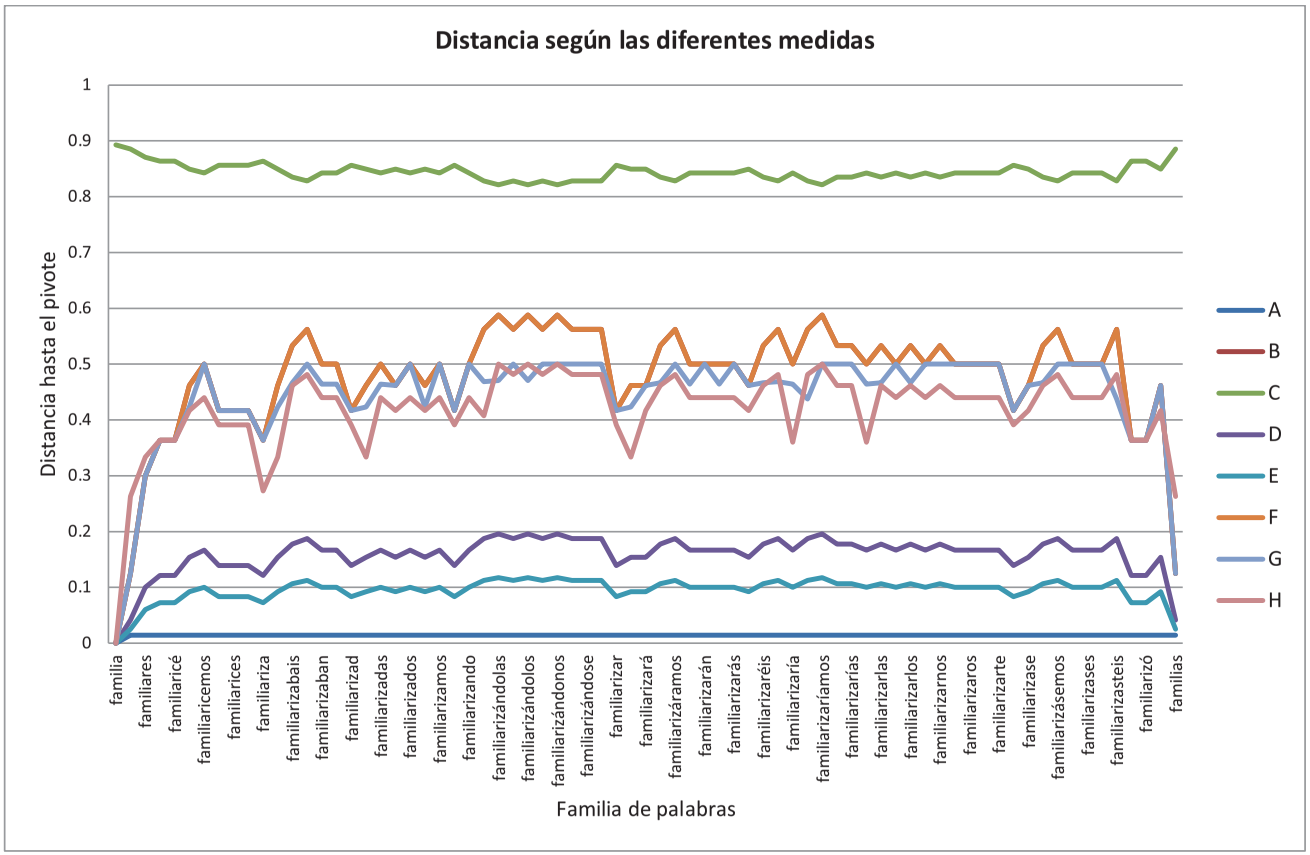


Figura 3.4. Distancia entre el pivote familia y su familia de palabra

3.1.1.3 Análisis de los resultados

Como se puede apreciar en las figuras anteriores, la distancia DEx mantiene los valores entre todas las palabras con una tendencia a una línea recta, lo que es lógico, pues la distancia entre palabras de una misma familia debe ser muy pequeña.

Se puede ver en las gráficas que las restantes distancias, en algunos casos, dan valores con una gran desviación, este es el caso de las palabras acentuadas y algunas con un sufijo relativamente largo. Esto indica que las distancias utilizadas para la comparación, no tienen en cuenta algunos caracteres propios de algunas lenguas, como podrían ser el acento gráfico o la letra ñ en el español.

Además, se observa que las distancias, con excepción de ChapmanMeanLength y la DEx, son muy dependientes de la longitud de las palabras comparadas.

Por otra parte -estos algoritmos- en comparación con el que se implementa en la DEx, no tienen en cuenta que estas palabras comparten un tallo o raíz común entre ellas y que en la mayoría de los casos las diferencias están en los sufijos.

Este no es el único experimento realizado con esta distancia (DEx), en próximos epígrafes se estará valorando la efectividad de esta métrica en otras contextos.

En el siguiente epígrafe se describe el trabajo realizado con relación a la influencia de la polaridad sentimental en la implicación textual.

3.2 El Reconocimiento de la Implicación Textual

Como elemento fundamental en esta temática, se persigue comprobar la influencia que tiene la polaridad sentimental en el reconocimiento de la implicación textual. En los epígrafes siguientes se describen los experimentos realizados al respecto.

3.2.1 El RIT y la Polaridad Sentimental

El reconocimiento de la implicación textual, para este trabajo en específico, será visto como una tarea de clasificación. El único y principal atributo utilizado para el análisis es la polaridad sentimental extraída del par texto-hipótesis.

Se utilizan además, como conjunto de datos los ofrecidos por las competiciones Pascal Challengers RTE1 (Dagan, Glickman et al., 2005), RTE2 (Bar-Haim, Dagan et al., 2006), RTE3 (Giampiccolo, Magnini et al., 2007), TAC 2008 RTE track RTE4 (solo la versión 2-way) (Giampiccolo, Dang et al., 2009), ENGARTE[48], Lexical y BPI[49].

El objetivo no es crear un sistema de RTE, sino comprobar cómo se comporta la polaridad sentimental con relación a la implicación textual.

Para el buen funcionamiento de esta investigación se tuvieron en cuenta un grupo de recursos como ISR-WN (Gutiérrez, Fernández et al., 2010a), una herramienta que permite la integración de varios recursos semánticos mapeados a WordNet (Miller, Beckwith et al., 1990), el que es usado como núcleo para enlazar recursos como SUMO[50] (Suggested Upper Merged Ontology) (Niles y Pease, 2001), WordNet Domains (WND) (Magnini y Cavaglia, 2000), WordNet Affect (WNA) (Valitutti, Strapparava et al., 2004), Semantic Class (SC) (Izquierdo, Suárez et al., 2007) y SentiWordNet (SWN) (Esuli y Sebastiani, 2006). ISR-WN permite la navegación a través de estas redes semánticas, considerándolas como un todo.

Es preciso aclarar que, de todos estos recursos, sólo SWN es un recurso léxico. En él cada synset de WN se asocia a dos puntuaciones numéricas, Pos(s) y Neg(s) pudiendo obtener una tercera Obj(s) mediante el cálculo: Obj(s) = 1 -| Pos(s) - Neg(s)|. Cada puntuación describe cuán objetivo, positivo y negativo son los términos contenidos en el synset. Eso significa que un synset tendría tres propiedades de opinión relacionadas con un cierto grado (por ejemplo, atrocious#3 (sentido 3 de atroz), tendría [Pos: 0|Neg: 0.625 |Obj: 0.375]).

[48] http://nlp.uned.es/qa/ave

[49] http://wwww.cs.utexas.edu/~pclark/bpi-text-suite

[50] http://www.ontologyportal.org/

Usando ISR-WN, se etiquetan todos los pares texto-hipótesis con la información de polaridad sentimental correspondiente. Luego se chequea el vínculo existente entre una condición de implicación textual y la relación de polaridad existente.

Para este propósito, se realizan varios experimentos. Todos con el objetivo de determinar la influencia de la polaridad sentimental en el RTE.

Es importante, para poder entender cómo se realiza esta investigación, comentar brevemente los elementos fundamentales de IRS-WN.

3.2.1.1 Integración de los recursos semánticos basados en WordNet (ISR-WN)

Con el objetivo de aplicar la polaridad multidimensional que ISR-WN facilita, se analizan otras aproximaciones relacionadas con la temática como son (Zubaryeva y Savoy, 2010), (Nairn, Condoravdi et al., 2006), (Gutiérrez, Vázquez et al., 2011), así como otras publicaciones de la competición NTCIR-8 MOAT (NTCIR-8_MOAT, 2010), que tienen en cuenta la polaridad de las frases.

A partir de este análisis, es que se decide utilizar Senti-RST (Gutiérrez, Vázquez et al., 2011), debido a que esta es una aproximación que permite ser aplicada sobre varias dimensiones (léase recursos) al mismo tiempo, obteniéndose fácilmente la polaridad de la frase ofrecida por cada uno de ellos.

Por su parte, Senti-RST es un método no supervisado, basado en conocimiento que utiliza la técnica de Árboles Semánticos Relevantes (ASR), en inglés Relevant Semantic Tree (RST) (Gutiérrez, Vázquez et al., 2011); Estos árboles son combinados con SentiWordNet 3.0 (Esuli y Sebastiani, 2006). El objetivo de Senti-RST es obtener el ASR de cada frase y asociarlo con valores de polaridad sentimental.

Este proceso, involucra a los recursos antes mencionados: WND, WNA, WN, SUMO y SC. Debido a que las SC no tienen una estructura de árbol, se suple en este caso obteniendo las clases semánticas relevantes. Consecuentemente, se determinan las polaridades obtenidas para cada etiqueta, de cada ASR obtenidos, de acuerdo con la frase analizada.

Es importante destacar que, el ASR original es un método capaz de resolver la ambigüedad, creando ASRs de las frases basado en cada dimensión semántica de ISR-WN. Para medir la asociación entre conceptos y palabras en cada frase, de acuerdo con una perspectiva multidimensional, ASR usa la medida del radio de asociación (AR) (Vázquez, Montoyo et al., 2004) adaptada en (Gutiérrez, Vázquez et al., 2011).

Esta propuesta, lo que intenta es incluir el análisis semántico multidimensional en el análisis de opiniones usando ASR. La idea involucra cuatro pasos, los que serán explicados a continuación.

1. Obtención de los Árboles Semánticos Relevantes.

En este trabajo, se usa un fragmento del método original de ASR con el objetivo de obtener los ASR de las frases. Nótese que este paso tiene que ser realizado para cada recurso.

Una vez que cada frase es analizada, se obtiene el valor de AR relacionado con cada concepto en el árbol. La ecuación 3.2 se usa para medir y obtener los valores de los conceptos relevantes.

$$AR(C, f) = \sum_{i=1}^{n} AR(C, f_i); \quad 3.2$$

Donde:

$$AR(C, w) = P(C, w) * log_2 \frac{P(C, w)}{P(C)}; \quad 3.3$$

En ambas ecuaciones C es un concepto; f es una frase o conjunto de palabras (w); f_i es la i-ésima palabra de la frase f; $P(C, W)$ es la distribución de probabilidad conjunta y $P(C)$ es la probabilidad marginal.

Usando el recurso WND, es la forma típica de obtener el ASR. El primer paso involucra la lematización de las palabras de la frase. Luego, cada lema es buscado en ISR-WN y es relacionado con los conceptos de WND.

Después de obtener el vector inicial de dominios, se aplica la ecuación 3.4 para obtener el ASR relacionado con la frase.

$$AR(PC, f) = AR(ChC, f) - ND(IC, PC); \quad 3.4$$

Donde:

$$ND(IC, PC) = \frac{MP(IC, PC)}{TD}; \quad 3.5$$

En la ecuación 3.4, ChC es el concepto hijo de PC; ND es la distancia normalizada entre el concepto inicial y el concepto padre; IC es el concepto inicial de donde se tiene que adicionar el ancestro; PC es el concepto padre. En la ecuación 3.5, TD es la profundidad del árbol jerárquico del recurso usado, MP es el camino mínimo entre el concepto inicial y el concepto padre.

De aquí que, $AR(PC, f)$ representa el valor del radio de asociación del concepto padre (PC) relacionado con la frase (f); $AR(ChC, f)$ es el valor del radio de asociación, calculado con la ecuación 3.2 en caso de que ChC sea incluido en el vector inicial, de otra forma se calcula con la ecuación 3.4.

Aplicando la ecuación 3.4, el algoritmo decide cuál concepto padre será adicionado al vector. (Véase en ejemplo siguiente):

Si ($AR(PC, f)\ value\ >\ 0$)

{Si (PC no ha sido adicionado al vector)

PC es adicionado al vector con el valor de $AR(PC, f)$;

Sino $PC\ =\ PC\ +\ AR(PC, f)$;}

El vector obtenido representa el árbol de dominio asociado a la frase. Después que se obtiene el ASR. Valga la aclaración de que el dominio Factotum se elimina del árbol (ya que no provee de información útil (Magnini y Cavaglia, 2000) y experimentalmente se ha comprobado que introduce errores).

2. Obtención de los árboles semánticos positivos

Para obtener los Árboles Semánticos Positivos (ASP), en inglés Positive Semantic Trees (PST) de la frase, se sigue el mismo procedimiento que en el paso 1. En este caso, el valor de AR será remplazado por el valor de polaridad perteneciente al sentido analizado. La polaridad se obtiene del recurso SentiWordNet 3.0, donde cada sentido de la palabra en ISR-WN para WordNet versión 2.0, es mapeado a WordNet versión 3.0. Por lo tanto, se puede encontrar cada sentido de ISR-WN en SentiWorNet 3.0 y obtener las polaridades respectivas. Este nuevo valor será llamado Asociación Positiva ($PosA$). La $PosA$ es calculada usando la ecuación 3.6

$$PosA(C, f) = \sum_{i=1}^{n} PosA(C, f_i)\,; \tag{3.6}$$

$$PosA(C, w) = \sum_{i=1}^{n} PosA(C, w_i)\,; \tag{3.7}$$

Donde:

C es un concepto; f es una frase o conjunto de palabras w; f_i es la i-ésima palabra de la frase f; $PosA(C, w_i)$ es el valor positivo del sentido de w_i con relación a C.

$PosA$ es usado para medir el valor positivo asociado a las hojas del árbol semántico, donde los conceptos son colocados. Seguidamente, usando la misma estructura de ASR se crea el nuevo árbol semántico sin los valores se AR. En su lugar, las hojas con los conceptos de este nuevo árbol semántico serán anotadas con los valores se $PosA$.

Más tarde, para asignar los valores positivos al concepto padre, cada concepto padre acumulará el valor positivo de los conceptos hijos. La ecuación 3.8 es aplicada en un proceso de abajo hacia arriba.

$$PosA(PC) = \sum_{i=1}^{n} PosA(ChC\ ); \qquad 3.8$$

Donde PC es el concepto padre; ChC es el concepto hijo de PC y $PosA(ChC)$ representa el valor positivo de ChC.

3. Obtención de los árboles semánticos negativo

Esta fase es idéntica a la descrita en el paso anterior, solo que se adaptan todos los mecanismos orientados a obtener los valores negativos en lugar de los positivos. Para este caso se designan los Árboles Semánticos Negativos como ASN.

4. Obtención la polaridad de la frase

En este paso, es necesario decidir cuál polaridad es más representativa de acuerdo a los árboles semánticos obtenidos de cada dimensión (recurso). Para esto, se combinan los ASR con los ASP y ASR con los ASN. Dependiendo del resultado obtenido, se clasifica la frase en positiva, negativa o neutral. Por supuesto, sino existe un ASR la frase es etiquetada como UNKNOW (UNK).

Antes de realizar este paso, no es necesario normalizar (con valores entre cero y uno) los tres tipos de árboles semánticos (ASR, ASP y ASN) para cada dimensión, porque en el artículo publicado en (Gutiérrez, Vázquez et al., 2011) se demostró que esto introduce errores.

El objetivo es asignar más peso a la polaridad relacionada al concepto más relevante en cada árbol ASR. La ecuación 3.9 muestra los pasos a seguir para obtener los valores semánticos positivos.

$$ACPosA(RST, PST) = \sum_{i=1} RST_i * PST_i\ ; \qquad 3.9$$

Donde $ACPosA$ es el valor semántico positivo de la frase analizada, obtenido para una dimensión; ASR es el árbol semántico relevantes ordenado con el formato: ASR [Concept | AR]; ASP es el

árbol semántica positivo ordenado según la estructura ASR con el formato: ASP [Concept | $PosA$]; RST_i es el valor i-ésimo AR del concepto i; PST_i es el valor i-ésimo $PosA$ del concepto i.

Luego, para medir el valor semántico negativo $ACNegA$, se emplea una ecuación similar remplazando ASP con ASN. Después de obtener los requisitos de opinión semánticos, se evalúa este enfoque en 13 conjuntos de datos anotados del RTE, para las tareas de inglés monolingüe.

Esta propuesta consiste en la acumulación de los valores $ACPos$ y $ACNeg$ de todas las dimensiones y compararlas. Para una representación más corta nombramos este método ALLRec. Los valores acumulados positivos y negativos, serán nombrados $ACPosD$ y $ACNegD$ respectivamente. En el caso que $ACPosD > ACNegD$ el valor asignado es POS, si $ACPosD < ACNegD$ el valor asignado es NEG. Contrariamente a como se realiza en (Gutiérrez, Vázquez et al., 2011), en esta propuesta, si no es posible la creación de la ASR de la frase analizada, entonces, no hay ASN, ni ASP, tampoco $ACPosD$ y $ACPosD$, por lo tanto no habrá resultado de polaridad. En este caso, el valor asignado es UNK (respuesta Desconocido). De lo contrario, el valor es asignado como NEU (neutral).

5. Calculando la polaridad del par texto-hipótesis

La idea en este caso es determinar qué sucede con la polaridad sentimental del par texto-hipótesis con relación a la condición de implicación.

La hipótesis de partida es la siguiente: Si la polaridad del texto o de la hipótesis son desconocidas, no se puede clasificar. De otra forma, si la polaridad del texto es diferente a la polaridad de la hipótesis, no debe haber implicación textual. Por último, si la polaridad del texto es igual a la polaridad de la hipótesis, hay ambigüedad, no se puede decidir si hay o no implicación textual.

Con el fin de evaluar el enfoque se usan las reglas y corpus referentes a la tarea del reconocimiento de la implicación textual monolingüe para el inglés (RTE-1, RTE-2, RTE-3, RT4, ENGARTE2, léxico y BPI).

Estos conjuntos de datos antes mencionados, no tienen la misma estructura, por ejemplo, RTE3 y RTE4 incluye la clasificación de 3 vías.

Por este motivo, todos los conjuntos de datos se normalizan convirtiéndolos a un mismo formato, con el fin de utilizar la misma estructura para la clasificación de la implicación. Se utilizan solo los archivos anotados con la información de implicación textual. La Tabla 3.5 muestra todos los parámetros de la transformación de los conjuntos de datos.

Tabla 3.5. Parámetros para el par texto-hipótesis.

Id	Número de identificación del par
entailment	condición de implicación
task	Tarea de la que se extrae el par
length	Longitud del par
<t>“”</t>	Palabras del texto
<h>“”</h>	Palabras de la hipótesis

La condición del implicación toma los valores: entailment o non-entailment. Todos los valores se convierten a este tipo de clasificación.

Después de esto, todos los conjuntos de datos son procesados con Senti-RST. Luego, se ejecutan los pasos 1, 2, 3 y 4 explicados anteriormente. Posteriormente, se obtiene la salida de los seis ficheros anotados con la información de polaridad (Pos, Neg y Neu), un fichero para cada recurso (WNA, WND, SC, SUMO, WN y ALLRec) descritos en el paso 4, de acuerdo a $ACNegD$ y $ACPosD$. Como fue mencionado, en el paso 4 cada dimensión (recurso) es combinada con SentiWordNet como núcleo para la detección de polaridad.

Finalmente, se etiquetan 6484[51] pares de texto-hipótesis de 13 conjuntos de datos con información de polaridad y la relación de implicación, dependiendo de las reglas siguientes:

1. Si un par de texto-hipótesis tiene una polaridad desigual y no hay implicación, esta relación se etiqueta como VERDADERA.
2. Si un par de texto hipótesis tiene una polaridad desigual y existe implicación, se etiqueta esta relación como FALSA. Podría suceder que haya una relación de polaridad Positivo/Neutro o Negativo/Neutro y sin embargo sí exista implicación, pero no debería suceder con una relación opuesta de Positivo/Negativo. Para este experimento, los casos en los que ocurre una relación Positivo/Neutro o Negativo/Neutro también fueron tomados como una relación FALSA.

Los siguientes casos introducen cierta ambigüedad, por lo que se analizan de la forma siguiente:

3. Si un par de texto-hipótesis tiene una polaridad idéntica y no hay implicación, esta situación es ciertamente probable, por lo que se etiqueta esta relación como POS; por lo tanto será considerada VERDADERA.
4. Si al menos una de las frases del par texto-hipótesis fue etiquetado por Senti-RST con UNK en sus polaridades, este par se excluirá de cualquier cálculo. Debido a que es imposible tomar cualquier decisión sin tener el valor de polaridad.

[51]RTE1, RTE2, RTE3, RT4, Lexical, Engarte2 and BPI datasets.

Principalmente se intenta demostrar la verdad de la primera regla a través de un experimento, el que se presenta a continuación.

3.2.1.2 El experimento con la Polaridad Sentimental y la Implicación Textual

Se realizan varios experimentos intentando verificar el efecto de la polaridad en la implicación. Específicamente, se desea buscar qué polaridad existe en cada tipo de relación (entailment y el non-entailment), que un par texto-hipótesis podría tener. Y verificar -si se puede clasificar- a partir de la relación de polaridad, la condición de implicación en dicho par de frases.

Como existen varios recursos que aporta información de polaridad, se intenta unificar el criterio para dar un valor definitivo de polaridad al par texto-hipótesis. Usando la salida obtenida de Senti-RST, se hace una decisión de votación para asignar la polaridad final. La votación toma en consideración el resultado (13 archivos de texto con los conjuntos de datos del par-hipótesis anotados con la polaridad y la información de implicación) obtenido de los recursos WNA, WND, SC, SUMO, WN y ALLRec.

Las condiciones para la votación son las siguientes:

1. Si la decisión de polaridad positiva y la negativa de los diferentes recursos obtienen el mismo valor, entonces se decide asignar polaridad neutra, ya que no hay acuerdo entre ellos.
2. Si la decisión de polaridad positiva y la neutra de los diferentes recursos obtienen el mismo valor, entonces se decide asignar polaridad positiva.
3. Si la decisión de polaridad negativa y la neutra de los diferentes recursos obtienen el mismo valor, entonces se decide asignar polaridad negativa.
4. De otra manera, la polaridad que más sea seleccionada es la que prevalece como polaridad de la frase.

Al final de la etapa de clasificación y etiquetado (pasos del 1 al 4), se obtienen 6484 pares de texto-hipótesis para los 13 conjuntos de datos, anotados con información de la polaridad asignada por cinco diferentes recursos, más una decisión en la que la polaridad se calcula utilizándolos a todos. También, se debe recordar que se decide hacer una votación considerando las seis evaluaciones para otorgar la calificación final.

La Tabla 3.6 muestra un resumen de los resultados de los experimentos. Las filas desde la uno a la cuatro exhiben los análisis obtenidos por pares con polaridad desigual y la relación non-entailment (UPNE), los pares con polaridad desigual y relación de entailment (UPE), para los pares con

igualdad de polaridad y relación de entailment (EPE) y por último para los pares con igual polaridad y la relación non-entailmnet (EPNE). Las columnas desde la dos a seis contienen los resultados de cada uno de los recursos. La columna siete visualiza el resultado de ALLRec y en la última columna se puede ver el resultado de la votación final.

Los términos en la fila seis a la nueve de la Tabla 3.6, se calculan con la ecuación 3.10 que aparece a continuación.

$$PNE = \frac{UPNE}{UPNE + EPNE}; \quad 3.10$$

Donde:

PNE: es la precisión detectando la relación non-entailment. El término UPNE, es el número de pares con relación de non-entailment con polaridad desigual (es considerado como CORRECTO non-entailment detectado) y por último EPNE, es el número de pares con relación de non-entailment con la misma polaridad. Aún cuando se sabe que una relación de non-entailment puede tener la misma polaridad, esta situación se considera como INCORRECTO non-entailment detectado, ya que contrasta con nuestro propósito de determinar la condición de non-entailment a partir de una relación desigual de polaridad de los pares texto-hipótesis.

$$PPE = \frac{EPE}{EPE + UPE}; \quad 3.11$$

Donde:

PPE: es la precisión detectando la posibles relación de entailment. El término EPE, es el número de pares con relación de entailment y con la misma polaridad (es considerado como CORRECTO entailment detectado) y finalmente UPE, es el número de pares con relación de entailment con polaridad desigual, este caso es considerado INCORRECTO entailment detectado.

También se calcula la exactitud usando la ecuación 3.12 de la forma siguiente:

$$A = \frac{H}{H + F}; \quad 3.12$$

Donde:

A: es la exactitud, H es el número de éxitos detectando correctamente la relación de non-entailment y posible entailment. F: es el número de fallos cometidos también al detectar la relación non-entailment y posible entailment.

La precisión promedio (Bar-Haim, Dagan et al., 2006) se calcula mediante la ecuación 3.13 que aparece a continuación:

$$AP = \frac{1}{R}\sum_{i=1}^{n}\frac{E(i)x\#EntailmentUptopair(i)}{i}; \qquad 3.13$$

Donde:

n: es el número de pares en el conjunto de prueba, R es el número total de pares positivos en el conjunto de prueba, $E(i)$: es 1 si el i-ésimo par es positivo y 0 en caso contrario, i es el rango de pares, ordenados por su ranking.

Como se ha mencionado anteriormente, en el paso 5, los pares anotados con *POS* o *NPR* son excluidos de este análisis. Los pares anotados con *POS*, debido a que no se puede decidir si es o no una implicación. Los anotados con *NPR* también son excluidos porque que no existe algún tipo de información de polaridad para poder asignarla a los pares. Esto pares no son un error de este método, recuérdese que se parte de la información aportada por otros recursos.

Es importante señalar que -hasta la fecha en que se realiza este experimento- los sistemas para la determinación de la relaciones de polaridad entre frases, mostraban gran inexactitud (NTCIR-8_MOAT, 2010), con un 51% para el mejor resultado de F-medida. El sistema que obtiene esta puntuación, UNINE (Zubaryeva y Savoy, 2010), obtiene los términos relevantes identificados con un tipo de polaridad específica, con el fin de obtener la polaridad de la frase.

El método elegido (Senti-RST), para detectar la polaridad de las frases presenta una diferencia de 10 puntos con respecto a UNINE, cuando fue evaluado sobre el corpus de FOSA (Gutiérrez, Vázquez et al., 2011). Sin embargo, se ha elegido este método porque se quiere evaluar la influencia de la polaridad, pero obtenida de cada uno de los recursos integrados en ISR-WN, con el fin de hacer futuras comparaciones.

3.2.1.3 Resultados y análisis

A continuación se presentan en la Tabla 3.6 los resultados de todos los experimentos realizados. En las columnas, desde la dos a la siete, se muestran los valores alcanzados por cada uno de los recursos y al final la votación entre ellos.

Haciendo un análisis de los resultados en la Tabla 3.6, se puede ver que SUMO no solo obtiene el mayor número de pares con relación non-entailment con polaridad desigual (*UPNE* =1801), sino

que también obtiene el mayor número de pares con relación de entailment con igual polaridad entre ellos (*UPE*=1816).

Al ver la cantidad de pares con *UPNE*, se puede pensar que SUMO podría tener el mejor valor de precisión en la determinación de la relación non-entailment (*PNE)*, pero no es ninguna sorpresa que la técnica de votación obtenga el mejor resultado para este parámetro (*PNE= 0,562*). Si se observa bien la relación *UPNE* y *EPNE*, se entenderá mejor este comportamiento.

Tabla 3.6. Resultados de determinación de influencia de la polaridad en la implicación.

Descripción	WNA	WND	SC	SUMO	WN	ALLRec	Votación
pares con polaridad desigual y non-entailment con (UPNE)	476	1733	1474	1801	1724	1409	1752
pares con polaridad desigual y entailment (UPE)	426	1809	1565	1816	1751	1485	1755
pares con polaridad igual y entailment (EPE)	355	667	839	730	744	880	1462
pares con polaridad igual y non-entailment (EPNE)	680	1432	1792	1466	1540	1855	1366
no hay relación de polaridad (NPR)	2498	95	0	1	4	0	0
Precisión detectando non-entailment (PNE)	0.412	0.548	0.451	0.551	0.528	0.432	0.562
Precisión detectando posible entailment (PPE)	0.455	0.269	0.349	0.287	0.298	0.372	0.454
exactitud (A)	0.662	0.568	0.598	0.581	0.584	0.609	0.646
Precisión promedio (AP)	0.745	0.678	0.684	0.675	0.690	0.725	0.678

Tomando en consideración la relación entre la *UPE* (pares con polaridad desigual y entailment) y la *EPE* (pares con polaridad igual y entailment), WNA obtiene el mejor resultado detectando una relación de entailment cuando hay polaridades iguales (*PPE* (0.455)). También obtiene el mejor valor de precisión promedio (véase la Tabla 3.6). Por desgracia, este recurso contiene etiquetas de dominios afectivos que se correlacionan solo con palabras que denotan estados emocionales. WNA, no está lo suficientemente poblado como para ofrecer una etiqueta para cada palabra examinada, fallando en muchos casos (38,5%).

Por supuesto, WNA no puede ser utilizado de forma aislada, pero sí es un buen aporte para complementar otros recursos. Se decide mantener WNA en la votación, a pesar de sus problemas, porque cuando puede actuar lo hace muy bien.

La Tabla 3.6, también muestra más casos anotados con *UPE* que como *EPE*, sin embargo no tiene sentido que ocurra una relación de entailment con una distribución de polaridad desigual entre los pares. Esto es debido a los posibles errores de Senti-RST estimando la polaridad. Téngase en cuenta que Senti-RST también depende de otros recursos que introducen errores. Considérese el siguiente ejemplo extraído del conjunto de datos del RTE1 (dev.xml, id par = 19):

<t>Researchers at the Harvard School of Public Health say that people who drink coffee may be doing a lot more than keeping themselves awake - this kind of consumption apparently also can help reduce the risk of diseases.</t>

<h>Coffee drinking has health benefits.</h>

Estas dos frases parecen ser positivas, pero la relación de polaridad estimada por Senti-RST no las muestra así (véase la Tabla 3.7). Por esta razón, también la votación asignada es incorrecta.

Tabla 3.7. Polaridades asignadas por Senti-RST y la votación.

Par	WNA	WND	SC	SUMO	WN	ALLRec	Voting
T	Neu	Pos	Pos	Pos	Neg	Pos	Pos
H	Neu	Neu	Neu	Neg	Pos	Neg	Neu

Continuando con el análisis, se puede ver también en la Tabla 3.6 que la técnica de votación recibe el mayor número de pares con relación de polaridad desigual y non-entailment (*UPNE* =0.562). También obtiene un mayor número de éxitos (véase la Figura 3.5). Sin embargo, WNA presenta la mejor relación comparando número de éxitos con el número de fracasos, obteniendo una mayor precisión (0.662). La técnica de votación, supera a WNA en alrededor de 2376 éxitos encontrados y muestran una buena precisión (0.646).

Por otra parte, es bueno aclarar que el conjunto de datos número tres (correspondiente a ENG2_NP_R) obtiene valores de precisión de uno, esto es debido a que en este corpus solo existen pares de frases con relación de non-entailment y en todos los casos se ha identificado correctamente.

Un resumen de la precisión promedio de los 13 conjuntos de datos se muestra en la Tabla 3.8. Solo en los conjuntos de datos cuatro y cinco, en los que WND y SUMO obtienen mejores valores; WNA obtiene los valores máximos de AP. Sin embargo, ya se ha comentado el problema de la baja recuperación de este recurso, debido a que no está lo suficientemente poblado.

Tabla 3.8. Precisión promedio para los 13 conjuntos de datos analizados.

	WNA	WND	SC	SUMO	WN	ALLRec	Voting
1	0.77	0.64	0.71	0.71	0.67	0.71	0.70
2	0.79	0.65	0.66	0.59	0.74	0.71	0.67
3	1.00	1.00	1.00	1.00	1.00	1.00	1.00
4	0.68	0.77	0.52	0.49	0.55	0.64	0.59
5	0.79	0.85	0.73	0.92	0.88	0.81	0.88
6	0.78	0.64	0.69	0.69	0.67	0.71	0.67
7	0.80	0.63	0.63	0.62	0.63	0.66	0.60
8	0.78	0.61	0.67	0.63	0.62	0.70	0.65

9	0.71	0.64	0.70	0.64	0.66	0.70	0.66
10	0.70	0.64	0.69	0.64	0.66	0.70	0.66
11	0.75	0.60	0.64	0.62	0.66	0.69	0.66
12	0.70	0.54	0.62	0.61	0.60	0.64	0.64
13	0.74	0.60	0.66	0.65	0.64	0.67	0.64

De acuerdo al juicio de este autor, los alentadores resultados obtenidos por la técnica de votación (véase la Figura 3.5 y en la Tabla 3.6) es debido a que es capaz de tomar una mejor decisión del análisis del veredicto de todos los recursos en conjunto. Como se puede ver en la Tabla 3.8, todos los recursos tienen comportamiento muy similar, pero además también se complementan. Esta afirmación se puede corroborar observando el comportamiento de ALLRec -excluyendo WNA- sus resultados son mejores, o al menos iguales, que cada uno de los recursos aislados.

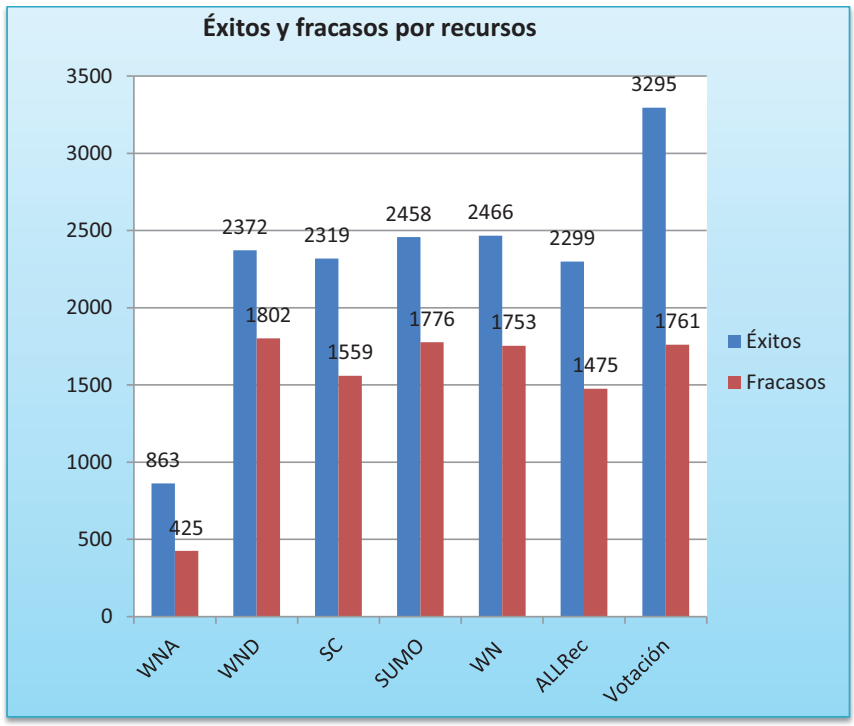


Figura 3.5. Fallos y éxitos de cada recurso.

Hasta donde se tiene conocimiento, este trabajo representa el primer intento de descubrir la influencia de la polaridad sentimental en el Reconocimiento de Implicación Textual.

En los a experimento llevados a cabo con varios conjuntos de datos para la tarea del reconocimiento de la implicación, se ha mostrado que la polaridad sentimental puede ser un factor a considerar en el

intento por reconocer la implicación; sobre todo si se tiene en cuenta que solo con este parámetro, se obtienen valores de precisión considerables.

También, estos experimentos corroboran que recursos como WNA, SUMO y el método ALLRec, parecen ser importante para esta tarea. Es preciso una la taxonomía de WNA enriquecida con una buena cantidad de afectos, porque este es su punto débil.

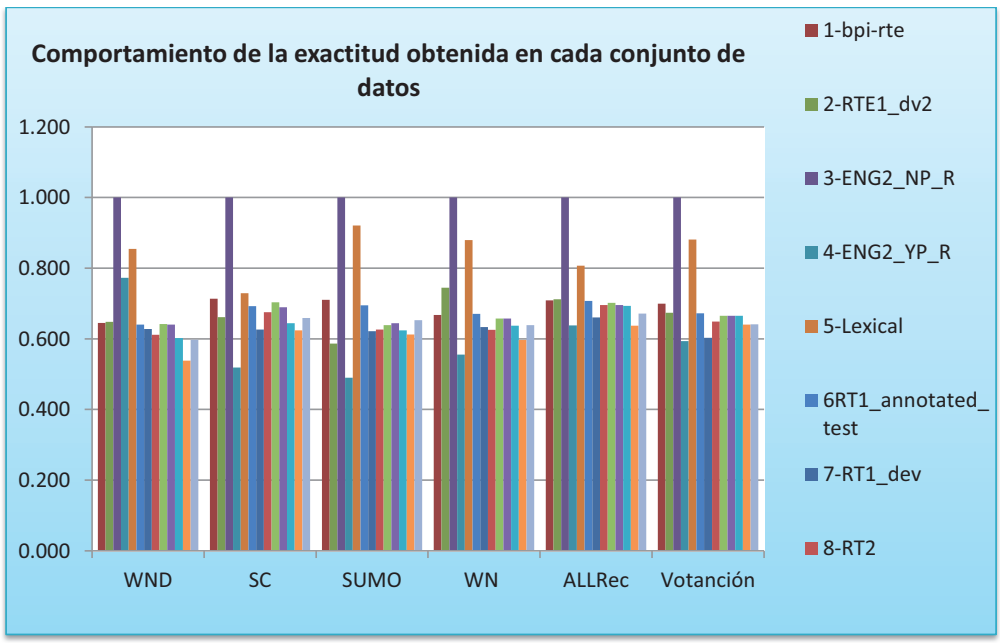


Figura 3.6. Comportamiento de la exactitud obtenida con cada conjunto de datos.

Como se puede ver la exactitud obtenida (ver Figura 3.6), para poder asumir la hipótesis de partida como cierta, es bastante alta para la mayoría de los conjuntos de datos. El valor más bajo reportado (0.49), se obtiene para el recurso SUMO en el conjunto de datos `4-ENG2_YP_R`. Sin embargo, para este mismo conjunto de datos -con WND- se obtiene un valor de exactitud de 0.773.

El número de casos que demuestran la hipótesis contraria, podría deberse a aquellos en los que ocurre una relación desigual entre los pares Positivo/Neutro o Negativo/Neutro, que fueron tomados como fallos al existir una relación de entailment.

Esta es, tal vez, una decisión demasiado rigurosa, pues podría suceder que -en realidad- una frase con polaridad neutra esté en relación de entailment con otra de polaridad positiva o negativa. Tal vez un análisis más flexible, en el que se verifique la posibilidad de que pares con relación de entailment, pudieran tener cualquier combinación con neutro, podría haber mejorado la precisión.

Por ejemplo haciendo esta adaptación en el conjunto de datos perteneciente al RTE-1, bpi-rte.txt, se obtienen las siguientes mejoras:

Tabla 3.9. Antes de modificar relaciones NEG/NEU y POS/NEU como posibles con relación de entailment.

Precisión en **non-entailment**	0.384	Exactitud	Precisión promedio
Precisión en **entailment**	0.584	0.699421965	0.676996557
Aciertos	121		
Fallos	52		

Tabla 3.10. Después de modificar relaciones NEG/NEU y POS/NEU como posibles con relación de entailment.

Precisión en **non-entailment**	0.40573	Exactitud	Precisión promedio
Precisión en **entailment**	0.78495	0.858156028	0.839531598
Aciertos	121		
Fallos	20		

Lo mismo va a ocurrir con los 12 conjuntos de datos restantes pues este tipo de relación se presenta con mucha frecuencia.

A juicio de este autor, a partir de los resultados obtenidos en este experimento, se debería considerar la posibilidad de incluir la polaridad sentimental como un atributo más en el reconocimiento de la implicación textual.

Como observación final, aunque en general las diferencias no son extremadamente grandes, se puede decir que algunos recursos son mejores que otros para detectar una relación de implicación partiendo de la polaridad, por lo que es necesario abordar este problema desde un punto de vista multidimensional, para obtener con ello una complementación general.

Como se vio en el capítulo anterior, el RIT ha sido tratado por algunos investigadores usando alineamiento entre las frases analizadas. A continuación se describe en detalle el método de alineamiento diseñado en esta investigación para el reconocimiento de la implicación textual.

3.3 El alineamiento Léxico-Semántico en la determinación de similitud

La propuesta de Alineamiento Léxico-Semántico (ALS) se divide en tres etapas principales como se ilustra en la Figura 3.7. La primera etapa se encarga de tokenizar el corpus en análisis, lematizar y finalmente extraer las categorías gramaticales de cada una de las frases contenidas en dicho corpus.

Más adelante, en una segunda etapa se calculan las distancias léxicas y medidas de similitud entre las frases; se realiza el alineamiento léxico con un ligero aporte semántico, de donde se obtienen los diferentes atributos derivados de este análisis. Por último, se obtiene la polaridad sentimental de cada par de frases. También en esta fase se determinan los dominios a los que pertenecen ambas frases.

La tercera etapa y última, se encarga de obtener el modelo entrenado utilizando para ello la herramienta Weka y un clasificador seleccionado experimentalmente. Este modelo entrenado es el que permite luego poder hacer la clasificación de las nuevas instancias a evaluar. En la Figura 3.7 se muestra el esquema general de este método.

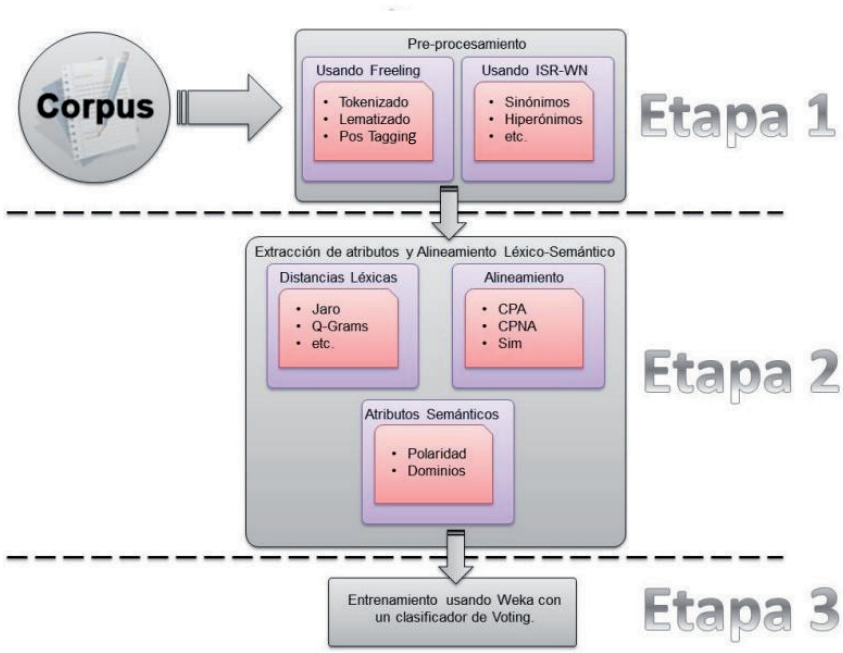


Figura 3.7. Etapas de desarrollo de la propuesta.

En los próximos epígrafes se describe en detalles cada una de las etapas del proceso.

3.3.1 Descripción de la etapa 1

Como se explicó anteriormente, esta etapa es la encargada del pre-procesamientos del corpus. Aquí se realiza, en una primera instancia, el tokenizado de todas las frases, se lematizan y se realiza el POS-tagging a cada una de ellas. Para la realización de estas operaciones se utiliza el recurso Freeling 2.2[52], el que permite obtener cada uno de estos elementos con suficiente precisión.

A continuación, se realiza el proceso de extracción de las características semánticas. En esta etapa solo se dependerá del recurso ISR-WN (Gutiérrez, Fernández et al., 2010a), el que –al tener como núcleo central a WN- permite obtener las relaciones de sinonimia, hiperonimia, e hiponimia entre los pares de frases. También se obtienen otras relaciones que resultan de gran importancia como:

- Relación de causa, ejemplo, encourage (alentar, estimular) causa hope (esperanza)
- Relación de implicación, ejemplo: encourage causa encouragement (estímulo, aliento)
- Relación de la forma Similar a, ejemplo: indigenous (indígena) es similar a native (nativo)
- Grupos verbales, ejemplo: Los grupos verbales de pay (pagar) son: give (dar), sacrifice (sacrificio), etc.
- Formas derivadas relacionadas, ejemplo: las formas derivadas de violence (violencia) son: storm (tormenta), force (fuerza), etc.

También en esta etapa fue implementado un método de comprobación de equivalencias, con el objetivo de no dejar pasar por alto -en el análisis de las frases- algunas de las características con determinada carga semántica, las que influyen en el cálculo de la similitud entre palabras.

Desafortunadamente, estas características no están representadas en ninguno de los recursos utilizados en el proceso de análisis. Para ello, se tienen en cuenta elementos como abreviaturas de países, de monedas, onomatopeyas[53] y gentilicios (ver Tabla 3.11).

Tabla 3.11. Tipos de equivalencias analizadas y sus ejemplos.

Tipo de equivalencia	Valor	Equivalencia
Meses	Jan	January
Días	sun	sunday
Monedas	money	ARS, CUC, USD, etc.
Onomatopeyas	beard	tweet
Países y Capitales	Cuba	CUB
Gentilicios	Cuba	cubano
Origen de Nombres propios	Aaron	Hebreo

[52] http://nlp.lsi.upc.edu/freeling/

[53] Según la Real Academia Española, es la imitación o recreación del sonido de algo en el vocablo que se forma para significarlo o vocablo que imita o recrea el sonido de la cosa o la acción nombrada

Como se ha visto, la Tabla 3.11 ilustra solo algunas de las características tenidas en cuenta en este método. A través de él, se hace posible la obtención de equivalencias tales como: Al hablar de EEUU, se está hablando de USA.

Para este objetivo, este lexicón de equivalencias fue desarrollado manualmente, partiendo de varios lexicones, desarrollados de forma semiautomática a partir de corpus y textos recopilados en internet y otras fuentes de información.

Otro análisis, que también se realiza en esta etapa, es el de la negación. Se consideraron las formas posibles de negar en inglés, ya que este fue el lenguaje con el que se trabajó en la obtención de similitud (ver epígrafe 2.5.1, formas de negación en inglés). Este análisis es de gran importancia debido a que es posible que dos palabras compartan un mismo lema, pero debido a que una de las dos se encontrara negada, no podrían ser consideradas como palabras alineadas.

En este sentido, para detectar si una palabra está negada, es necesario buscar la presencia de una partícula de negación cercana a la palabra en análisis, en ese caso -si existe- se considera entonces como negada. El análisis se extiende a una ventana de dos palabra a la izquierda, pues pudiera darse el caso, por ejemplo: It was the pupils, not the teachers, who went on strike (Fueron los alumnos, no los profesores, quienes se declararon en huelga), donde la palabra teachers debe anotarse como negada por la partícula not, que a su vez se encuentra en la segunda posición anterior a teachers.

Otro análisis, es buscar si la palabra contiene los prefijos de negación de adjetivos y sustantivos, comprobándose que el token en análisis no sea in (en), ni incoming (entrante), que contienen el prefijo de negación in, pero no están formando una negación.

Partiendo del análisis de las formas de negar en inglés, explicado en el epígrafe 2.5.1, es posible buscar en las frases las palabras o tokens que se encuentran negados.

3.3.2 Descripción de la etapa 2:

En esta etapa, el alineamiento se desarrolla extrayendo diferentes atributos léxicos como las son las distancias y medidas de similitud entre los textos de cada una de las frases que componen el corpus. Se usan para ello las bien conocidas medidas de similitud entre cadenas de caracteres como: Needleman-Wunch, Smith-Waterman, Smith-Waterman-Gotoh, Smith-Waterman-Gotoh-Windowed-Affine, Jaro, Jaro-Winkler, Chapman-Length-Deviation, Chapman-Mean-Length, QGrams-Distance, Block-Distance, Cosine-Similarity, Dice-Similarity, Euclidean-Distance, Jaccard-Similarity, Matching-Coefficient, Monge-Elkan y Overlap-Coefficient. Estas medidas fueron explicadas en epígrafe 2.6.1.

Por otra parte, los atributos extraídos a partir de las métricas son almacenados en un vector para su posterior uso.

En particular, a la distancia de Levenshtein -vale aclarar- que se le realiza una modificación para también obtener de ella otro atributo. En esta ocasión, las unidades a comparar no son los caracteres, sino las palabras. A partir de esta modificación se generan tres variantes de esta distancia. A continuación se presentan las formas de comparación empleadas:

- Comparar las palabras por su morfología (LevForma): En esta variante se analiza si existe similitud entre las palabras solo comparando su grafía.
- Comparar las palabras por sus lemas (LevLema): En esta variante se analiza si existe similitud entre las palabras solo comparando por sus lemas.
- Comparar las palabras usando dos veces la distancia de Levenshtein, a la que se le llama Levenshtein Doble (DLED): Se establece un umbral numérico indicando que, si la distancia entre las palabras analizadas es menor o igual a ese umbral, entonces no existe modificación, las palabras son consideradas como iguales. De forma experimental fue comprobado que el umbral de comparación con un valor de dos (2), es el de mejores resultados para esta propuesta (ver detalles en la Tabla 3.12).

Atendiendo a lo antes mencionado, en las dos primeras variantes la idea consiste en usar la distancia de Levenshtein en su forma clásica, una vez para la frase original y la otra trabajando con los lemas. A partir de aquí, se obtienen dos atributos para el entrenamiento. Contrario a lo que se hace en (Tatu, Iles et al., 2006), no se remueven los signos de puntuación, ni las stopwords de las frases. Tampoco se consideran costos diferentes para las operaciones de transformación y se usan todas las operaciones (borrado, inserción y substitución).

Por su parte, la última variante aporta otro nuevo atributo, DLED (ver Tabla 3.12). Para este algoritmo se usa la distancia de Levenshtein (LED) para medir la distancia entre las frases, pero para la comparación entre las palabras se vuelve a usar LED nuevamente. La Tabla 3.12 muestra el pseudo-código la función DLED, la que toma dos cadenas, `s` de longitud `m` y `t` de longitud `n`, para con ellas calcular la distancia entre ellas. En la Tabla 3.12, en la línea 9 se puede ver la diferencia entre el algoritmo LED clásico y DLED. Esta línea para el algoritmo clásico sería `s[i]=t[i]`, sin embargo para DLED se calcula la similitud de las palabra aplicando LED nuevamente. Los valores que estén por debajo de un valor de umbral de dos (2), experimentalmente seleccionado, significa que las palabras son iguales. Como resultado de este algoritmos se obtienen un nuevo atributo para el entrenamiento.

Tabla 3.12. Pseudo-código del algoritmo DLED.

```
int DLevDist (string s[1..m],string t[1..n])

1. declare int d[0..m, 0..n]
2. for i from 0 to m
3. d[i, 0]:= i
4. for j from 0 to n
5. d[0, j]:= j
6. for i from 1 to m
7. for j from 1 to n {
8. if (LevDist(s[i],t[i])<=2)
9. then cost:= 0
10.      else cost:= 1
11.        d[i, j] := minimum(
12.        d[i-1, j] + 1, // deletion
13.        d[i, j-1] + 1, // insertion
14.        d[i-1, j-1] + cost) // substitution
15.      }
16. return d[m,n]
```

Además de las anteriores, otras distancias consideradas para extraer atributos de la relación entre las frases son:

Normalización de LevenshteinForma: Esta medida viene dada por la ecuación 3.14.

$$NormLevenForma = \frac{t - (h - LevForma)}{t} \quad 3.14$$

Donde:

h: es la cantidad de palabras de la hipótesis, t es la cantidad de palabras del texto.

$LevForma$: como ya se vio es la distancia de Levenshtein entre el par texto-hipótesis.

La siguiente distancia es la Normalización de LevenshteinLema: para calcular esta medida se utiliza la ecuación 3.15

$$NormLevenLema = \frac{t - (h - LevLema)}{t} \quad 3.15$$

Donde:

$LevLema$: como ya se mencionó es la distancia entre el par texto-hipótesis, pero considerando los lemas de las palabras.

La ecuación 3.15 normaliza la distancia $LevLema$ obteniendo valores en el intervalo [0,1].

Finalmente, otra distancia que se utiliza es la distancia extendida, DEx (ver detalles en el epígrafe 3.1). Como se puede ver en este epígrafe, el algoritmo presentado es una extensión del algoritmo de Levenshtein. Además de una distancia, esta medida ofrece la subsecuencia común más larga y otros elementos importantes para la determinación de la similitud en una solo iteración.

Otro elemento, que cobra gran importante en esta etapa es la obtención de los atributos semánticos. Estos atributos se extraen a partir del análisis realizado al par de frases, intentando comparar significados, relaciones y sentimientos.

Uno de estos atributos, lo constituye la relación de Polaridad Sentimental que posee el par de frases analizadas. Para el cálculo de las polaridades de las frases fue utilizado el recurso ISR-WN (Gutiérrez, Fernández et al., 2010a), el que brinda una integración de los recursos ya mencionados anteriormente como: WND, SUMO, WNA, WN, SM y la combinación de todos ellos a la que se le ha llamado AllRec. Cada uno de estos recursos en combinación con Senti-WordNet como base, permiten obtener la polaridad para cada frase a través el método Senti-RST (Gutiérrez, Vázquez et al., 2011). El problema radica en escoger la polaridad resultante a partir de las polaridades de los seis recursos. Las polaridades son obtenidas de la forma siguiente: uno (1) (Positiva), cero (0) (Neutra), menos uno (-1) (Negativa) y dos (2) (Desconocida).

Para eliminar los posibles empates en los valores de la polaridades ofrecida por cada recurso, se fijan dos de ellos como pivotes de desempate, experimentalmente para esta propuesta, fueron escogidos como primer pivote de desempate el recurso SC y como segundo pivote el recurso WN, se tomó esta decisión partiendo de los resultados obtenidos en (Fernández, Gutiérrez et al., 2012b). Estos recursos obtienen buenos valores de exactitud y precisión promedio en la determinación de la implicación textual (ver Tabla 3.6). En el cálculo de la polaridad final se tienen las siguientes posibilidades:

- Que exista un predominio numérico en cuanto a una de las polaridades, asignándole en este caso esa polaridad a la frase (ver Figura 3.8).

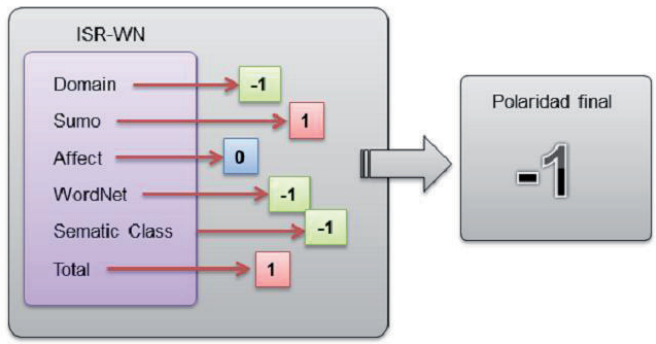


Figura 3.8. Mayoría de recursos con polaridad negativa (-1).

Si existe un empate en número entre las polaridades predominantes, entonces se escoge la polaridad del grupo donde está el primer pivote.

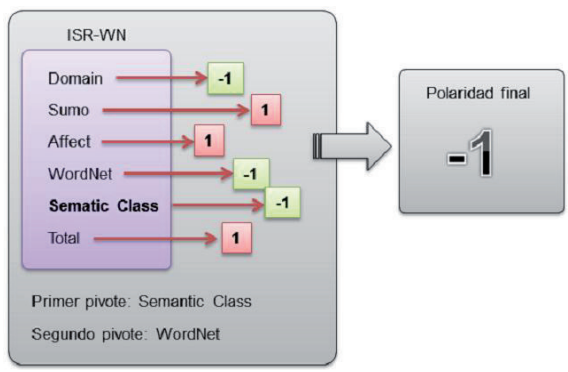


Figura 3.9. Empate entre polaridades, decide el pivote principal (uno).

En la Figura 3.9 se observa un empate entre las polaridades 1 y -1, pero como el primer pivote principal (Semantic Class) tiene polaridad -1, el resultado final de la polaridad es -1.

- En caso de existir un empate entre las polaridades dominantes en cantidad y el primer pivote no estar entre las que provocaron el empate, se acude al segundo pivote, que será el que decidirá (ver Figura 3.10).

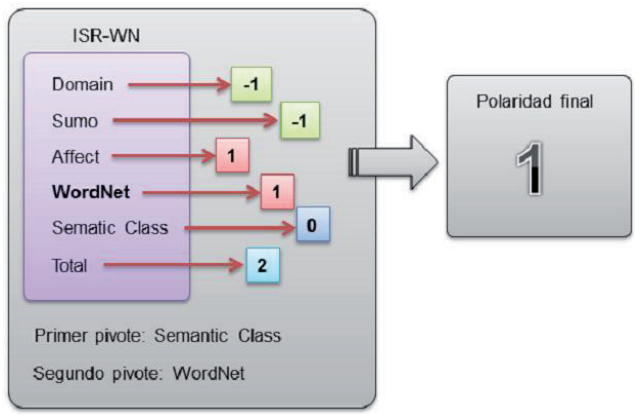


Figura 3.10. Empate entre polaridades, decide el pivote secundario (dos).

En la Figura 3.10 se puede observar un empate en las polaridades 1 y -1, pero el primer pivote no se encuentra entre los recursos que generaron los valores del empate, ofreciendo una polaridad neutra (0). Se analiza entonces el segundo pivote de desempate y la polaridad final es decidida por este pivote.

Hasta este punto se han obtenido dos atributos que son guardados en un vector y representan dos valores con la polaridad del Texto y la polaridad de la Hipótesis.

También, se extrae un atributo que relaciona las polaridades de ambas frases (ver ecuación 3.16).

$$DistP = PolTexto + PolHipótesis \qquad 3.16$$

Donde:

$PolTexto$: es la polaridad del texto.

$PolHipótesis$: es la polaridad de la hipótesis.

Otro atributo, es el obtenido del dominio al que pertenecen el par de frases. Entiéndase por dominio de la frase a la selección de unos de los dominios aportados por el recurso WordNet Domain, por ejemplo: The man went to the hospital because he had a headache (El hombre fue al hospital pues tenía dolor de cabeza). Esta frase pertenece al dominio de la medicina, porque involucra términos tales como hospital y headache, las demás palabra se subordina a este dominio ya que no aportan ninguno nuevo. Para obtener los dominios de cada una de las frases también se utiliza el recurso ISR-WN. La distancia entre dominios está dada por la cantidad de nodos a recorrer para llegar desde un dominio a otro en el árbol de dominios.

Por último, en esta etapa se realiza un alineamiento léxico-semántico basado en tokens que unifica las características léxica y semánticas de esta comparación entre las frases.

El primer paso del alineamiento consiste alinear los tokens por la coincidencia entre sus lemas, luego de alineados por sus lemas se procede a verificar sus categorías gramaticales. En caso de no existir coincidencia entre estas características, entonces no se pueden alinear. De otro modo, si existe coincidencia entre estos aspectos, se procede al análisis de la negación, si existen diferencias entre las negaciones de las palabras, no se alinearán.

Hasta este momento, en este tipo alineamiento solo se han tratado aspectos léxicos. En caso de existir coincidencia entre estas características léxicas, se alinean las palabras de ambas frases. De otro modo, si no existen alineamiento se procede a contrastar estos tokens con el análisis semántico basado en relaciones de WordNet en lo adelante solo (AS-WN).

El primer paso del AS-WN consiste en la comparación de las relaciones de WordNet como: la sinonimia, hiperonimia, hiponimia, relaciones de causa e implicación, similar a, grupos verbales y formas derivadas relacionadas.

De esta forma, en el caso de la sinonimia, grupos verbales, similar a, así como las formas derivadas relacionadas, se desarrolla la comparación de forma bidireccional, es decir, si una palabra del Texto (T) está contenida en la lista de sinónimos de otra palabra de la Hipótesis (H), o viceversa, entonces se pueden alinear.

Así también, para el caso de la hiperonimia debe ocurrir que una palabra de H contenga en su lista de hiperónimos otra palabra de T, mientras que para la hiponimia debe ocurrir lo contrario.

Además, en cuanto a las relaciones de causa e implicación, se debe dar el caso que una palabra de T contenga en su lista de relaciones de causa o implicación otra palabra de H.

Continuando con este tratamiento, otro tipo de relación que se analiza de forma bidireccional es la comprobación de equivalencias, explicada en la etapa 1.

Finalmente, como resultado del alineamiento son calculados los atributos siguientes:

- (CPA) representa la cantidad de palabras alineadas.
- (CPNA) representa la cantidad de palabras no alineadas (ver ecuación 3.17).

$$CPNA = CPFL - CPA \quad 3.17$$

Donde: $CPFL$ es la cantidad de palabras de la frase más larga.

La similitud final de alineamiento será:

$$SIM = \frac{CPA}{CPFC} \quad 3.18$$

Donde:

$CPFC$: es la cantidad de palabras de la frase más corta. De esta forma si la cantidad de palabras alineadas es igual a la cantidad de palabras de la frase más corta, entonces la similitud de alineamiento es igual a uno (1), ya que la frase más corta estará implicada en la frase mayor.

Hasta éste punto, se tienen una serie de atributos léxicos y atributos semánticos, resultantes y no resultantes del alineamiento. En el ANEXO 7 se hace una descripción de cada uno de los atributos. Éstos serán utilizados en la tercera etapa para el proceso de entrenamiento y aprendizaje automático.

En resumen, del proceso de alineamiento se extraen diferentes atributos que ayudan a mejorar los resultados de sistema de aprendizaje automático. La Tabla 3.13 muestra el grupo de atributos léxicos y semánticos (basados en WordNet). A este grupo de atributos se le llama grupo F1. Cada uno de ellos ha sido designado con un nombre compuesto por un prefijo, un guión y su sufijo. La Tabla 3.14 describe el significado de cada uno de los prefijos. La Tabla 3.15 muestra el significado de los sufijos.

Tabla 3.13. F1. Grupo de atributos léxicos y semánticos.

CPA_FCG, CPNA_FCG, SIM_FCG, CPA_LCG, CPNA_LCG, SIM_LCG, CPA_FCGR, CPNA_FCGR, SIM_FCGR, CPA_LCGR, CPNA_LCGR, SIM_LCGR

Tabla 3.14. Significado de cada prefijo.

CPA	Número de palabras alineadas
CPNA	Número de palabras no alineadas
SIM	Similitud

Tabla 3.15. Sufijo que describe cada tipo de alineamiento.

Atributo	**Comparar palabra por:**
FCG	Morfología y Categoría Gramatical
LCG	Lema y Categoría Gramatical
FCGR	Morfología, Categoría Gramatical y relación de WordNet
LCGR	Lema, Categoría Gramatical y relación de WordNet

De esta forma el atributo CPA_FCG significa: número de palabra alineadas comparándolas por su morfología y su categoría gramatical. A continuación se muestra en la Tabla 3.16 otro grupo de atributos nombrado F2, en el que se utilizan varias medidas.

Tabla 3.16. F2. Medidas del alineamiento léxico.

Medida	Descripción
LevForma	Distancia de Levenshtein entre dos frases comparando por la morfología
LevLema	Lo mismo que el anterior, pero ahora comparando por los lemas
LevDoble	Igual al anterior, pero volviendo a comparar con Levenshtein y aceptando las palabras si el valor de comparación es ≤ 2.
DEx	Distancia Extendida
NormLevF, NormLevL	Forma Normalizada de LevForma y LevLema.

Las Tabla 3.17 muestra el grupo de atributos correspondientes a las diferentes medidas de similitud utilizadas como atributos. Esta medidas fueron obtenidas usando SimMetric library. A este grupo se le ha llamado F3.

Tabla 3.17. F3. Medidas léxicas obtenidas con SimMetrics Library.

NWunch, SWaterman, SWGotoh, SWGAffine, Jaro, JaroW, CLDeviation, CMLength, QGramsD, BlockD, CosineS, DiceS, EuclideanD, JaccardS, MaCoef, MongeElkan, OverlapCoef.

Por último, en la Tabla 3.18 se encuentra el grupo de atributos correspondientes al alineamiento de las frases usando todos contra todos, con diferentes características. Este grupo es denominado F4.

Tabla 3.18. F4. Diferentes alineamientos comparando todos contra todos para ambas frases.

AxAQGD_L	Todos contra todos aplicando QGramsD y comparando por lemas
AxAQGD_F	Lo mismo que el anterior, pero comparando por la morfología de las palabras
AxAQGD_LF	Igual al anterior, pero comparando por la morfología y los lemas
AxALev_LF	Todos contra todos aplicando Levenshtein comparando por lemas y la morfología de las palabras
AxA_Stems	Igual al anterior, pero comparando por el stem

3.3.3 Descripción de la Etapa 3

En esta nueva etapa, una vez obtenidos los valores descritos en la segunda etapa, se procede a la obtención del modelo, resultante de un entrenamiento usando un sistema de aprendizaje automático. Para el desarrollo de este aprendizaje se usó la herramienta Weka. Con ella, se realizaron pruebas con gran cantidad de clasificadores en búsqueda de los mejores resultados.

Finalmente, se decide -de forma experimental- seleccionar un sistema de validación cruzada con paquetes de diez (10) (Ten fold cross validation) y un clasificador de votación compuesto por las siguientes técnicas: Bootstrap Aggregating (Bagging) (Breiman, 1996), usando M5P[54], Bagging usando REPTree (Regression Trees Induction Algorithm) (Witten y Frank, 2005), Random SubSpace (Ho, 1998), usando REPTree y MP5.

[54] Implementación de Weka para el algoritmo M5 Wang, Y. and I. H. Witten (1997). Inducing model trees for continuous classes. Poster Papers of the 9th European Conference on Machine Learning (ECML 97), Prague, Czech Republic..

3.3.4 Resultados y análisis

Para la puesta en funcionamiento de este método se implemento un prototipo (Chávez y Fernández, 2012), el que permite realizar los experimentos con gran comodidad.

Con este prototipo, se realizan tres pruebas para validar la influencia de los atributos seleccionados. Como se puede ver en la Tabla 3.19, en la prueba 1, al utilizar los atributos del grupo F1 (Alineamientos léxicas y semánticos), se obtiene un valor de correlación de 0.7534 (usando los corpus de SemEval 2012).

Por otra parte, en el prueba 2 se usan las medidas léxicas, pero con algún soporte semántico (grupos F1) y se obtiene un valor de correlación de 0.7549. Al unir todos los atributos, en la prueba 3, se obtienen una correlación de 0.7987, más de un 4% de diferencia. Lo que demuestra que es necesario atacar el problema de la similitud desde un punto de vista multidimensional. Por otra parte, se ha podido comprobar que, tanto las medidas léxicas como los alineamientos hacen un aporte importante a la determinación de la similitud entre frases.

También, se puede apreciar en los resultados reflejados en el ANEXO 10, que los valores para el coeficiente de correlación de Pearson de cada atributo, utilizando el conjunto de entrenamiento de SemEval-2012, arroja que hay varios atributos que ejercen gran influencia. En este anexo se puede ver que los atributos DEx, CPA, QGramsD, CMLength, BlockD, SIM, tienen un coeficiente por encima de un 50%, mientras que atributos como OverlapCoef, MongeElkan, DiceS, MaCoef, CosineS, JaccardS, DLED, LED, NormDLE, NormLED, CLDeviation y EuclideanD tienen una influencia significativa, pues su correlación está alrededor de un 40%.

Tabla 3.19. Influencia de los atributos en la determinación de la similitud.

Atributos	**Correlación obtenida usando los conjuntos de entrenamientos de SemEval-2013**		
	Prueba 1	Prueba 2	Prueba 3
F1. Grupo de atributos léxicos y semánticos		0.7549	**0.7987**
F2. Medidas del alineamiento léxico			
F3. Medidas léxicas obtenidas con SimMetrics Library	0.7534		
F4. Diferentes alineamientos todos contra todos			

Nota: las celdas en gris indican que los atributos no fueron tomados en consideración.

Los resultados de este método de alineamiento se prueban en las competiciones del SemEval de los años 2012 y 2013. En la Tabla 3.20, se muestran los resultados obtenidos en SemEval-2012, para los cinco conjuntos de prueba de esta competición (MSRpar, MSRvid, SMT-eur, On-WN, SMT-news).

MSpar[55]: MSR-Paraphrase, Microsoft Research Paraphrase Corpus con 750 pares de frases.

MSRvid[56]: MSR-Video, Microsoft Research Video Description Corpus con 750 pares de frases.

SMT-eur[57]: SMTeuroparl, WMT2008 development dataset (Europarl section) con 459 pares de frases.

On-WN: Pares de frases donde la primera viene de Ontonotes[58] y la segunda de definiciones de WordNet con 750 pares de frases.

SMT-news: Pares de oraciones conversacionales de noticias de WMT con 399 pares de frases.

Tabla 3.20. Resultados oficiales de SemEval 2012 para el alineamiento léxico-semántico.

ALS (*SEM 2012)	**MSRpar**	**MSRvid**	**SMT-eur**	**On-WN**	**SMT-news**
RUN2	0.6022	0.7709	0.4435	0.4327	0.4264

De los 90 participantes que se presentaron en SemEval 2012, esta variante se posicionó en los puestos 26, 46 y 49 en los diferentes rankings que se implementaron para esta competición. Comparada con la variante mejor posicionada obtiene resultados aceptables.

Un año después, esta variante se prueba nuevamente en la competición del SemEval-2013. La Tabla 3.21 muestra los resultados alcanzados por el ALS para esta competición. En esta ocasión se logra alcanzar las posiciones 55 39 50 28 y 44 respectivamente en los cuatro corpus de esta competición.

En general los resultados de todos los participantes de esta competición no fueron buenos, si se tienen en cuenta el mejor equipo obtienen 0.7642 para el corpus Headlines, 0.7529 para OnWN, 0.5818 para FNWN, 0.3804 para SMT y finalmente solo 0.6181 para Mean. Más adelante se explican las causas a las que este investigador acredita los malos resultados obtenidos por esta variante.

Tabla 3.21. Resultados en la competición de SemEval 2013 del ALS.

ALS (*SEM 2013)	**Headlines**	**OnWN**	**FNWN**	**SMT**	**Mean**
RUN2	0.6168	0.5557	0.3045	0.3407	0.4833

Leyenda:

Headlineas: Titulares de noticias extraídos de varios medios de comunicación europeos mediante los RSS.
OnWN: Mapeos de recursos léxicos OnWN. Las frases son las definiciones de sentido de WordNet y OntoNotes.
FNWN: Las frases son definiciones de sentidos de WordNet y FrameNet.
SMT: Proviene del DARPA GALE HTER y HyTER. Una frase es una salida de MT y la otra una traducción de referencia donde la referencia se genera con la edición posterior por humanos.
Mean: Media ponderada de los cuatro conjuntos de datos, en los que el peso depende de la cantidad de pares en el conjunto de datos.

[55] http://research.microsoft.com/en-us/downloads/607d14d9-20cd-47e3-85bc-a2f65cd28042/
[56] http://research.microsoft.com/en-us/downloads/38cf15fd-b8df-477e-a4e4-a4680caa75af/
[57] http://www.statmt.org/wmt08/shared-evaluation-task.html
[58] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2011T03

3.4 El alineamiento Semántico

En este epígrafe se describe la forma de capturar la similitud entre dos frases utilizando un alineamiento con un mayor aporte semántico. A diferencia del alineamiento visto en el epígrafe anterior, en este caso se utilizan elementos léxicos en menor cuantía que el alineamiento ALS. Este método se nombra Alineamiento Semántico (AS). Ya se ha visto anteriormente que la similitud entre frases es una puntuación de confianza que refleja la relación entre los significados de dos cadenas de caracteres, por lo general consistentes de varias palabras o acrónimos.

En este sentido, el objetivo principal en este tipo de alineamiento es determinar la similitud entre las frases utilizando un mayor aporte semántico. Por ello, solo se utilizará el aporte léxico que proporcionan la distancia de Levenshtein y la métrica QGrams Distances. Otro elemento importante en este caso es la aplicación del método Húngaro o algoritmo de Kuhn-Munkres (Kuhn, 1955), este algoritmo de optimización permite resolver el problema de asignación de los sentidos de las palabras para cada frases. Este método de alineamiento debe satisfacer los siguientes requisitos:

- Un reflejo real de similitud léxica: las frases con menos diferencias son reconocidas como similares. En partículas, un solapamiento significativo debe apuntar a un alto nivel de similitud entre las frases.
- Solidez a los cambios de orden de las palabras (simetría): dos cadenas que contienen las mismas palabras, pero en un orden diferente, deben ser reconocidas como análogas. Por otro lado, si una cadena es sólo un anagrama aleatorio de los caracteres contenidos en la otra, entonces debería (por lo general) ser reconocido como diferente.
- Independencia del lenguaje: el algoritmo debe trabajar no solo para el inglés, sino también en otros lenguajes.

Este método de alineación obtiene el cálculo de la similitud semántica entre dos frases, a partir de un análisis basado en los relacionamientos en el árbol semántico que ofrece ISR-WN.

3.4.1 Descripción de la etapa 1

En primer lugar, al igual que en el método de ALS antes mencionado, las dos frases son pre-procesadas con Freeling y las palabras se clasifican en función de sus categorías gramaticales (sustantivo, verbo, adjetivo y adverbio).

Luego, se toma el 30% entre los sentidos más probables de cada palabra y se tratan como un grupo. La distancia entre los dos grupos será el valor mínimo, aportado por el par de sentidos más cercano que pertenezca al grupo. Por ejemplo:

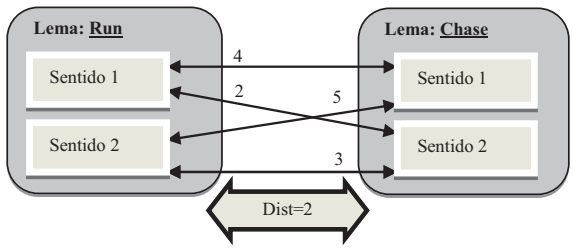


Figura 3.11. Distancia mínima entre las palabra Run (correr) y Chase (persecución).

En el ejemplo de la Figura 3.11 se obtiene una distancia de dos (Dist=2) para el par Run - Chase, por lo tanto este par tiene costo mínimo dos (cost=2), ya que la mínima distancia entre los sentidos de estas palabras es igual a dos. Para los sustantivos que no sean encontrados en WordNet, como nombres propios o nombres comunes, la distancia se calcula de forma diferente, en este caso se utiliza la distancia de Levenshtein. Véase el siguiente ejemplo:

Se toma el par 99 del corpus MSRvid (del conjunto de entrenamiento de SemEval 2012), al que se le han hecho pequeñas transformaciones para un mejor entendimiento del método.

El par original es:

A: A polar bear is running towards a group of walruses.

B: A polar bear is chasing a group of walruses.

Se ha transformado en:

A_1: A polar bear runs towards a group of cats.

B_1: A wale chases a group of dogs.

Luego, usando el algoritmo mostrado en el ejemplo de la Figura 3.11, se crea una matriz con las distancias entre todos los grupos de ambas frases (ver la Tabla 3.22). Nótese que en las celdas de intersección entre dos grupos estará el costo mínimo del par de sentidos que se relacionan con menor distancia.

Tabla 3.22. Matriz con la distancia entre los grupos

Frases	polar	bear	runs	towards	group	cats
wale	Dist:=3	Dist:=2	Dist:=3	Dist:=5		Dist:=2
chases	Dist:=4	Dist:=3	Dist:=2	Dist:=4		Dist:=3
group					Dist:=0	
dogs	Dist:=3	Dist:=1	Dist:=4	Dist:=4		Dist:=1

Usando el algoritmo Húngaro (Kuhn, 1955) para la asignación del costo mínimo, cada grupo de la frase más pequeña es contrastado con los elementos de la frase mayor y el resto es marcado como palabras no alineadas. De esta forma, el recorriendo será por filas determinando la distancia del grupo de la frase menor con todos los grupos de la frase mayor. Al finalizar, se selecciona como emparejados los grupos de menor distancia (en la tablas marcados en azul claro). Al marcar un emparejamiento entre un grupo de la frase menor con uno de la mayor, ese grupo de la frase mayor sale de la búsqueda. De esta forma, en la siguiente iteración no sería necesario volver a comparar ese elemento.

En el ejemplo previo, la palabra toward y polar son palabras que no fueron alineadas (columnas marcadas en naranja claro), así que el número de palabras no alineadas es dos (2). Existe solo un macheo perfecto (las palabras group – group, celda marcada en verde claro) las que se emparejan con costo cero (cost=0). La distancia total de macheo óptimo es cinco (5) (suma de las celdas marcadas en azul claro). La longitud de la frase más corta es cuatro (4) (número total de lemas de la frase más corta). La Tabla 3.23 muestra los atributos extraídos de las frases analizadas.

Tabla 3.23. Atributos extraídos del las frases analizadas.

Número de coincidencias exactas (Same)	**Distancia total de macheo óptimo (Cost)**	**Número de palabras no alineadas (Dif)**	**Número de lemas de la frase más corta (Min)**
1	5	2	4

Los cuatro atributos son los siguientes: Número de coincidencias exactas (Same), Distancia total de macheo óptimo (Cost), Número de palabras que no se emparejan (Dif), Número de lemas de la frase más corta (Min).

Se obtienen varios atributos de este alineamiento. Cuatro atributos de la frase completa, 16 atributos más en grupos de cuatro considerando solo los verbos, solo los sustantivos, solo adjetivos y solo los adverbios. La Tabla 3.24 muestra el análisis hecho para los sustantivos.

Tabla 3.24. Cálculo de la distancia solo para sustantivos.

Sustantivos	bear	group	cats
wale	Dist := 2		Dist := 2
group		Dist := 0	
dogs	Dist := 1		Dist := 1

Tabla 3.25. Atributos extraídos del análisis de los sustantivos.

Número exacto de coincidencias (SameS)	Distancia total del macheo óptimo (CostS)	Número de palabras no alineadas (DifS)
1	3	0

Como resultado final se obtienen 20 atributos de este método de alineamiento. Para cada categoría gramatical los atributos se representan adicionándoles una N para los sustantivos, una V apara los verbos, una A para los adjetivos y una R para los adverbios.

3.4.2 Descripción de la etapa 2

En este momento, solo resta la etapa de entrenamiento. Para la fase de entrenamiento, al igual que en el ALS, se usa el framework para aprendizaje supervisado Weka. Para el entrenamiento se usa una validación cruzada de diez pliegues (ten fold cross validation), con un clasificador de votación compuesto con los mismos clasificadores que se utilizan para el entrenamiento del método de alineamiento descrito en el epígrafe 1.1, Bagging con M5P, Bagging con REPTree, Random SubSpace con REPTree y MP5. Este método se prueba en la competición SemEval 20102. Para ello, se incluyen como conjunto de entrenamiento todos los ofrecidos por la competición (MSRpar, MSRvid y SMTeuroparl).

3.4.3 Resultados y análisis del Alineamiento Semántico en SemEval-2012

Los resultados obtenidos por esta variante de alineamiento, en cada corpus, para la competición de SemEval 2012 se pueden ver en la Tabla 3.26.

Tabla 3.26. Resultados oficiales del AS en SemEval-2012.

Variante	MSRpar	MSRvid	SMT-eur	On-WN	SMT-news
AS	0.5269	0.7756	0.4688	0.6539	0.547

En esta competición se emitieron tres medidas para establecer tres rankings diferentes. ALL (Correlación de Pearson con el gold standard para los cinco conjuntos de datos, y el rango correspondiente), ALLnrm (Correlación de Pearson después de las salidas del sistema para cada conjunto de datos, se ajusta al gold standard por mínimos cuadrados, y el rango correspondiente) y Mean (Media ponderada de los cinco conjuntos de datos, en los que el peso depende de la cantidad de pares en el conjunto de datos).

De las 90 variantes que se presentaron el Alineamiento Semántico (RUN3) obtiene los lugares 29, 23 y 16 respectivamente en los tres métodos de evaluación utilizados (ver Tabla 3.27).

Tabla 3.27. Posicionamiento en el ranking de SemEval 2012 del alineamiento semántico.

Variante	ALL	Rank	ALLnrm	RankNrm	Mean	RankMean
AS	0.6529	29	0.8115	23	0.6116	16

Come se pude apreciar este método de alineamiento semántico obtiene buenos resultados si se le compara con la variante mejor posicionada de la competición de SemEval 2012, incluso superándola para el corpus SMT-news. Para más detalle véase la Tabla 3.28.

Tabla 3.28. Comparación con la variante mejor posicionada del ranking se SemEval 2012.

Variante	1	2	3	4	5	6	7	8	9	10	11
A	0.8239	1	0.8579	2	0.6773	1	0.6830	0.8739	0.5280	0.6641	0.4937
AS	0.6529	29	0.8115	23	0.6116	16	0.5269	0.7756	0.4688	0.6539	**0.5470**

Leyenda:

A-baer/task6-UKP-run2_plus_postprocessing_smt_twsi (Bär, Biemann et al., 2012), **AS**-Alineamiento Semántico, **1**-ALL, **2**-Rank, **3**-ALLnrm, **4**-RankNrm, **5**-Mean, **6**-RankMean, finalmente **7**-MSRpar, **8**-MSRvid, **9**-SMT-eur, **10**-OnWN, **11**-SMT-news, son los conjuntos de prueba ofrecidos por la competición.

Un año más tarde, para la competición de SemEval-2013, se realizan algunas modificaciones al método de alineamiento semántico. En primer lugar, se persigue comprobar si reduciendo el aporte léxico el método continúa ofreciendo los mismos resultados. Además, se desea valorar el método de determinación de distancias entre los grupos utilizados en la versión anterior. En este caso el método va a ser substituido por el del sentido más frecuente.

3.4.4 Modificación al alineamiento semántico

Para una segunda versión del método de Alineamiento Semántico (en lo adelante ASM), se modifica -en primera instancia- la cantidad de categorías gramaticales a las que se les realiza el análisis. Se incluyen en esta versión -además de los sustantivos, verbos, adjetivos y adverbios- las preposiciones, conjunciones, pronombres, determinantes, modificadores, dígitos y expresiones temporales.

La distancia entre dos palabras es la distancia -basada en WordNet- del sentido más probable de cada palabra en el par. Contrario a la versión previa, aplicada en SemEval 2012, en la que el sentido es seleccionado después de aplicar el método Húngaro. Tampoco en este caso, se utiliza una distancia alternativa para comparar las palabras que no aparecen en WordNet.

Finalmente, la distancia entre el par de frases es computada de acuerdo a la ecuación 3.19 que se muestra a continuación:

$$d(x,y) = \sum_{i=0}^{i=m} w * r(L[i], L[i+1]); \quad 3.19$$

Donde:

L: es la colección de synsets correspondientes al camino mínimo entre los nodos x y y

m: es la longitud de L restándole uno.

r: es una función que busca la relación que conecta al nodo x y al nodo y a través del sentido más frecuente usando el algoritmo Breadth First Search (BFS) (en español búsqueda primero a lo ancho) (Cormen, Leiserson et al., 2001).

w: es el peso asociado a la relación buscada por r (ver Tabla 3.29). Los valores asignados a los pesos de cada relación se obtienen a través de la experimentación con los corpus de entrenamiento. Finalmente, se escoge la combinación de valores que mejores resultados aportan.

Tabla 3.29. Pesos asociados a las relaciones de WordNet

Relación	**Peso** (w)
Hiperonimia, Hiponimia	2
Miembro_Holonimo, Miembro_Merónimo, Casa, Implicación	5
Similar_A	10
Antonimia	200
Otra relación diferente a Sinonimia	60

Para obtener el resultado final, este proceso tiene que repetirse para los sustantivos, verbos, adjetivos, adverbios, preposiciones, conjunciones, pronombres, determinantes, modificadores, dígitos y expresiones de tiempo. Por el contrario, la matriz de relacionamiento se crea solo con los grupos similares de las frases.

De esta forma, se extraen de este proceso varios atributos con información sobre el par de frases. Tres atributos para cada una de las categorías considerándolas de forma independiente: verbos, sustantivos, adjetivos, adverbios, preposiciones y conjunciones, pronombres, determinantes, modificadores, dígitos y por último expresiones de tiempo. Los atributos son:

- Número de coincidencias exactas
- Distancia total de la coincidencia
- Número de palabras que no coinciden

No todos los atributos son determinados de la misma forma, muchos grupos tienen características particulares en función de sus categorías gramaticales. El grupo de los sustantivos tiene una característica más, que indica si las dos frases tienen el mismo número (singular o plural). Para esta función se toma, como un número final de la frase, el promedio del número de cada sustantivo en la frase.

Así también, para el grupo de adjetivos se añade una característica que indica la distancia entre los sustantivos que estos modifican.

Por su parte, para los verbos se buscan los substantivos antes y después, definiéndose definen dos grupos. Se calcula la distancia para alinear cada grupo con cada par de verbos alineados. Los verbos tienen otra característica que especifica si todos están en el mismo tiempo verbal.

Mientras que, con los adverbios se busca el verbo al que modifican y se calcula su distancia con relación a todos los pares alineados; por contrario, con los determinantes y los adverbios se detecta si alguno de los pares alineados están expresando negaciones (don't o do not).

Por último, se determina si las dos frases tienen la misma acción principal. Para todas estas características se utiliza el recurso Freeling.

Como resultado, finalmente se obtienen 42 atributos de este método de alineamiento. Es importante aclarar que este proceso intenta verificar si cada grupo en la fila tiene un grupo relacionado en las columnas.

Para su comprobación, esta nueva modificación se prueba en la competición del SemEval-2013 obteniéndose los resultados que se muestran a continuación.

3.4.4.1 Resultados y análisis del Alineamiento Semántico en SemEval 2013

Tabla 3.30. Posicionamiento en el ranking de SemEval-2013 del alineamiento semántico

Variante	1	R	2	R	3	R	4	R	5	R
ASM	0.3846	85	0.1342	88	-0.0065	85	0.2736	72	0.2523	87

Leyenda: 1- Headlines, 2- OnWN, 3- FNWN, 4- SMT, 5- Mean.

Se puede apreciar en la Tabla 3.30 que los resultados obtenidos por este método no son buenos, si se comparan con los alcanzados en la versión anterior del SemEval-2012. Aunque, también es cierto que no podría decirse que los resultados generales en esta competición hayan sido los mejores. Contradictoriamente, este método obtiene un mejor desempeño en la fase de entrenamiento, incluso superior al método ALS, sin embargo en la fase de prueba es ampliamente superado.

3.4.4.2 Comparación de resultados de los métodos de alineamiento

Para tener una idea del comportamiento de ambos métodos de alineamiento (ALS y AS), se realiza la comparación de resultados que aparece en la Tabla 3.31. Se puede ver que el método de AS supera al ALS en los resultados obtenidos con los corpus: MSRvid, SNT-eur, On-WN y SMT-news. El AS solo es superado en el corpus MSRpar, precisamente este corpus requiere de menor tratamiento semántico para la identificación de similitud entre sus frases. No ocurre así en los demás corpus, en los que se hace necesario un mayor aporte semántico para detectar la similitud entre las frases.

Tabla 3.31. Comparación de resultados de ambos métodos de alineamiento en la competición SemEval-2012.

Métodos	MSRpar	MSRvid	SMT-eur	On-WN	SMT-news
ALS (*SEM 2012)	0.6022	0.7709	0.4435	0.4327	0.4264
AS (*SEM 2012)	0.5269	**0.7756**	**0.4688**	**0.6539**	**0.547**

Inesperadamente, como se puede ver en la Tabla 3.32, en la competición del SemEval-2013, contrario a lo ocurrido en SemEval-2012, hay un desequilibrio grande a favor del método de ALS.

Tabla 3.32. Comparación de resultados de ambos métodos de alineamiento en la competición SemEval-2013.

Métodos	Headlines	On-WN	FNWN	SMT	Mean
ALS (*SEM 2013)	**0.6168**	**0.5557**	**0.3045**	**0.3407**	**0.4833**
AS (*SEM 2013)	0.3846	0.1342	-0.0065	0.2736	0.2523

Como es conocido, el método AS utiliza un mayor análisis semántico que el método ALS, a partir de esto, debería conseguir mejores resultados que los alcanzados con el corpus FNWN, debido a que este corpus se extrae de FrameNet (Baker, Fillmore et al, 1998), precisamente una red semántica. Haciendo valer aún más la contradicción, FNWN proporciona ejemplos con mayor contenido semántico que léxico.

Es de destacar que, totalmente opuesto a lo sucedido en la fase de prueba, en el entrenamiento el método AS obtiene un coeficiente de correlación de 0.8137, incluyendo todos los corpus de entrenamiento de SemEval-2013. Sin embargo, el ALS obtiene un 0.7976.

Por este motivo, para asegurar la factibilidad de esta modificación, se realizan experimentos con los conjuntos de prueba ofrecidos por SemEval-2012. De esta forma se comprueba si los cambios introducidos afecta el desenvolvimiento del método. Los resultados obtenidos se muestran en la Tabla 3.33.

Tabla 3.33. Resultados de la modificación del métodos AS con los corpus de SemEval-2012.

Variante	MSRpar	MSRvid	SMT-eur	On-WN	SMT-news
ASM	0.3726	0.8162	0.4584	0.5157	0.4225

Como se aprecia, los resultados que arroja la modificación introducida al AS, aunque no tan bajo como los obtenidos en el SemEval-2013, continúan estando por debajo de los alcanzados por el método original, solo se superan para el corpus MSRvid.

A juicio de este autor, una posible consecuencia de los bajos resultados en la competición de SemEval-2013, podría haber estado motivado por determinados problemas en la clasificación de los corpus de evaluación. Por ejemplo, en los corpus FNWN y SMT, según (Agirre, Cer et al., 2013a), la calidad de anotación es FNWN: 69.9% y SMT: 65.8%, lo que indica que pueden existen algunos pares de frases incorrectamente clasificados. Otros factores que también pueden haber afectado son, por ejemplo:

1. En el corpus FNWN hay gran cantidad de entidades acrónimos y gentilicios que no se tienen en cuenta en el método ASM. Sin embargo sí son analizaron en ALS.
2. El corpus FNWN presenta un desbalance con relación a la longitud de las frases.
3. En el corpus OnWN, a criterio de este investigador, muchas frases son mal evaluadas. Por ejemplo, en la línea 7 de este corpus el valor propuesto para la similitud de las frases fue 0.6, sin embargo, ambas frases son semánticamente muy parecidas:
    - The act of lifting something (El acto de levantar algo).
    - The act of climbing something (El acto de subir algo). Usando el sentido upload de climbing.

A juicio de este investigador, el valor 0.6 no es una evaluación correcta para este ejemplo.

Además de los problemas ya mencionados, existen dos elementos que hay que considerar. El primero es la forma diferente en la que se calcula la distancia entres los grupos. Si bien en el método AS, se realizaba a través del método Húngaro, en la modificación ASM se realiza utilizando el sentido más frecuente. Esto no ha sido comprobado, pero es sabido que el cálculo del sentido más frecuente es muy dependiente de los corpus con los cuales se realiza este cálculo. Por lo que este hecho puede haber afectado el resultado al utilizar los corpus del SemEval-2013.

Otro elemento, que en este caso sí ya se ha demostrado en experimentos anteriores, es el hecho de no haber utilizado el aporte léxico de una distancia de edición, en el análisis de las palabras que no se encuentran en WordNet. Para la versión ASM estas palabras se consideran, simplemente, como no alineadas.

3.4.5 Combinación de los métodos de alineamiento AS y ALS.

Basado en los resultados por separado, obtenidos por cada uno de los métodos de alineamiento y analizando la complejidad de los corpus enfrentados, se decide hacer una combinación de los métodos de alineamiento AS y ALS. El objetivo fundamental es utilizar un tratamiento balanceado entre el tratamiento semántico y el léxico, siempre desde una perspectiva multinivel aportada a través de los diferentes recursos aglutinados en la herramienta ISR-WN.

En la Tabla 3.34 se muestran los resultados obtenidos por la combinación ALS+AS en la competición de SemEval-12. Como se puede ver esta variante obtiene mejores resultados que los métodos aislados ALS y AS. La ubicación en el ranking también se mejora, alcanzando los lugares 18, 14 y 15 en los tres rankings de la competición (ver Tabla 3.35).

Tabla 3.34. Resultados oficiales de la combinación ALS+AS en SemEval-2012.

ALS+AS (*SEM 2012)	**MSRpar**	**MSRvid**	**SMT-eur**	**On-WN**	**SMT-news**
RUN1	0.6205	0.8104	0.4325	0.6256	0.4340

Tabla 3.35. Posicionamiento en el ranking de SemEval 2012 de la combinación de alineamientos.

ALS+AS	**ALL**	**Rank**	**ALLnrm**	**RankNrm**	**Mean**	**RankMean**
RUN1	0.7213	18	0.8239	14	0.6158	15

La comprobación de este método con los corpus del SemEval-2013 no obtuvo los resultados esperados (ver Tabla 3.36). Esto, evidentemente estuvo provocado por los bajos resultados ya analizados del método ASM en esta competición.

Tabla 3.36. Resultados oficiales de la combinación ALS+ASM en SemEval 2013.

ALS+AS	**Headlines**	**OnWN**	**FNWN**	**SMT**	**Mean**
RUN1	0.5841	0.4847	0.2917	0.2855	0.4352

En el epígrafe que sigue a continuación se describen los trabajos realizados con otra medición de similitud entre frases, en este caso el reconocimiento de la paráfrasis.

3.5 Reconocimiento de la paráfrasis

Como ya se ha podido ver, específicamente en el epígrafe 2.6.6, el principal objetivo de la paráfrasis es detectar si la verdad de una frase es igual a la de otra. Además, retomando el concepto expuesto en el capítulo II, dichas frases pueden o no tener pequeñas variaciones en su estructura interna y también ser formuladas con palabras diferentes, pero siempre manteniendo el mismo sentido semántico.

Para lograr este objetivo, se implementa un primer método no supervisado (en lo adelante Variante 1), el que recibe como entrada de información un corpus con parejas de frases, específicamente el corpus de paráfrasis de Microsoft (MSRPC) (Dolan, Quirk et al., 2004). En este corpus una frase es considerada como la hipótesis (h) y la otra como el texto (t).

El corpus MSRPC, comprende 5612 pares candidatos de paráfrasis, los que han sido obtenidos de fuentes de noticias de la Web. Los pares de frases han sido marcados por jueces humanos con una clasificación binaria, la que determina: si es paráfrasis (1) o si no, entonces (0). Desafortunadamente, en este corpus los datos se encuentran desbalanceados en una proporción de 65% de ejemplos positivos y un 32% de negativos. Los datos han sido arbitrariamente divididos en un conjunto de entrenamiento con 3775 ejemplos y un conjunto de prueba que contienen 1837 ejemplos. Estos conjuntos de entrenamiento y prueba se usan para todas las aproximaciones que se muestran en este documento.

3.5.1 Arquitectura y descripción de los métodos no supervisados de detección de paráfrasis.

El corpus antes mencionado se somete a un pre-procesamiento, a través de un analizador léxico-sintáctico (Freeling), se extraen las categorías gramaticales, los lemas y se tokenizan las frases. En una primera instancia solo interesan los verbos, así como determinar si existe relación entre ellos usando WordNet. También se utiliza WordNet para descubrir si hay relación entre las frases, basado en las relaciones del tipo sinonimia, hiponimia o hiperonimia. En caso de que no haya similitud entre las frases, la salida del método asume que el par de frases no constituye una paráfrasis.

Por el contrario, si se encuentra alguna relación entre los verbos de ambas frase, se continúan analizando los complementos para finalmente poder decidir si la verdad de la hipótesis es igual a la verdad del texto.

Al final, la salida del analizador léxico sintáctico es un fichero con las frases tokenizadas y cada una de las palabras identificadas con su categoría gramatical y su lema.

Una vez obtenida la transformación con el pre-procesado, el análisis se realiza sobre los verbos de la hipótesis, comparando cada uno para determinar si tienen un análogo en el texto. Si se encuentra una coincidencia del 50%, se pasa a verificar las coincidencias en los complementos, si se obtiene también un 50% de coincidencias, entonces se determina que hay paráfrasis. De no encontrarse coincidencias entre los verbos, se hace una búsqueda por sus respectivas relaciones de WordNet. Se obtiene así una lista por cada relación. Estas listas son comparadas con los verbos encontrados en el texto.

Finalmente, se tiene un contador que almacena las coincidencias. Si el 50% de los verbos están en las listas obtenidas, entonces se pasa a comparar los complementos.

Para analizar los complementos, se utiliza el mismo método de comparación entre las listas explicado anteriormente para los verbos, pero esta vez con parámetros diferentes: listas de palabras con algún sentido semántico (se eliminan las stopwords). Se utiliza el mismo criterio de medida que para el caso de los verbos, lo que quiere decir que si la hipótesis contiene la mitad de las palabras del texto, entonces presentan equivalencia, por lo tanto son paráfrasis.

Resumiendo, si el 50% de los verbos de la hipótesis tienen un análogo en el texto o con cualquiera de sus relaciones (sinónimo, hipónimos, hiperónimo) y lo mismo pasa con las palabras del contexto, entonces hay una paráfrasis. En caso contrario no se declarará la paráfrasis entre ambas frases (ver detalles en la Figura 3.12).

A continuación, en las ecuaciones 3.20, 3.21, 3.22 y 3.23 se detallan las métricas utilizadas, calculadas a partir de las fórmulas presentadas en (Fernando, 2007):

Exactitud: cuantifica la capacidad que tiene el sistema para clasificar como paráfrasis aquellas que realmente lo son y como no paráfrasis las que no lo son.

$$exactitud(Acc) = \frac{VP + VN}{VP + VN + FP + FN} \quad 3.20$$

Precisión: cuantifica la capacidad que tiene el sistema para detectar correctamente las paráfrasis.

$$precisión(Prec) = \frac{VP}{VP + FP} \quad 3.21$$

Cobertura: cuantifica la capacidad que tiene el sistema para capturar la mayor cantidad posible de paráfrasis.

$$cobertura(Rec) = \frac{VP}{VP + FN} \quad 3.22$$

Donde:

VP: Verdaderos positivos.

VN: Verdaderos negativos.

FP: Falsos positivos.

FN: Falsos negativos.

También se usa la siguiente formulación para calcular la F_Medida: función que regula el balance entre precisión y cobertura.

$$F\_Medida(F) = \frac{2 * precisión * cobertura}{precisión + cobertura} \quad 3.23$$

Evaluando en las fórmulas anteriores, con los valores arrojados por este método, se obtiene una exactitud de 66.3%, para una precisión de 70.6%, en una cobertura de 84.5%, lo que representa una F_Medida de 76.9%. Para las pruebas de este método se implementa un prototipo (García y Fernández, 2008) que permite desarrollar los experimentos con mayor facilidad.

De las 1147 instancias positivas y las 578 negativas del corpus utilizado, este método obtiene los valores: VP= 970, VN= 175, FP= 403 y FN= 177. Como se puede apreciar, el mayor problema que tiene el método radica en que clasifica como paráfrasis reales a muchas que no lo son. Esto se debe a que existen varios casos donde los verbos y los complementos están en un 50% de coincidencia entre ambas frases, pero no hay una paráfrasis. Esta situación evidencia la necesidad de utilizar un análisis semántico más profundo.

Después de obtener estos resultados, se decide realizar dos versiones más en las que se introducen pequeñas transformaciones con el objetivo de mejorar los resultados anteriores. Estos esquemas se pueden ver en las Figura 3.13 y Figura 3.14.

La siguiente variante, denominada variante 2, se diferencia de la anterior en el análisis que se hace, pues primero se estudian los verbos, luego los sustantivos y finalmente con los adjetivos, siempre verificando que se cumple que hay coincidencias mayores al 50%. Igualmente que en el caso anterior, se busca en las relaciones de WordNet si no se encuentran las coincidencias exactas. Esto se realiza para cada una de las categorías gramaticales.

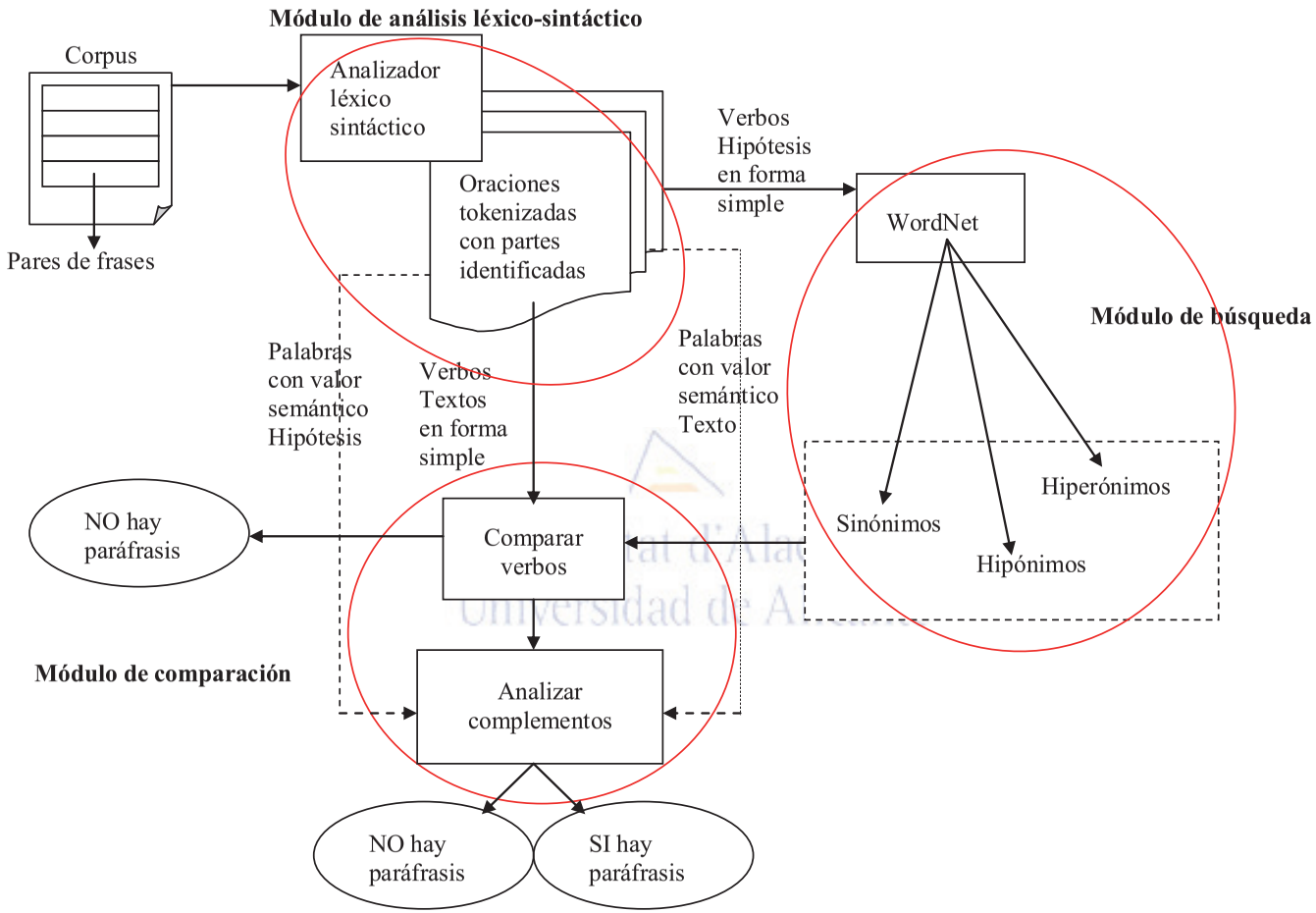


Figura 3.12. Arquitectura del método de reconocimiento de paráfrasis (Variante 1)

Continuando con la siguiente propuesta, la nueva modificación que se realiza es nombrada variante 3. La diferencia fundamental con la anterior es que no se utiliza WordNet, en su lugar se utiliza la lista de sentidos más frecuentes que ofrece Freeling. También, a diferencia de las anteriores, para los adjetivos solo se busca que exista al menos una coincidencia. Estas dos nuevas variantes también son soportadas con sendos prototipo (Ávila, García et al., 2010a; Ávila, García et al., 2010b).

3.5.1.1 Resultados y análisis de los métodos no supervisados de detección de la paráfrasis

Luego de realizar los experimentos con las dos nuevas variantes los resultados obtenidos son los que aparecen en las Tabla 3.37 y Tabla 3.38.

Tabla 3.37. Resultados de las tres versiones en el análisis del corpus de paráfrasis de Microsoft.

	Variante 1	**Variante 2**	**Variante 3**
VP	970	764	998
VN	175	262	139
FP	403	316	439
FN	177	383	149

Tabla 3.38. Resultados de de exactitud, precisión, cobertura y f-medida.

	Exactitud	**Precisión**	**Cobertura**	**F-Medida**
Variante 1	**66.30**	70.60	84.50	76.90
Variante 2	59.40	**70.70**	66.60	68.60
Variante 3	66.00	69.40	**87.00**	**77.20**

En la Tabla 3.37, se puede ver que la variante 3 es la que mejor captura la relación de paráfrasis, pero al mismo tiempo es la que más se confunde asignando como paráfrasis los pares que no lo son. También es la que menos se equivoca en clasificar los casos en los que no hay paráfrasis. A su vez, la Tabla 3.38 muestra que la variante 3 obtiene el mejor valor de F-Medida, esto es debido a que logra una mayor cobertura que las demás. Otro elemento a favor de esta versión es que solo utiliza un recurso (Freeling), a diferencia de la variante 1 y la variante 2 que utilizando dos, lo que hace que esta variante tenga una carga computacional relativamente menor.

Como muestra la variante 2, al introducir el análisis de otras categorías gramaticales como sustantivos y adjetivos, aumenta la cantidad de casos clasificados como negativos. Esto provoca una mejor precisión ya que mejora la cantidad de casos verdaderos negativos (no hay paráfrasis y se detecta correctamente). Sin embargo, es evidente que la cobertura desciende grandemente y esto es provocado por una mayor cantidad de casos falsos negativos (no detecta como correcta una relación de paráfrasis que sí lo es).

La variante 1, a pesar de ser la más simplista en cuanto a la decisión que adopta para clasificar la paráfrasis, se ubica -aunque muy ligeramente- por encima de la variante 3 en exactitud y precisión.

También en un análisis de la Tabla 3.38 se aprecia claramente que las variantes se comparten las mejores posiciones con respecto a exactitud, precisión y cobertura, lo que hace pensar que es preciso buscar una variante que proponga un balance entre las técnicas aplicadas.

Después de realizado este experimento y detectadas las deficiencias de los método, se decide incorporar un análisis semántico multinivel con mayor profundidad. Este nuevo método es nombrado NESP (Nuevo Evaluador Semántico de la Paráfrasis). En el siguiente epígrafe se describe en detalles la arquitectura de este método.

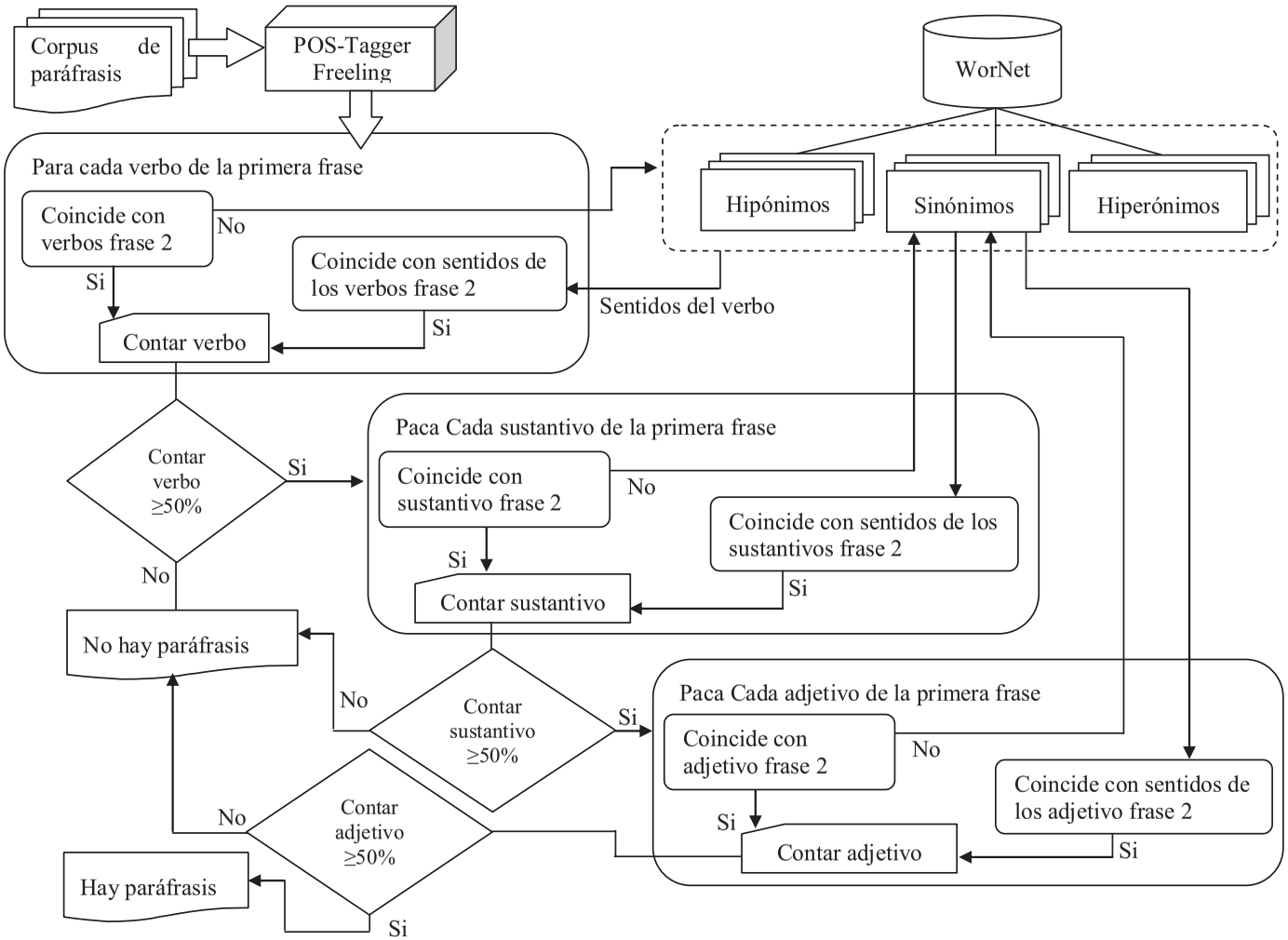


Figura 3.13. Esquema general de la variante 2. (con Freeling1.4 + WordNet).

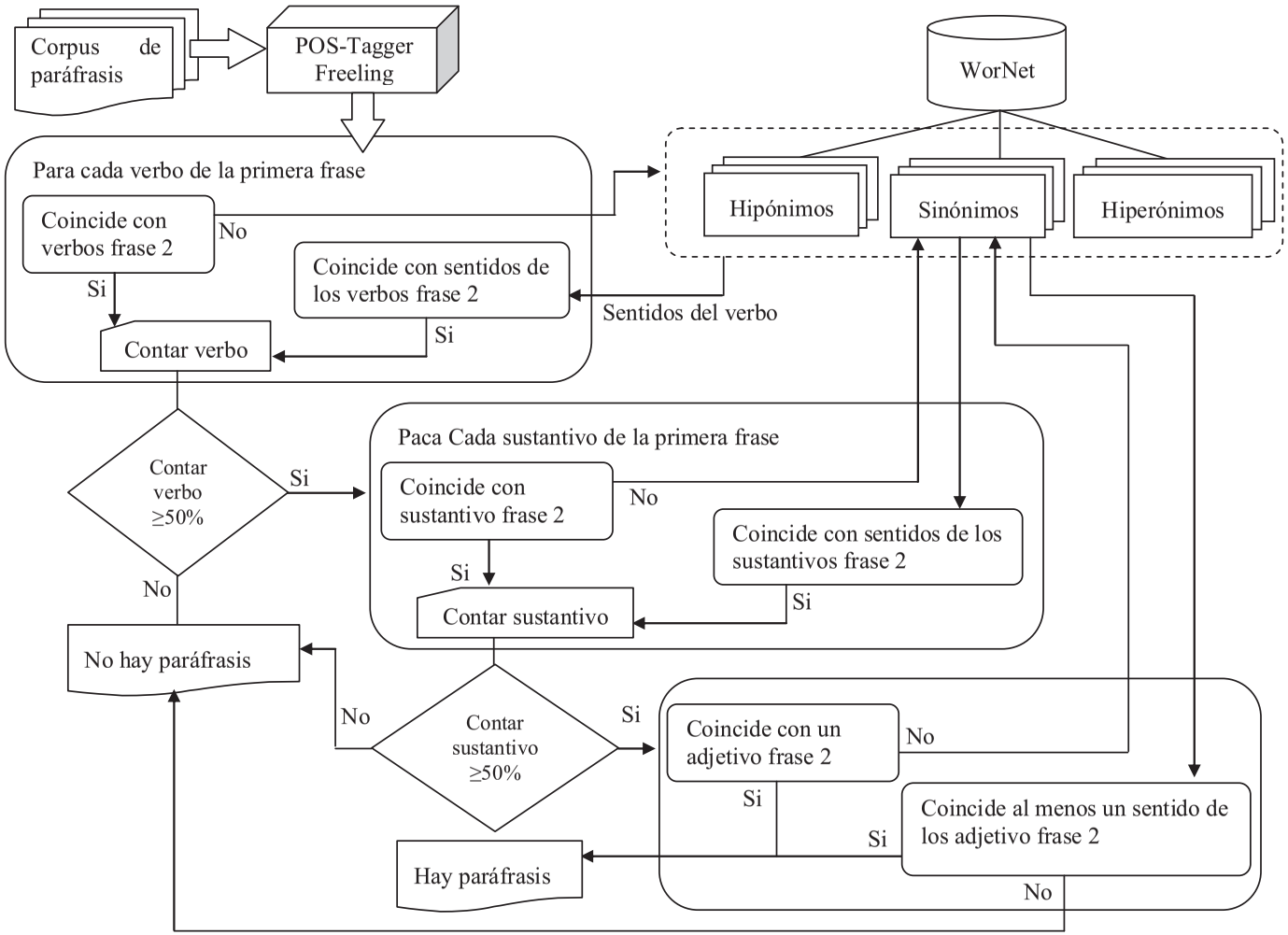


Figura 3.14. Esquema general de la variante 3. (con Freeling 2.2)

3.5.2 Arquitectura y descripción del método de detección de paráfrasis NESP.

Para este nuevo método (NESP), se decide utilizar un método supervisado. Como se puede ver en la Figura 3.15 este método, comienza con el pre-procesamiento del corpus. Cada par de frases es tokenizado, lematizado y se le realiza el POS-tagging usando el recurso Freeling 2.2 (Atserias, Casas et al., 2006).

Posteriormente, se aplican varios algoritmos con el objetivo de extraer los atributos para el sistema de aprendizaje. Este sistema entrena un clasificador usando el modelo basado en Bagging (bagSizePercent:100, numIterations:10, seed:1), utilizando JRip[59] (folds:3, minNo:2.0, optimizations:2, seed:1).

Al concluir la ejecución, como resultado de este entrenamiento se obtiene un modelo capaz de detectar la similitud entre dos frases. Para validar el método también se utiliza el conjunto de test del corpus de paráfrasis de Microsoft (MSRPC) (Dolan, Quirk et al., 2004). En el siguiente apartado se describe el proceso de extracción de los atributos.

3.5.3 Descripción de los atributos usados en el sistema de aprendizaje automático

Para detectar la similitud entre un par de frases, se desarrolla un algoritmo que busca la distancia semántica, basada en WordNet (Miller, Beckwith et al., 1990), entre cada palabra de la primera frase con cada una de la segunda. Se seleccionan cuatro atributos que intentan medir el nivel de proximidad entre ambas frases:

- La distancia mínima para alinear la primera frase con la segunda (MinDist). En el epígrafe 0 se describe en detalle estas medidas;
- La distancia máxima para alinear la primera frase con la segunda (MaxDist);
- El promedio de todas las distancias como resultado de alinear la primera frase con la segunda (AverageDistance);
- El error relativo absoluto de todas las distancias como resultado de alinear la primera frase con la segunda respecto al promedio de estas (Error).

Otros atributos, que también se incluyen son: las relaciones más frecuentes encontradas en el camino más corto a la distancia mínima; como resultado de alinear la primera frase con la segunda.

[59] JRip is an inference and rules-based learner.

La Tabla 3.39 muestran las relaciones seleccionadas como las más frecuentes. Se le añade un peso a cada uno de estos atributos, de acuerdo al lugar que ocupan en el camino más corto entre dos synsets de WordNet.

El camino más corto se calcula mediante el algoritmo de búsqueda BFS.

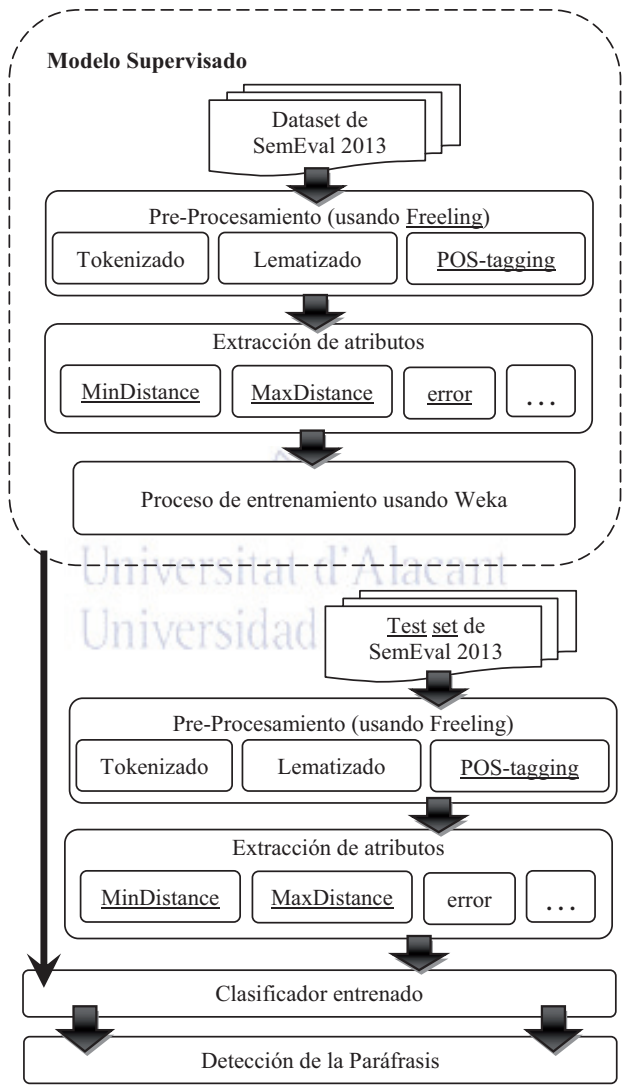


Figura 3.15. Arquitectura del sistema.

Además, se utiliza un atributo que tiene en cuenta cualquier relación que no se haya considerado con anterioridad. Finalmente, como resultado, se obtienen 22 atributo para esté método de alineamiento.

Tabla 3.39. Relaciones más frecuentes con sus pesos asociados.

Relaciones	**Pesos ($función\ W$)**
Antonym	1000
Synonym	0
Hyponym/ Hypernym	100 si existe un antónimo antes, 30 si existe otra relación antes (excepto synonym, hyponym, hypernym), 5 en otro caso.
Meber_Holonym/ PartHolonym	100 si existe un antónimo antes, 20 si existe un hyponym o un hypernym, 10 en otro caso.
Cause/ Entailment	100 si existe un antónimo antes, 2 en otro caso.
Similar_To	100 si existe un antónimo antes, 3 en otro caso.
Attribute	100 si existe un antónimo antes, 8 en otro caso.
Also_See	100 si existe un antónimo antes, 10 en otro caso.
Derivationaly_Related_Form	100 si existe un antónimo antes, 5 en otro caso.
Domain_Of_Synset_Topic	100 si existe un antónimo antes, 13 en otro caso.
Domain_Of_Synset_Usage	100 si existe un antónimo antes, 60 en otro caso.
Member_Of_Domain_Topic	100 si existe un antónimo antes, 13 en otro caso.
Member_Of_Domain_Usage	100 si existe un antónimo antes, 60 en otro caso.
Other	100

Nota: Los pesos asignados a cada relación fueron determinados experimentalmente en función de la importancia que reviste la relación para el objetivo perseguido en eta tarea.

3.5.4 Distancia Semántica para la determinación de la paráfrasis

Según lo que se ha visto hasta este punto, la distancia depende del cálculo de similitud entre las frases, basada en un análisis de las relaciones de WordNet (Miller, Beckwith et al., 1990)y solo se toman para esto las más frecuentes.

Cuando se busca el camino más corto entre dos synsets de WordNet, las relaciones más frecuentes se consideran aquellas extraídas de un análisis realizado en el corpus de entrenamiento. La distancia entre dos synsets se calcula simplemente sumando de los pesos asignados a cada conexión.

$$MinDistP(P,Q) = MinDistS(P_X, Q_Y), \forall\ (X,Y) \tag{3.24}$$

$$MinDistS(X,Y) = Min(X_i, Y_j), \forall (i,j) \tag{3.25}$$

$$Min(X_i; Y_j) = \sum_{k=0}^{k=m} W(Rel(L[k], L[k+1])) \tag{3.26}$$

$$L = BFS(X_i; Y_j) \tag{3.27}$$

Donde i y j representa el i-ésimo y j-ésimo sentido de la palabra, P y Q representan colecciones de palabras; P_X es la X-ésima palabra de P; Q_Y es la Y-ésima palabra de Q; $MinDistP$ obtiene un valor que representa la distancia semántica mínima a través de WordNet (Este recurso está embebido en el recurso integrador ISR-WN (Gutiérrez, Fernández et al., 2010a; 2011). $MinDistS$ la mínima distancia semántica entre dos palabras; Min representa la distancia semántica mínima entre dos colecciones de sentidos; L es una colección de synsets que representa la trayectoria mínima entre dos synsets utilizando BFS; Rel obtiene los tipos de relaciones semántica entre dos synsets;. W es un funciones que aplica las reglas descritas en la Tabla 3.39.

La distancia máxima y promedio se calcula de una manera similar, pero usando el máximo y el promedio en lugar del mínimo.

3.5.5 Alineamiento Semántico en el reconocimiento de la paráfrasis

Primeramente, las dos frases son pre-procesadas con Freeling 2.2 y se clasifican las palabras de acuerdo a su categoría gramatical. Luego, todos los sentidos de cada palabra son agrupados y tratados como un grupo. La distancia entre dos grupos será la distancia mínima entre los sentidos de cualquier par de palabras pertenecientes a un grupo.

En el ejemplo de la Figura 3.16, se selecciona la distancia `Dist=280` es seleccionada para el par Balance-Culture (costo mínimo). Siguiendo la explicación en el epígrafe 3.5.3, se extraen los atributos con el objetivo de medir el nivel de proximidad entre ambas frases.

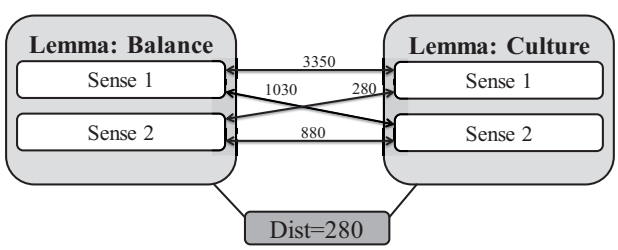


Figura 3.16. Distancia entre los grupos Balance y Culture, camino mínimo entre las relaciones de dos sus sentidos.

La distancia máxima y la distancia promedio son calculadas de la misma forma, pero usando la relación más distante y el promedio.

3.5.6 Descripción de la fase de entrenamiento

Para el proceso de entrenamiento, se utiliza el framework de aprendizaje supervisado Weka, incluyendo todos los conjuntos de entrenamiento (instancias positivas y negativas) como corpus para entrenar el clasificador. Para desarrollar los experimentos se implementa un prototipo (Dávila,

Gutierrez et al., 2012) que facilita la experimentación. Se realizan varios experimentos para seleccionar el clasificador correcto. El mejor resultado se obtiene con un modelo basado en Bagging (usando el algoritmo JRip). Finalmente, se usa una técnica de validación de diez pliegue (10-fold cross validation).

3.5.6.1 Resultados y análisis del método NESP con el corpus de paráfrasis de Microsoft.

Debe recordarse, que tanto el conjunto de entrenamiento como el conjunto de prueba están totalmente desbalanceados, lo que evidentemente afecta el resultado de la prueba. Al ejecutar el método sobre el conjunto de prueba se obtienen los resultados que se muestran en la Tabla 3.41.

Tabla 3.40. Resultados al clasificar con NESP (conjunto de prueba del corpus MSRPC)..

	NESP
VP	1096
VN	63
FP	515
FN	51

Los resultados de exactitud, precisión, cobertura y F-medida, se muestran en la Tabla 3.41

Tabla 3.41. Resultados de exactitud, precisión, cobertura y F-medida de la variante NESP.

Exactitud	Precisión	Cobertura	F-Medida
67.19	68.03	95.55	79.48

Los valores obtenidos son insertados y comparados con las otras variantes y otros sistemas en la Tabla 3.42, extraída de la página 33 de la tesis de maestría del Samuel Fernando, titulada Paraphrase Identification (Fernando, 2007) y el artículo (Fernando y Stevenson, 2008). La tabla se ha ordenado por la exactitud. Estos sistemas se explican en el epígrafe 2.6.6.

Los valores obtenidos con los métodos aplicados en el epígrafe 3.5.1, variantes de la uno a la tres, aunque no son los mejores, no están tan lejos de los alcanzados por los otros métodos teniendo en cuenta la simpleza del planteamiento por el cual se decide la paráfrasis. A pesar de eso, con la variante 2 y la variante 1, se supera -en la precisión alcanzada- a la variante nombrada Mihalcea2006, descrita en (Mihalcea, Corley et al., 2006).

No obstante, a pesar de que obtiene buenos resultados, NESP solo se puede ubicarse en la posición 10 entre las 15 variantes analizadas. Sin embargo, tomando en cuenta la cobertura se ubica en la segunda posición de este ranking.

Tabla 3.42. Comparación con diferentes sistemas.

Variantes	Exactitud	Precisión	Cobertura	F-Medida
matrixJcn	74.1	75.2	91.3	82.4
matrixLch	73.9	74.8	91.6	82.3
matrixLin	73.7	74.2	92.5	82.4
matrixLesk	72.9	73.5	92.6	82
matrixRes	72.2	73.8	90.4	81.2
Qiu2006	72	72.5	93.4	81.6
Zhang2005	71.9	74.3	88.2	80.7
matrixWup	71.6	75.2	85.4	80
Mihalcea2006	70.3	69.6	97.7	81.3
NESP	67.19	68.03	95.55	79.48
Variante 1	66.3	70.6	84.5	76.9
Variante 3	66	69.4	87	77.2
Vector-based	65.4	71.6	79.5	75.3
Variante 2	59.4	70.7	66.06	68.6
Random	51.3	68.3	50	57.8

Además de los casos anteriores, el método NESP también se prueba en la competición de SemEval-2013. En este caso se usa como conjunto de entrenamiento el ofrecido por el comité organizador, específicamente para la tarea Evaluating Phrasal Semantics (EPS).

La tarea EPS de SemEval-2013, ofrece varias medidas para comparar los sistemas participantes. Algunas de esta son: F-Medida (FM), respuestas correctas (RC) que serían las instancias correctamente clasificadas, verdaderos positivos (VP) que son las instancias correctamente clasificadas como positivas, falsos positivos (FP) que representa a la Instancias incorrectamente clasificadas como positivas, verdaderos negativos (VN) que reúne a las instancias correctamente clasificadas como negativas y los falsos negativos (FN) que contendría las instancias incorrectamente clasificadas como negativas.

3.5.6.2 Resultados del método NESP en SemEval-2013

El comportamiento de este método se ofrece en la Tabla 3.43, para los corpus del inglés y el italiano.

Tabla 3.43. Resultados oficiales de SemEval-2013 para el método utilizado (inglés y español).

Corpus	FM	RC	VP	FP	VN	FN
English	0.6892	2826	1198	325	1628	755
Italian	0.6396	574	245	96	329	180

El único cambio que se realiza para procesar el corpus en italiano es el procesamiento de las palabras de este idioma usando Freeling y luego se obtienen los synsets de WN para el inglés. El proceso continúa de la misma forma que para el inglés.

Como se muestra en la Tabla 3.43, el principal problema de este método es la clasificación de las instancias positivas. En ocasiones, la distancia entre las frases positivas es muy grande. Esto es provocado por las relaciones encontradas en el camino mínimo, las que son muy similares a otras que se encuentran en los pares de instancias negativas, por lo que el clasificador las descifra como negativas y así las clasifica (ver Figura 3.17).

La Figura 3.17 muestra un gráfico distribucional de 200 ejemplos de instancias negativas y positivas. El gráfico ilustra cuán cercanos a cero están los valores de las instancias positivas, mientras las negativas están más alejadas de este valor. Sin embargo, en un rango aproximado entre 80 y 200, se puede apreciar los valores de las instancias positivas y negativas posicionándose bastante próximos. Esta puede ser la causa de equivocación del clasificador, cuando confunde instancias positivas como negativas.

Los resultados de este método se comparan con la variante mejor posicionada de esta competición en la Tabla 3.44.

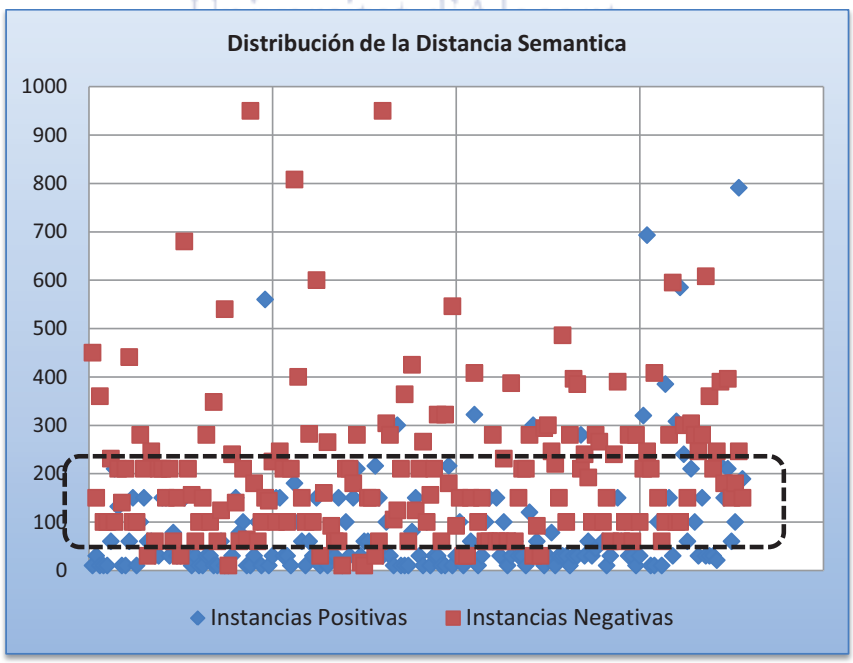


Figura 3.17. Distancia Semántica entre instancias negativas y positivas.

Tabla 3.44. Resultados comparativos par el corpus del inglés.

Equipo	exactitud	cobertura	precisión
Primer lugar[60]	0.802611	0.751664	0.836944128
NESP	0.723502	0.613415	0.786605384

Nota: Primer Lugar → Hochschule Hannover - University of Applied Sciences and Arts (HsH).

[60] http://aclweb.org/anthology/S/S13/S13-2008.pdf

3.6 El stemming

Para poder describir el método de stemming que se implementa es necesario, primeramente explicar el método utilizado para el agrupamiento en familias de palabras. Valga aclarar que una familia estará compuesta por el conjunto de todas las palabras que comparten un lexema común, mayor de un tri-grama y que posee un relacionamiento semánticos. Para este objetivo, se utiliza la DEx descrita en el epígrafe 3.1.

3.6.1 Agrupamiento de palabras usando la Distancia Extendida

El agrupamiento de palabras en familias parte de la siguiente hipótesis de similitud entre dos palabra: Dos palabras son similares y por consiguiente pueden pertenecer a la misma familia, si las diferencias entre ellas ocurren en caracteres lo más lejos posible de la raíz y además esta diferencia es menor que un cierto umbral, determinado experimentalmente. Para este agrupamiento se utiliza la DEx según la ecuación 3.1.

El algoritmo de agrupamiento consta de varios pasos, primero es necesario obtener una lista de palabras únicas (LPU), extraídas de un corpus del lenguaje para el que se desea realizar el stemming. Para este trabajo se usaron 28390 palabras de la lista de términos para el español del diccionario del editor de textos MiKTeX 2.6[61]. Este editor es una implementación actualizada de TeX (Knuth, 2007) y programas relacionados para Windows. El algoritmo continúa de la forma siguiente:

Paso 1: Organizar la LPU alfabéticamente en orden ascendente (A-Z).

Paso 2: La primera palabra de la lista se toma como pivote y se compara (usando la DEx) con las restantes en un rango R de 500 palabras (este valor es tomado experimentalmente, partiendo de que no existe una familia de palabra para el español que superen esta cantidad). Este valor, tiene que ser `obligatoriamente mayor` que el máximo número de palabras, obtenidas por flexión o derivación, que una familia del lenguaje utilizado pueda tener.

Paso 3: La lista obtenida se ordena teniendo en cuenta las distancias calculadas en el paso anterior. De esta forma las palabras más cercanas al pivote estarán al comienzo de la lista (ver Tabla 3.45).

Paso 4: Se aplica un punto de corte para separar las posibles familias. Una palabra será parte de la familia solo si se cumple que $\mathrm{DEx}(x, y_0) \leq 0.1$, donde x representa el pivote y y_0 la palabra inmediata. Si no se cumple esta premisa, la familia será constituida solo por el pivote.

[61] http://www.miktex.org

Paso 5: Son seleccionadas para pertenecer a la familia todas las palabras donde la comparación obedezca la relación $\Delta \mathrm{DEx} = \mathrm{DEx}(\mathrm{x}, \mathrm{y}_{i+1}) - \mathrm{DEx}(\mathrm{x}, \mathrm{y}_i) \leq \mathrm{E}$; donde E es un umbral experimentalmente determinado.

Paso 6: La familia obtenida es recuperada y eliminada de la lista original antes de repetir nuevamente el proceso, que continuará hasta que la lista esté vacía.

La expresión $\mathrm{DEx}(\mathrm{x}, \mathrm{y}_0) \leq 0.1$ en el paso 4 y $\Delta \mathrm{DEx} = \mathrm{Dx}(\mathrm{x}, \mathrm{y}_{i+1}) - \mathrm{DEx}(\mathrm{x}, \mathrm{y}_i) \leq \mathrm{E}$ en el paso 5, impiden agrupar palabras alrededor del pivote cuando la similitud no es significativa. En este caso, los valores fueron establecidos en 0.1 y 0.06 respectivamente, después de experimentar con un gran número de valores y buscar los mejores resultados.

No se descarta que el valor de E cambie en dependencia de las características del corpus que se está analizando, incluso, podría darse el caso en el que se usen valores distintos para diferentes familias. El valor propuesto se ha tomado para el corpus del español antes mencionado.

Ya se sabe que la primera restricción ($\mathrm{DEx}(\mathrm{x}, \mathrm{y}_0) \leq 0.1$), garantiza que cambios significativos en posiciones cercanas a la raíz de las palabras no sean permitidos. Una vez que se logra esta premisa, la segunda ($\Delta \mathrm{DEx} \leq \mathrm{E}$) determina dónde debe finalizar la familia de palabras, es el punto de corte. Con la inclusión del parámetro E, puede ser detectado a tiempo un cambio abrupto entre la distancia de dos palabras, esto indica que es un posible punto en el cual se termina una familia. Por tanto, este valor será el punto de inflexión para determinar la longitud de la familia.

Una vez obtenidos los conjuntos de familias de palabras, se procede a la determinación del stem. El siguiente epígrafe explica este paso en detalle.

3.6.2 Obtención del stem a partir del lexicón de familias de palabras

Gracias al lexicón con todas las familias de palabras, obtenidas a partir de la DEx, el algoritmo para obtener el stem puede ser reducido a buscar (a través de las características lexicográficas) la sub-secuencia común inicial (SCI) de la última palabra aceptada en la familia. Esta SCI, será el stem mínimo necesario que permite agrupar las diferentes palabras en una familia. En la Tabla 3.45 se muestra un pequeño ejemplo de una familia (reducida para ahorrar espacio), la que se ha construido con un parámetro $\mathrm{E}=0.06$ como punto de corte.

Como se puede apreciar, la celda extrema inferior derecha contiene la SCI más corta para la familia de palabras analizada, la que será seleccionada como stem. Al analizar la Tabla 3.45, la palabra que sigue a `adapto` es `adepto`, pero su DEx es mayor que el umbral $\mathrm{E}=0.06$, por lo que esta lista es truncada en este punto y se obtiene la familia de la palabra `adaptabais`.

Tabla 3.45. Familia (reducida) para la palabra adaptabais.

Pivote	Palabra	DEx	SCI
adaptabais	adaptabas	0.00882400706110968	adaptaba
	adaptaba	0.008824036325403270	adaptaba
	adaptaban	0.00885651031878123	adaptaba
	adaptabilidad	0.0149803569314887	adaptab
	adaptas	0.0252930222190664	adapta
	adaptar	0.0252930429293315	adapta
	adaptases	0.0253398951817534	adapta
	adaptaría	0.025338545649806	adapta
	adapto	0.0427819187890526	adapt
	adeptos	0.302810286744015	ad

Se puede ver fácilmente en la Tabla 3.45, que la SCI que ha quedado es precisamente la porción o raíz que caracteriza a esta familia de palabras. Nótese además, que la aparición de la letra `í` acentuada, no ha marcado ninguna diferencia. Esto se logra, como bien se explicó en el epígrafe 3.1, con la incorporación del diccionario con los pesos para los caracteres. Para este caso, los pesos de las letras `i` e `í` se igualan, por lo que no hay diferencias al analizar las posibles permutaciones entre estos caracteres.

A continuación, se muestra una pequeña porción de la salida del algoritmo de stemming (ver Figura 3.18). El stem, queda reflejado en la variable del mismo nombre `Stem`, para el caso de la figura mostrada, el stem obtenido es `adapt`. La variable `Count` es 77, que representa la cantidad de palabras que se han incluido en la familia. La variable `Pivot`, es el pivote que se ha colocado, adaptabais. Por otro lado, `Word` es la palabra que se compara con el pivote. Así como SCI, es la subsecuencia común inicial máxima entre las palabras comparadas en cada iteración. La variable `SubsComunMax`, es la subsecuencia común máxima. A su vez, `Operations`, es la cadena de operación de transformación que se ha tenido que ejecutar para intentar igualar el pivote y la palabra en análisis. Finalmente, `Extended`, es el valor de la distancia entre el pivote y la palabra en la variable `Word`.

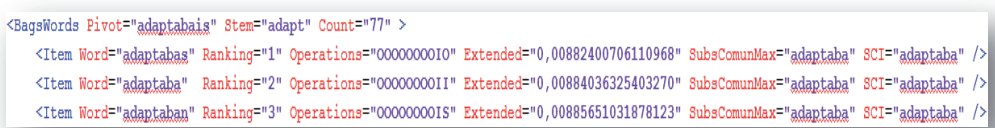


Figura 3.18. Porción de captura de pantalla de la salida del algoritmo de stemming.

Para validar este método de stemming, así como la formación de familias de palabras, se han diseñado varios experimentos los que se muestran en el siguiente epígrafe.

3.6.3 Experimentos con el stemming

Para poder realizar la experimentación se implementa un prototipo (Díaz y Fernández, 2009) el que permite desarrollar el método creado. Se realizan cuatro experimentos, el primero para medir la capacidad de la DEx en el agrupamiento de familias palabras del español. El segundo intenta medir la precisión del método de stemming también para el español.

3.6.3.1 Procedimiento de formación manual de las familias de palabras del español

Para garantizar resultados aceptables, esta tarea es realizada por un personal vinculado al estudio del lenguaje, la selección se realiza a partir de profesores investigadores de las Facultades de Idiomas y de Informática de la Universidad de Matanzas. Se les proporciona el lexicón con todas las palabras, organizadas por orden alfabético, a tres grupos (A, B, C) constituidos por cinco personas cada uno. Un cuarto grupo (D) -con los especialistas de mayor experiencia- se reserva para las decisiones finales. Los grupos A, B y C están constituidos por estudiantes de 4to año de la Facultad de Idioma de la mencionada universidad. El grupo D está constituido por profesores investigadores de las Facultades de Idiomas y de Informática de la misma universidad.

De esta forma, cada grupo es independiente y no tiene contacto con los grupos restantes. Una vez que los grupos ha realizado su clasificación, se reúne el grupo A con el B y confrontan la clasificación realizada. Una vez que tienen consenso (si no lo tienen, el cuarto grupo (D) de especialistas decide la clasificación final), se reúnen con el grupo restante (C) y llegan a un acuerdo final (igualmente si no llegan a un consenso, el cuarto grupo (D) decide). La Tabla 3.46 muestra la cantidad de las familias finalmente seleccionadas.

Ahora, es el momento de pasar a ver la capacidad de agrupamiento en familias de la DEx. Para ellos se detalla lo realizado en el próximo epígrafe.

3.6.3.2 Capacidad de la DEx en el agrupamiento de familias palabras del español

Como se mencionó anteriormente, este experimento persigue la finalidad de evaluar la capacidad de la DEx de agrupar correctamente las palabras en sus respectivas familias. Se utiliza para esto el lexicón antes mencionado con 28390 palabras del español. Luego, se calcula que una muestra significativa (Pita Fernández y Pértega Díaz, 2001) para esta población es de 384 palabras. Pero, como el experimento intenta evaluar el agrupamiento de palabras en familias, se decide verificar 356 familias de palabras en lugar de 384 palabras.

La Tabla 3.46 muestra la composición del lexicón utilizado en el experimento. Existen 10628 palabras en las 356 familias seleccionadas. Dichas familias, fueron obtenidas aleatoriamente

tratando de escoger grupos representativos de cada letra del alfabeto. Finalmente, solo se excluyen las que comienzan con x, y, z, debido a que no existen muchas familias que se inicien con estas letras y además, las que existen poseen muy pocas palabras.

Por último, la verificación tiene lugar interrogando el indexado generado por el algoritmo diseñado y contrastando con las 356 familias de palabras seleccionadas por los especialistas.

Tabla 3.46. Características de las familias creadas por los expertos

Comienza con	Familias	Palabras	Comienza con	Familias	Palabras
a	17	572	l	17	513
b	17	407	m	17	358
c	17	341	n	17	474
d	17	802	o	17	520
e	17	689	p	17	425
f	17	611	q	10	387
g	17	400	r	17	651
h	17	514	s	17	546
i	17	729	t	17	429
j	17	442	u	17	257
k	6	22	v	17	539
Total				356	10628

Para obtener un veredicto de los resultados alcanzados, la comprobación se realiza a través de las ecuaciones siguientes:

$$\text{precisión} = \frac{CPF}{CPF+FP} \quad 3.28$$

$$\text{cobertura} = \frac{CPF}{CPF+FN} \quad 3.29$$

Donde:

- CPF: Cantidad de palabras identificadas que pertenecen a la familia.
- FP: Cantidad de palabras identificadas que no pertenecen a familia, falsos positivos.
- FN: Cantidad de palabras no identificadas que sí pertenecen a la familia, falsos negativos.

$$\text{exactitud} = \frac{CPF}{CPF+FN+FP} \quad 3.30$$

Otra medida, utilizada para obtener un balance entre precisión y cobertura es la F-Medida (Fm_β). En este caso, se selecciona el parámetro $\beta=1$, con ello se intenta dar la misma importancia a ambos términos. Para el cálculo de esta medida se usa la ecuación 2.3.

Una vez concluido los experimentos, los resultados para la creación de las familias de palabras, utilizando la DEx son: Precisión=0.9862658, Cobertura=0.9979343, Fm_{β}= 0.9920657, Exactitud=0.9842563.

Una vez comprobada la formación de familias, se procede a la comprobación del método de stemming. Esto podrá ser apreciado en el siguiente epígrafe.

3.6.3.3 Experimento para comprobación del método de stemming

En este experimento se chequean los resultados del algoritmo de stemming. Se utilizan las mismas familias de palabras del ejemplo anterior. Para este caso, cada stem calculado con el método explicado en el epígrafe 3.6.2, es obtenido aleatoriamente del indexado generado por este método, verificado con una lista de stem obtenida manualmente. Para tener una idea de lo ocurrido, se hará el análisis partiendo de los errores cometidos en el agrupamiento de familias, finalmente esto es lo que provoca una incorrecta selección del stem. La Tabla 3.47 muestra solo los pivote seleccionados de las familias que `no fueron correctamente creadas`, H (número de palabras seleccionadas por los especialistas y que forman parte de las familias, M (número de palabras incluidas en la familia por el algoritmo), FP (falsos positivos), FN (falsos negativos), las últimas tres columnas representan los valores de cobertura, precisión y exactitud obtenidos en la generación del stem de cada palabra de la familia.

Tabla 3.47. La 16 Familias con errores

no	H	M	FN	FP	pivote	cobertura	Precisión	F-meddida
13	101	103	0	2	mecánica	1	0.980952381	0.990384615
15	85	87	0	2	toca	1	0.97752809	0.988636364
14	73	75	0	2	necesidades	1	0.974025974	0.986842105
6	102	106	0	4	dedicaban	1	0.963636364	0.981481481
12	21	23	0	2	leonada	1	0.92	0.958333333
8	8	9	0	1	fiera	1	0.9	0.947368421
3	12	14	0	2	cigarreras	1	0.875	0.933333333
5	61	89	0	28	decadencias	1	0.760683761	0.86407767
16	4	7	0	3	vagones	1	0.7	0.823529412
7	4	9	0	5	docentes	1	0.642857143	0.782608696
9	9	35	0	26	graves	1	0.573770492	0.729166667
10	5	67	0	62	humilde	1	0.519379845	0.683673469
2	7	8	0	1	chequear	1	0.888888889	0.941176471
1	8	16	0	8	celular	1	0.666666667	0.8
11	8	16	0	8	lenta	1	0.666666667	0.8
4	6	20	0	14	cuaternarias	1	0.588235294	0.740740741

Haciendo un análisis del comportamiento del método, la mayor parte de los errores fueron provocados por la inclusión de palabras erradas en las familias (falsos positivos); no se existen falsos negativos en las familias. Estos errores, son productos de la selección del punto de corte para este experimento （E=0.08）. Se puede ver en la Tabla 3.48 que las palabras: vagos y vago, son

erróneamente incluidas en la familia con pivote vagones. Estas dos palabras fueron incluidas debido a que el umbral que se utilizó fue E=0.08, esto no hubiera sucedido de usar, por ejemplo, el valor 0.06. Pero, a su vez puede suceder que con este valor queden fuera de las familias, aquellas palabras que -aún perteneciendo al mismo concepto- estén más distantes del pivote.

Tabla 3.48. Ejemplo de familia de palabras con error debido al punto de corte seleccionado (E=0.08).

Palabras	Distancia de edición extendida
vagonetas	DEx =0.025340643184534
vagoneta	DEx =0.0253869282742426
vagón	DEx =0.0428624492675192
vagos	**DEx =0.0724997208568067**
vago	**DEx =0.0725017097933701**

Es difícil comparar este método con otros, debido a que se utilizan diferentes técnicas, sin embargo, se decide hacer el mismo experimento utilizando el algoritmo de Porter para el español, con el objetivo de tener un punto de comparación. Luego de efectuado, se obtienen los resultados siguientes:

Tabla 3.49. Comparación con el algoritmo de Porter.

Stemmer	Resultados para la 356 familias		
	cobertura	precisión	F-medida
Stemmer con la DEx	1	0.9842563	0.9920657
stemmer de Porter para español	0,528847764	0,995084215	0.690645

Se puede ver en la Tabla 3.49, que el algoritmo de Porte obtiene una buena precisión, pero por otra parte, los valores de cobertura y exactitud son bastante discretos. Esto indica, específicamente que el algoritmo de Porte deja de incluir muchas palabras en las familias que en realidad pertenecen a ellas (FN). Por esta razón, también se ve afectada su exactitud.

3.6.4 Análisis de los resultados obtenidos

Como se aprecia en los resultados de los experimentos, la capacidad de agrupamiento en familias de la EDx es prometedora, alcanzando resultados por encima de un 98% en todos los indicadores de calidad.

El algoritmo diseñado para la obtención del stem alcanza muy buenos resultados, comete errores solo en 16 de las 356 familias utilizadas en el experimento.

Como se puede apreciar en la Tabla 3.49, el stemmer de Porter reporta una buena precisión, pero un valor de cobertura muy bajo y por ende también una baja exactitud. Esto indica que el algoritmo de

Porter no agrupa correctamente -bajo el mismo concepto- todos los términos que debería, produciendo más familias de lo normal y por consiguiente una mayor lista de stems.

En el siguiente epígrafe se detalla la investigación realizada en otra de las tareas intermedias de gran importancia para el PLN, el Reconocimiento de Entidades Nombradas.

3.7 El Reconocimiento de Entidades

Se debe recordar en este punto, que toda la experimentación realizada en este epígrafe parte de la premisa de que se utilizan documentos sin procesamiento, es decir, sin realizar los clásicos: análisis léxico-sintáctico o semántico que muchas veces son utilizados en la mayoría de las tareas de PLN. Precisamente, se intenta con esto responder a una de las preguntas de investigación planteadas.

Antes de comenzar a explicar los métodos creados y todos los elementos desarrollados, es necesario aclarar la propuesta de evaluación de efectividad y eficiencia utilizada en este trabajo.

3.7.1 Medidas de evaluación de efectividad y eficiencia

La medición de la efectividad se encarga de determinar la precisión con que se acomete la tarea de identificación y clasificación de las entidades. En este trabajo se utilizan las ecuaciones definidas en (Fernández, 2005). Según este trabajo, en la fase de identificación la precisión se asume como el resultado de dividir las entidades identificadas correctamente entre el total de entidades identificadas.

$$precisión = \frac{entidades\ identificadas\ correctamente}{entidades\ identificadas} \qquad 3.31$$

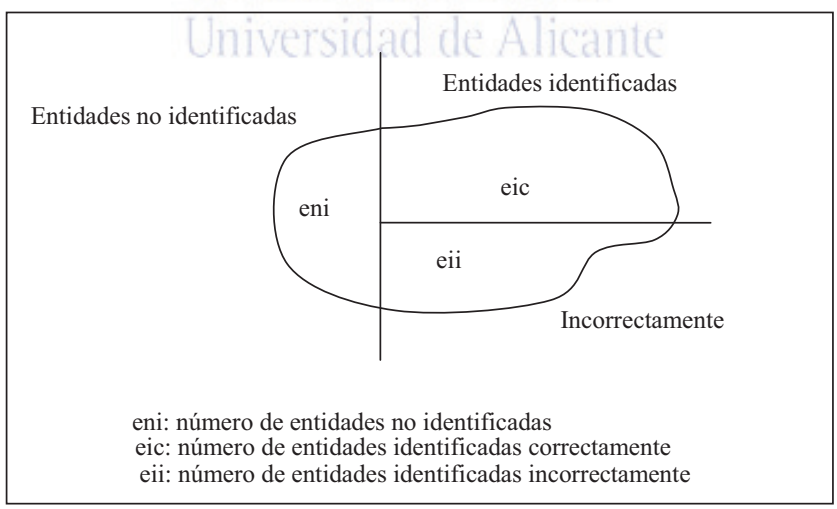


Figura 3.19. División de los términos en la clasificación de entidades

Tomando el análisis de la Figura 3.19 se hace el ajuste de las fórmulas para determinar la precisión (ver ecuación 3.32).

$$precisión = \frac{eic}{eic + eii} \quad 3.32$$

La cobertura se define como la división de las entidades identificadas entre el total de entidades que existen en el documento en análisis.

$$cobertura = \frac{entidades\ identificadas\ correctamente}{entidades\ en\ el\ documento} \quad 3.33$$

Esta fórmula se transformó también según el análisis hecho en la Figura 3.19 y se obtiene la ecuación 3.34.

$$cobertura = \frac{eic}{eic + eii + eni} \quad 3.34$$

Donde: *ei* → total de entidades identificadas

eni → número de entidades no identificadas

eic → número de entidades identificadas correctamente

eii → número de entidades identificadas incorrectamente

Otra de las medidas utilizadas es la F-medida, para ello se usa la ecuación 2.3.

En la fase de clasificación se asumen estas mismas ecuaciones, solo se cambiaría el tipo de operación, que en este caso sería clasificación. Por lo tanto, se sustituye en las ecuaciones el término `identificadas` por `clasificadas`.

La evaluación final conjunta de las dos etapas sería:

$$\text{PrecTotal} = \frac{2\ (\text{precIdentif} * \text{precClasif})}{\text{precIdentif} + \text{precClasif}} \quad 3.35$$

Donde:

PrecTotal: es la precisión total considerando ambas fases (identificación y clasificación);

precIdentif: es la precisión en la fase de identificación;

precClasif: es la precisión en la fase de clasificación.

3.7.2 La identificación de las entidades

El proceso de identificación es un elemento fundamental dentro de esta investigación, pues es a partir de aquí que se podrán reconocer correctamente las entidades que aparecen en el documento en estudio. El proceso consiste, no sólo en encontrar las entidades, sino poder definir su tamaño correcto, es decir, donde empieza, donde termina y cuantas palabras componen la entidad (Fernández Orquín, Muñoz Guillena et al., 2004).

El proceso de identificación es el encargado de detectar las entidades encontradas en su forma original. Para ello se utilizan las reglas de colocación de las mayúsculas según la gramática española. De esta forma, al hacer un análisis del texto se pueden detectar entidades tales como nombres de personas, de organizaciones y de lugares geográficos, pues son palabras que se escriben con inicial mayúscula.

También se implementa la identificación de entidades de otro tipo como las fechas, horas, números, cantidades, direcciones electrónicas, siglas, entre otras. Estas entidades se describen más adelante.

Uno de los problemas de esta etapa, es determinar correctamente dónde empieza una entidad y donde termina. Para ello se implementa un algoritmo que identifica las entidades y hace un tratamiento especial para aquellas que se encuentran en inicio de frase.

Como muestra el algoritmo de la Figura 3.21, en este método se lee la lista de tokens completa y luego se recorre iterando por palabras. A partir de aquí se van analizando las diferentes posibilidades.

El análisis tiene dos vertientes, una para cuando es inicio de frase y otra para cuando no lo es. La diferencia estriba en que las palabras que están en inicio de frase, siempre se van a escribirse con mayúscula, por lo que es necesario determinar si forman parte de la entidad o no.

El análisis para cuando es inicio de frase se convierte en saber si es o no una entidad, o si es un artículo y forma parte de una entidad que esté a continuación. Para cuando no es inicio de frase, el análisis parte de determinar si la palabra está escrita con inicial mayúscula o no.

Este algoritmo va a recorrer el corpus completo, identificando y etiquetando las entidades. Serán clasificadas aquellas que no sean ambiguas, generalmente, algunas entidades débiles que se pueden reconocer a partir de reglas o patrones previamente definidos.

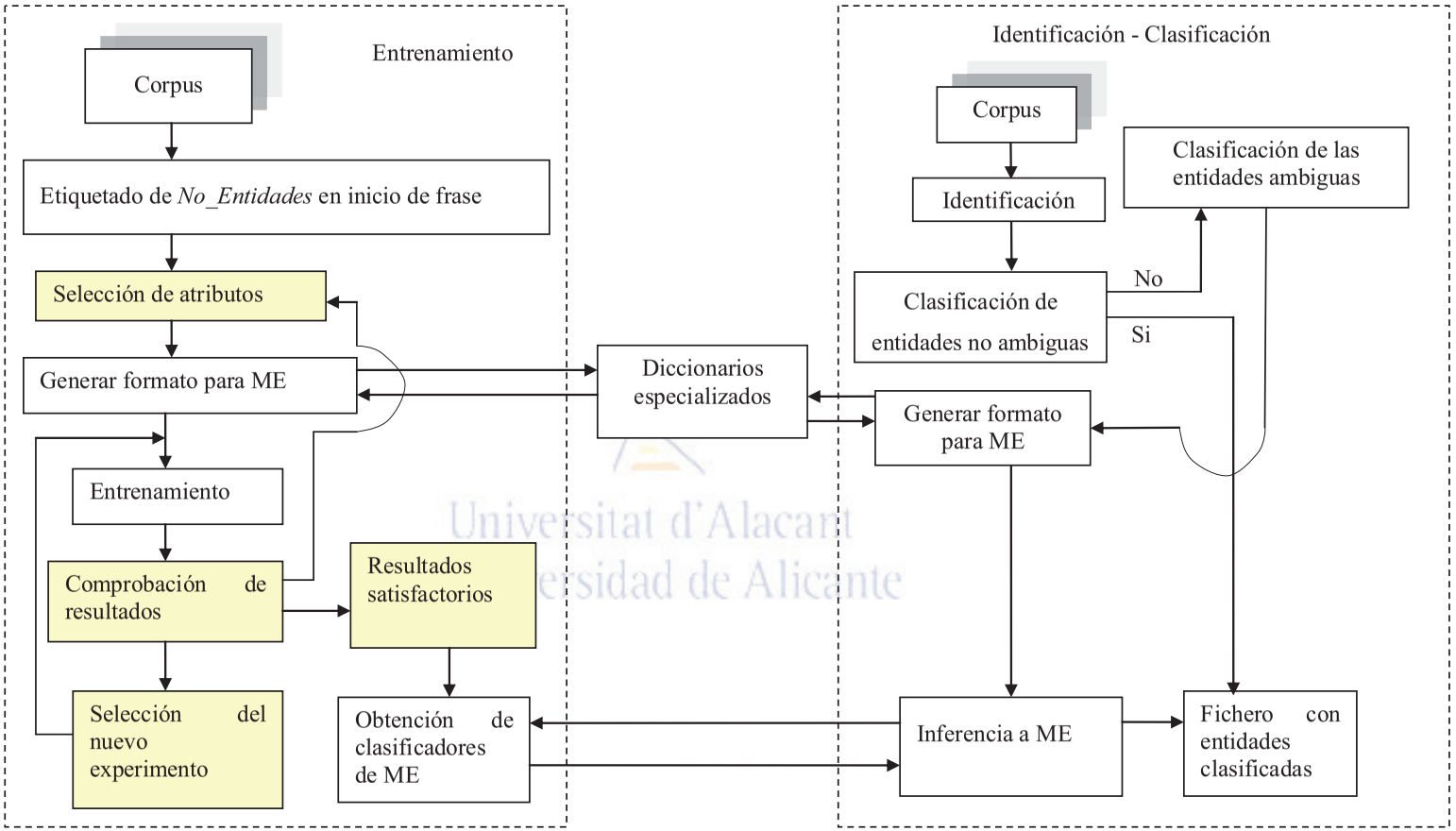


Figura 3.20. Esquema general del proceso de clasificación de entidades

Los principales problemas en esta etapa que se solucionan en este trabajo son:

- **La tokenización del corpus**

Un problema importante a resolver es la tokenización (separar por palabras una frase) del texto. Generalmente el tokenizado se realiza por espacios, aprovechando que es precisamente este elemento el que hace la división en palabras. Aquí se puede encontrar que es necesario, primeramente, hacer un pre-proceso para la preparación del texto a tokenizar, esto es debido a la presencia de los signos de puntuación, los que siempre son colocados al final de una palabra y seguidos de un espacio. Si no son separados de las palabras a las que acompañan, los signos de puntuación formarían parte de los tokens, elemento este que puede entorpecer el proceso de clasificación. A continuación se muestra un ejemplo.

```
Según Viana Baptista, Telefónica ya obtuvo la autorización de los
órganos reguladores de Argentina, Perú, y de la bolsa de Nueva
York.
```

Partiendo del texto anterior se procede a la separación de los signos para luego poder tokenizar por los espacios. El texto quedaría de la siguiente forma.

```
Según Viana Baptista , Telefónica ya obtuvo la autorización de los
órganos reguladores de Argentina , Perú , y de la bolsa de Nueva
York .
```

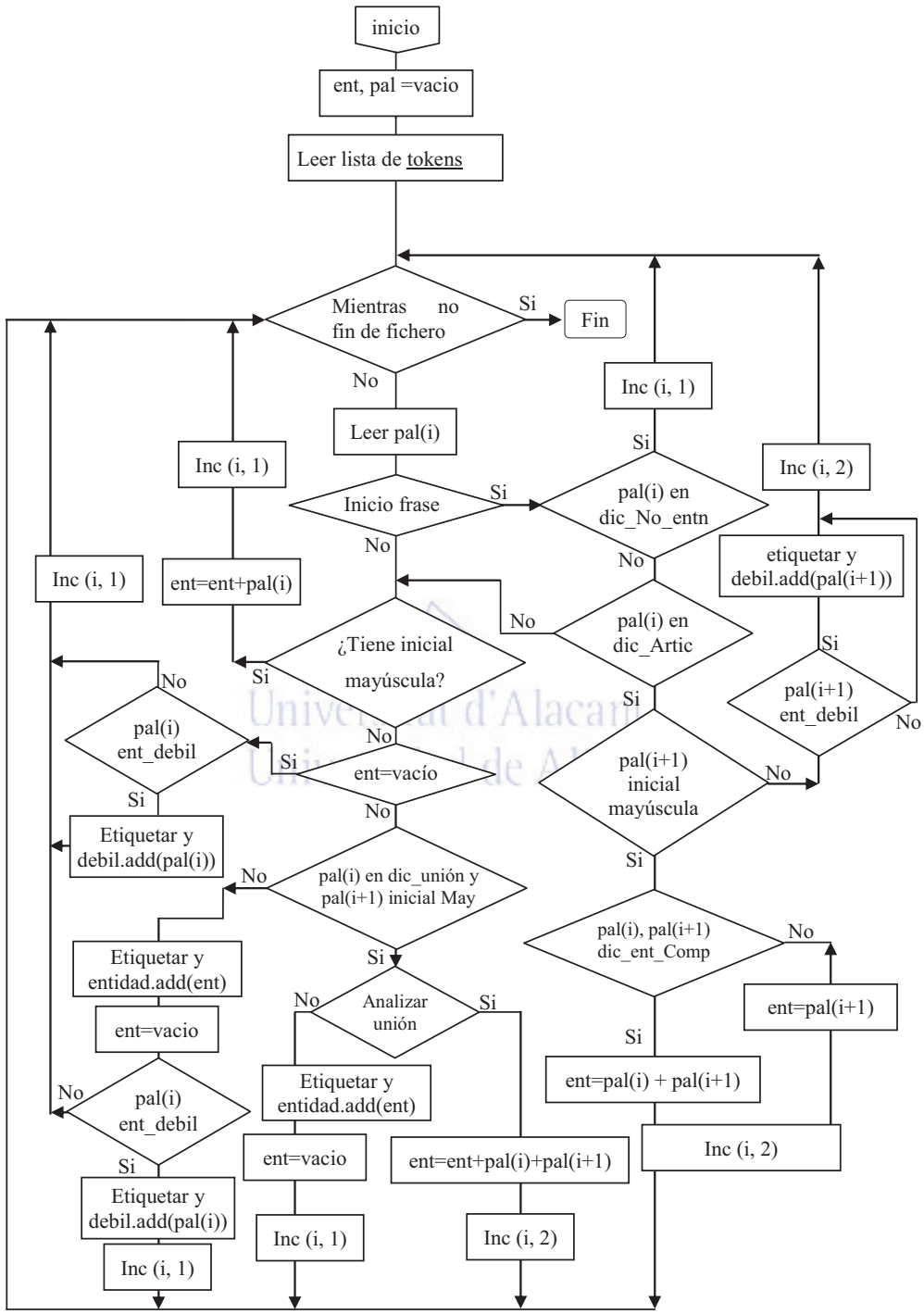


Figura 3.21. Algoritmo para determinar la longitud de la entidad y reconocer entidades débiles.

Al realizar el tokenizado se tendría la siguiente colección de palabras:

Tabla 3.50. Texto tokenizado

Según	los	la
Viana	órganos	bolsa
Baptista	reguladores	de
,	de	Nueva
Telefónica	Argentina	York
ya	,	.
Obtuvo	Perú	
la	,	
autorización	y	
de	de	

Como se muestra en la Tabla 3.50 los signos forman parte de los tokens, esto es de utilidad para algunos análisis, como por ejemplo la determinación de la longitud de las frases.

Como es de esperar, con todos los signos no se puede proceder de la misma forma, hay que hacer un tratamiento diferenciado para muchos signos como por ejemplo: las comillas, los signos de exclamación o de admiración. En estos casos es necesario hacer un análisis de como insertar los espacios. Debe recordarse que una frase o palabra entre comillas lleva las comillas pegadas a la parte delantera de la primera y a la parte posterior de la última, por ejemplo: “La multinacional española Telefónica”. Así sucede con los demás signos mencionados. Para poder tokenizar correctamente esta frase, se tiene que insertar un espacio entre la comilla inicial y la primera palabra de la frase, luego habrá que colocar el espacio delante de la última comilla. De esta forma los signos de puntuación pasarían a ser un token independiente. De lo contrario quedarían unidos a las palabras al hacer la tokenización por espacio.

Infelizmente, esto no es así para todos los casos, pues existen ocasiones en las que no se pueden insertar espacios en los signos, un ejemplo de ellos se ve en los números en los que se muestra la parte decimal o en los que se usa un signo como separador de miles, ejemplo: 125,234 ó 12.75. En estos casos hay que detectar que estos signos no pueden ser separados y por lo tanto pertenecerán al token en cuestión.

Otros caso que hay que tener en cuenta es el signo de punto, cuando es usado en las abreviaturas. Este también es un ejemplo de un signo que no puede ser separado de la palabra.

Además, es imprescindible analizar de forma particular el punto como un signo de puntuación especial, tal es el caso de los tres puntos suspensivos. En este caso tampoco se pueden separar los signos y se hace verdaderamente difícil diferenciar todas estas variantes de utilización del signo de punto.

En esta investigación se realiza un algoritmo basado en reglas para resolver esta problemática. A continuación se verá otro de los problemas acuciantes de esta etapa.

- **Inicio de frase.**

Debido a que la estrategia de esta etapa se basa, principalmente, en el estudio de las palabras que aparecen con su letra inicial en mayúscula, es un problema adicional que el principio de una frase siempre debe aparecer en mayúscula. Esto introduce un problema más, pues habría ocasiones en las que la palabra en análisis no sea una entidad propiamente, sin embargo tiene inicial mayúscula. En otros casos las palabras que a pesar de no ser una entidad, puede formar parte inseparable de otra, como una entidad compuesta por más de una palabra. Por ejemplo: `La Habana, La Coruña`. Esto provoca que al aparecer los artículos en el inicio de una frase, haya que analizar si forman parte de la entidad que le sucede.

Esta dificultad que se acaba de explicar, podría parecer un problema sencillo, pero existen realidades que demuestran lo contrario.

Además, también se puede presentar la situación en la que dos palabras -que no son entidades- se encuentren en inicio de frase, pero que sí formen parte de la entidad, por ejemplo: `De la Rosa es un empresario exitoso...`, esto indica que el problema no es tan sencillo como parece, pues será necesario determinar si más de una palabras -que además no son entidad- forman parte de ella.

Por ese motivo, se decide marcar las palabras -en inicio de frase- presentes en los documentos que se utilizan para la fase entrenamiento, con una etiqueta. Con esto, en la fase de clasificación se puede aprender a detectar cuáles palabras, de las que aparecen en inicio de frase, suelen formar parte de la entidad y cuáles no. Véase el siguiente ejemplo:

`Algunos equipos de fútbol decidieron hacer la pretemporada en el extranjero. La mayor temperatura del duro mes de agosto en España hizo tomar esta decisión.`

Al proceder al etiquetado de este párrafo se vería de la siguiente forma:

<ENT TYPE="NOENT">`Algunos`**</ENT>**`equipos de fútbol decidieron hacer la pretemporada en el extranjero`**. <ENT TYPE="NOENT">**`La`**</ENT>** `mayor temperatura del duro`**<ENT TYPE=>**`mes de agosto`**</ENT>** `en` **<ENT TYPE=>**`España`**</ENT>** `hizo tomar esta decisión.`

La palabra `Algunos` se encuentra con su letra inicial en mayúscula, sin embargo no es una entidad, por lo que recibe como etiqueta `NOENT`. Así también ocurre con el artículo `La`, que aparece después del punto y seguido. Sin embargo, es bueno destacar que no siempre ocurre así, existen ocasiones en las que, por ejemplo, un artículo forma parte de la entidad. En ese caso se encuentran algunos nombres de ciudades. A continuación se aborda un poco más en detalle esta problemática.

- **Palabras que forman la entidad.**

Como ya ha sido expuesto en el epígrafe 2.8.2, con las entidades del tipo `persona`, se puede apreciar que, en muchas ocasiones, aparecen otras palabras formando una entidad compuesta, en estos casos está la aparición de las preposiciones `de` y `del`, así como de los artículos `la` y `las` formando la entidad compuesta.

En las entidades del tipo `organización`, del mismo modo que para la entidad tipo `persona`, las palabras que van a formar parte de la entidad deben empezar por mayúsculas. También puede ocurrir que la entidad sea compuesta y existen algunos elementos que pudieran formar parte de esta entidad, como son -entre otros- el símbolo (`&`), la conjunción (`y`), así como también números.

Como se ha explicado en el epígrafe 2.8.2, existen muchos problemas asociados al proceso de la identificación de la entidad y de la determinación de su longitud. Algunas entidades suelen estar compuestas por solo una palabra, pero ya se ha visto que también existen muchas compuestas por más de una palabra.

El caso más complejo se presenta cuando aparecen dentro de la entidad, palabras que no se inician con mayúscula, como es el caso de algunos nombres propios, por ejemplo: `María de los Ángeles`. Existen además otras aún más complejas como: `Margarita del niño Jesús González`.

Resumiendo, la causa principal de este problema es que las entidades pueden estar formadas por una o varias palabras. Si la entidad está formada por varias palabras no necesariamente todas ellas tienen que comenzar en mayúscula (`Playa de Santa Lucía, Ramón y Cajal`). O sea, existen palabras como las conjunciones, preposiciones, etc., que pueden formar parte de la entidad uniendo las palabras para formar una sola entidad o separándolas en dos entidades diferentes. En este caso la fase de identificación debe ser capaz de detectar estos casos particulares. A continuación se muestran otro problema y la solución que se adopta.

- **Entidad con conjunción o entidades unidas por conjunción.**

La situación de la Playa de San Juan en verano se hace insostenible. Policía y Ayuntamiento pretenden poner en marcha un plan para frenar los robos.

Haciendo el etiquetado el texto podría quedar de la siguiente forma:

<ENT TYPE="NOENT">La**</ENT>** situación de la **<ENT TYPE="X">**Playa de San Juan**</ENT>** en verano se hace insostenible. **<ENT TYPE="X">**Policía y Ayuntamiento**</ENT>** pretenden poner en marcha un plan para frenar los robos.

Como se observa en el ejemplo anterior las organizaciones Policía y Ayuntamiento pueden ser identificadas como una única entidad cuando en realidad son dos, existen otros casos en los que la conjunción (y) aparece uniendo dos entidades, por ejemplo: Cuervo y Sobrinos; por lo que se hace necesario resolver esta ambigüedad.

La solución al problema fue aplicar un conjunto de reglas para decidir si se dividen en dos entidades o se considera como una sola. Estas reglas se basan principalmente en la aparición de parte de la entidad previamente en el documento como una entidad independiente, por ejemplo: si la entidad Policía, ya fue mencionada en el resto del documento. Otra solución es analizar, de existir disparadores, la posición que ocupan con respecto a las entidades. Si aparecen en el lado derecho o a ambos lados de las entidades, esto determinaría que son dos entidades independientes; por el contrario si el disparador aparece solo en la entidad final se trata de una sola entidad.

- **Identidades presentes en los títulos**

Generalmente los títulos y titulares aparecen totalmente escritos en mayúscula, lo que dificulta la detección de las entidades que puedan aparecer en estos. Para ello se siguen dos estrategias que pueden solucionar este problema. Por lo general las palabras que aparecen en los titulares, vuelven a aparecer en el primer párrafo, lo que permite detectarlas en esta posición y etiquetarlas luego en el titular. La otra idea que se implementa es guardar todas las entidades que sean detectadas en el texto, luego hacer una búsqueda de aquellas que no fueron detectadas en su momento e identificarlas en esta bolsa de entidades. Por lo general en un documento cuando aparecen entidades en los títulos, luego volverán a aparecer en alguna posición dentro del documento. A continuación se relacionan las entidades que son clasificadas en la fase de identificación.

3.7.2.1 Descripción de las entidades clasificadas en el módulo de identificación

En esta propuesta -en la fase de identificación- además de la identificar de las entidades, se clasifican aquellas que partiendo de reglas previamente definidas no presentan ninguna ambigüedad; o simplemente existe mucha seguridad de su posible clasificación. De esta forma se reduce la cantidad de entidades que se le enviarán al clasificador y por ende se reducirá el tiempo computacional.

Además de las entidades fuertes, se decide identificar y clasificar las de tipo:

Fechas: aquellas que corresponden con una expresión de tiempo absoluta, pueden tener cualquiera de los formatos de fecha utilizados normalmente en los textos, ejemplo: `Lunes, 24 de febrero de 2004`…, o tal vez solo: `24 de febrero`…, o simplemente: `lunes 24`. Se ha concebido la posibilidad de clasificar la mayoría de los formatos de fechas posibles de encontrar en un documento, teniendo en cuenta las diferentes culturas.

Expresiones de tiempo: se identifican un grupo de expresiones temporales deícticas que indican una marca en el tiempo o un período o espacio en el tiempo. Por ejemplo: el año entrante, este mes, el año pasado, la próxima semana, a fines de enero, para el año que viene, o sea, expresiones que de alguna forma incluyen hora, minuto, segundo, día, mes, año siglo, etc. y algunas de las palabras que aparecen en la Tabla 3.51:

Tabla 3.51. Algunas de las palabras que forman expresiones temporales deícticas.

entrante	próxima	venideros
este	próximas	a fines de
pasado	próximo	para el
pasados	próximos	en el
anterior	siguiente	hace
anteriores	siguientes	desde el (la)
dentro de	hasta el (la)	
que viene	venidero.	

Números: Serán aquellos dígitos que no hayan podido ser detectados, ni como parte de una fecha, ni de una unidad o cantidad específica, por ejemplo: …`el 13 es un número de mala suerte para muchos`.

Cantidades: Serán aquellos dígitos que seguidos de alguna palabra indican que el numeral encontrado constituye la cantidad correspondiente a esa palabra que lo acompaña, ejemplo: `24 libras, 14 Kg`, etc.

Entidad URL: serán las palabras que se encuentren formando parte de una URL, ejemplo: `ftp://anubis.umcc.cu`, `http://www.dlsi.ua.es`, `www.umcc.cu`, o simplemente: `google.com`.

Entidad Mail: Serán aquellas palabras que formen una dirección electrónica, ejemplo, `antonio.fernandez@umcc.cu`.

Siglas: Son aquellas entidades que se encuentran escritas en mayúscula rodeadas de palabras minúsculas, se les busca en un lexicón y si aparecen son clasificadas como tal. Si no están en el lexicón, se comprueba si tienen entre dos y cinco caracteres y parecen tres consonantes seguidas en algún lugar de la palabra. De no cumplirse estas premisas no se clasificará como sigla.

Es importante señalar que este sistema de detección de entidades se aplica a textos en su estado original, es decir, sin ningún pre-procesamiento, ni etiquetado previo. En la mayoría de los trabajos consultados, la detección de entidades es precedida de algún análisis léxico-sintáctico o semántico. En este caso se ha decidido no utilizar este tipo de recursos, lo que hace algo más difícil esta tarea para su desarrollo, pero por otra parte permite ahorrar tiempo y disminuir carga computacional, lo que facilita una respuesta más rápida.

Aún cuando todas estas entidades se sometieron a un proceso de depuración y comprobación de resultados, no se tendrán en cuenta en los cálculos de la evaluación de la fase de identificación. Esto es debido a que, en su mayoría se detectan a partir de patrones creados mediante expresiones regulares, los que tienen una alta eficiencia y precisión. En segundo lugar porque estas entidades no han sido estandarizadas como las entidades fuertes, por lo que no son las mismas que se reconocen en un sistema u otro. Por último, es válido decir que en CONLL2002, competición con la que se hacen las comparaciones, como en otras más recientes, no se etiquetan otras entidades que no sean las entidades fuertes. Este elemento, dificulta mucho el análisis comparativo de cualquier otro tipo de entidad.

Por esta razón, la identificación de las entidades fuertes (Persona, Localidad, Organización y Misceláneas), son las que se verifican, obteniéndose los resultados de Precisión, Cobertura y F-Medida en cada uno de los conjuntos de prueba evaluados en la fase de identificación.

Para este proceso de identificación, utilizando los conjuntos de prueba `esp.testa` y `esp.testb`[62] (ver características en la Tabla 3.52), se obtienen los resultados que se muestran en la Tabla 3.53 y Tabla 3.54.

[62] http://www.cnts.ua.ac.be/conll2002/ner/data/

Tabla 3.52. Características del conjunto (corpus) de prueba `esp.testa`.

Corpus	Total Palabras	LOC	ORG	PER	MISC
esp.testa	52922	984	1676	1211	403
esp.testb	51532	1084	1400	735	339

Tabla 3.53. Resultados de la fase de identificación con el conjunto de prueba `esp.testa`.

Entidad	eic	eii	eni	Precisión	Cobertura	F-medida
Localidad	942	14	28	0.985400000	0.95730000	0.97110000
Organización	1548	97	27	0.941033435	0.925837321	0.93337353
Persona	1148	61	11	0.958263800	0.949545100	0.95388450
Miscelánea	372	21	10	0.946564885	0.923076923	0.934673367

Tabla 3.54. Resultados de la fase de identificación con el conjunto de prueba `esp.testb`.

Entidad	eic	eii	eni	Precisión	Cobertura	F-medida
Localidad	697	14	22	0.980309423	0.950886767	0.965373961
Organización	681	37	15	0.948467967	0.929058663	0.938662991
Persona	667	62	4	0.914952	0.909959	0.912449
Miscelánea	274	17	2	0.941580756	0.935153584	0.938356164

Como se puede ver en las Tabla 3.53 y Tabla 3.54, en el proceso de identificación, aunque se logran resultados relevantes, siempre ocurren un grupo de errores que afectan la evaluación general del sistema.

En realidad no se han podido encontrar muchos artículos en los que se refleje claramente, por separado, las dos fases de este proceso (identificación o reconocimiento y clasificación). La mayoría de los artículos se concentran más en los resultados generales de precisión y cobertura (overall precision and recall).

Dicho esto, se estima que los resultados alcanzados son aceptables si se comparan con el mejor resultado, obtenido un año después en (Ferrández, Kozareva et al., 2005): precisión= 92.51%, cobertura= 92.61%, f-medida= 92.56%. Así como los obtenidos en el 2010 en (Toribio, Martínez et al., 2010): precisión= 78,60%, cobertura= 51,40%, f-medida= 62,16%. Otros trabajos, en los que también se reportaban los valores de precisión y cobertura para esta fase, no se utilizan porque fueron obtenidos sobre otros corpus (Lee, Hwang et al., 2003; Tsuruoka y Tsujii, 2004).

Para resolver una buena parte de la identificación de las identidades, se usan un grupo de lexicones con diferentes elementos. En el siguiente epígrafe se describe la construcción y los tipos de estos diccionarios especializados.

3.7.2.2 Diccionarios o lexicones utilizados

En la Figura 3.20 se puede ver que tanto la fase de entrenamiento, con las de identificación-clasificación, hacen uso de diccionarios especializados. Para el funcionamiento de este método se

confeccionan varios lexicones o diccionarios que ayudan en ambas fases del método, muchos se construyen de forma semi-automática, aunque la gran mayoría se crea de forma manual.

El primero de esto lexicones, ya se había comentado antes, contiene un grupo de palabras que usualmente aparecen en inicio de frase, pero que no son entidades. Se elabora haciendo un estudio estadístico de varios corpus del español, extrayendo las palabras en inicio de frase y posteriormente comprobando manualmente si en realidad no son entidades.

El segundo de los lexicones, fue creado con todas las palabras que con mayor regularidad aparecen en medio de palabras en mayúscula formando una entidad compuesta, este es el caso de algunos artículos, preposiciones, etc.

Posteriormente se crean otros lexicones más, los que contienen un conjunto de las llamadas entidades débiles. En ellos se agrupan, los llamados disparadores. Pueden ser disparadores de personas como, por ejemplo, profesiones, oficios, etc. También se agrupan los disparadores de localidades, como: accidentes geográficos, palabras que denotan un lugar específico, etc. Por último se agruparon aquellas palabras que dan las pistas para identificar entidades de tipo organización, como las palabras: compañía, empresa, etc.

También, se crean muchos otros lexicones con entidades débiles como: días de la semana, meses del año, números cardinales, ordinales, siglas, onomatopeyas, nombres de dominios, abreviaturas y de entidades compuestas, que normalmente aparecen con un artículo, así como de palabras que normalmente enlazan entidades.

En general se crean 24 lexicones, 18 de ellos se usan en la fase de identificación, 11 en la fase de clasificación y 10 en la generación de atributos para el clasificador. En la Tabla 3.55 se recogen los tipos y la cantidad de entradas que tienen cada uno.

Tabla 3.55. Tipos de lexicones creados y cantidad de elementos en cada uno

Tipo de lexicón	Entradas	Tipo de lexicón	Entradas
Períodos	20	Evidencias internas de Organización	40
Números ordinales	41	Disparadores de Localidad	221
NO entidades	559	Organizaciones	584
Meses	48	Días cardinales	31
Unidades de tiempo	24	Días de la semana	30
Expresiones temporales	22	Conjunciones	23
Extensiones de Dominios	250	Números cardinales	135
Disparadores de Persona	674	Entidades compuestas	21
Nombres de personas	10496	Siglas de moneda	158
Evidencia Interna de Persona	218	Siglas de países	435
Evidencias externas de Persona	614	Unidades de medida	435
Disparadores de Organización	70	Localidades	921

Finalmente, una vez concluida la fase de identificación, se pasa a la de clasificación. A continuación se recogen los detalles de esta etapa.

3.7.3 La fase de clasificación

En esta fase se clasifican todas aquellas entidades que han sido identificadas en la fase anterior y que no hayan podido ser clasificadas por presentar determinada ambigüedad. Generalmente, estas entidades son las entidades fuertes. Para esta clasificación, se utilizan los clasificadores obtenidos a partir de los modelos de Probabilidad Condicional de Máxima Entropía. Un modelo que, una vez entrenado debidamente, es capaz de clasificar las entidades dividiéndolas en cinco clases: Personas, Localidades, Organizaciones, Misceláneas, No Entidad.

En el modelo realizado en este trabajo, reviste gran importancia la posición de la no entidad en inicio de frase. Es por ello que se incorpora un atributo con esta finalidad, de forma tal que el modelo sea capaz de detectar cuándo las palabras que están en posición inicial de frase son o no una entidad.

Descripción de las entidades a clasificar

La entidad Persona (PER): corresponde a todos los nombres propios pertenecientes a una persona física.

Para realizar una correcta clasificación de este tipo de entidad fue necesario, en el entrenamiento, obtener una serie de atributos que caracterizan este tipo de entidad. Por ejemplo, se analiza si las palabras que forman parte de la entidad se hallan en los diccionarios de nombres y apellidos, elaborado para ser consultados por el sistema. Para el reconocimiento de personas se utilizan también los llamados disparadores, que también se encuentran en diccionarios.

La entidad Organización (ORG): corresponde a aquellos nombres que pertenecen a una empresa, compañía u otro tipo de organización. También son una pista importante para la clasificación de entidades de tipo organización, algunos disparadores que describen la actividad que realizan, por ejemplo: `Clínica`, `Hospital`, `Bar`, `Restaurante`, etc.

La entidad Localidad (LOC): será aquélla compuesta por nombres de lugares geográficos, por ejemplo: playas, provincias, países, etc. Para el reconocimiento de direcciones se utilizan también los disparadores almacenados en diccionarios.

La entidad Miscelánea (MISC): pertenecerán aquellas entidades que no han podido ser incluidas en algunas de las clasificaciones anteriores, pero que de alguna forma están escritas con inicial

mayúscula, constituyendo así una entidad. En este caso se encuentran los títulos de una obra de teatro de una película o una canción, etc.

La clasificación de No entidad (NOENT)**:** corresponde a aquellas palabras que se encuentran en inicio de frase y que como es lógico se escriben con mayúscula, pero que no son una entidad, como en el ejemplo anterior: `La situación de la Playa de San Juan`…, en este caso el artículo `La` hay que etiquetarlo como `No entidad`. No así en el caso: `La Habana`, en el que el artículo aparecería formando parte de la entidad.

A pesar de que la fase de clasificación hace uso de los disparadores y otros recursos almacenados en lexicones, esto no constituye el elemento por el cual se hará la clasificación de las entidades. Dichos elementos, solo se toman para extraer los atributos correspondientes, que serán los encargados de mostrar al clasificador las características de la entidad que se analiza. En la siguiente fase se estará comentando sobre el entrenamiento de los clasificadores utilizados.

3.7.4 La fase de entrenamiento

En el epígrafe 2.8, queda esclarecido que el proceso de reconocer una entidad en un texto pasa por dos fases, la identificación y la clasificación. Pero a su vez, los sistemas basados en aprendizaje automático necesitan de dos momentos también. La primera etapa de un sistema de aprendizaje es el entrenamiento, ya en la segunda, después de entrenado, el sistema podrá hacer las inferencias y ofrecer respuesta.

El método ideado en esta investigación para el reconocimiento de entidades pasa también por estas fases, con la diferencia de que algunas entidades se clasifican en la fase de identificación.

Como se muestra en la Figura 3.20, la etapa de entrenamiento está representada a la izquierda del gráfico y a la derecha las fases de identificación y calificación.

Para la fase de entrenamiento, se parte de los corpus de CoNLL-2002, los que se utilizan para extraer los atributos necesarios. Esto permite hacer los ajuste para formar el conjunto de entrenamiento para el sistema de ME.

Esta etapa (ver Figura 3.20), comienza con la lectura del corpus de entrenamiento, el que posee las entidades fuertes debidamente etiquetadas por expertos humanos.

El siguiente paso consiste en etiquetar todas las palabras que se encuentran en el inicio de frase y no son una entidad. Este proceso se realiza de forma semiautomática usando un lexicón compuesto por palabras que no son entidades y que con frecuencia aparecen como inicio de frase. Se utiliza

además un algoritmo que etiqueta estas palabras con el identificador: NOENT. De esta forma ya el corpus queda listo para el entrenamiento.

Selección de los atributos

Es en este momento, es donde se hace necesario realizar una selección de los atributos que serán utilizados por el entrenamiento de los modelos de ME. Esto se realiza prácticamente por intuición, a partir de un estudio de la aparición de las entidades en los textos y las características que llevarían a su posible clasificación (véase epígrafe 2.8.2). El sistema de aprendizaje seleccionado, usa un conjunto de propiedades o rasgos que permiten definir cada tipo de entidad. Estas propiedades aportan información de las palabras que forman la entidad y son llamadas evidencias internas y externas (McDonald, 1996).

Para este caso, se toman atributos que contienen cadenas de caracteres o valores binarios (ver ANEXO 7 con la relación de los atributos seleccionados). Para esta fase, en la Figura 3.20 se puede ver, en un recuadro amarillo, las operaciones que son realizadas manualmente, o sea, sin la intervención del ordenador.

Los atributos utilizados son 41, los que se agrupan de la siguiente manera:

- Palabras que forman la entidad.
- Posición que ocupa en la frase la primera palabra de la entidad.
- Las palabras anteriores y posteriores a la entidad en una ventana de ± 3.palabras.
- Pertenencia o no de la entidad o parte de esta a diccionarios específicos.

Esta última, es un conjunto de características que valoran si la entidad en su totalidad o parte de ellas, está en un conjunto de diccionarios especializados (diccionario de nombres de personas, diccionario de lugares, diccionario de empresas o en listas de disparadores específicos para cada tipo de entidad). También se estudia si las palabras que rodean a la entidad pertenecen a alguno de los diccionarios de disparadores.

Una vez seleccionados los atributos para el entrenamiento, se pasa el corpus al algoritmo que genera las líneas que interpreta el sistema de ME. Al concluir este proceso comenzará el entrenamiento invocando el recurso MXEC.exe (Suárez, 2004), pasando como parámetros el corpus y la lista de atributos.

Descripción de las líneas que serán pasadas a MXEC para clasificar las entidades:

```
<u>id:Entidad</u><l>Lista de atributos</l><c>Etiqueta de Clase</c>
```

donde:

<u>id:Entidad</u> → Identificador de la línea en análisis.

En la que:

id → Es una pareja de números que indican la posición de la palabra, el primer elemento es el número del párrafo y el segundo es la posición de la palabra en la lista de tokens, ejemplo: 1,1 sería una palabra que está en el primer párrafo en la posición uno de la lista.

: → Separador entre id y Entidad.

Entidad → Entidad en análisis.

<l>Lista de Atributos</l> → Lista de atributos seleccionados.

Los atributos estarán numerados y tendrán la siguiente estructura:

#A01:= → Esta es la forma de colocar el primer atributo, indicado por la numeración 01, después del signo de igual se colocaría el valor del atributo, sin dejar espacio. Ej. #A01:=1

Ejemplo. <l>#A01:=Pepe#A02:=1#A03:=</l> → Estos sería una lista con sólo tres atributos y un tercero vacío.

<c>Etiqueta de Clase</c> → Indicador de clase. En el entrenamiento se coloca la clase que tenga la entidad en análisis, para las inferencias como no se conoce la clase está etiqueta permanece vacía. Ejemplo: <c>L</c> → Esto sería una clase L, ósea, Localidad.

A continuación se muestra un ejemplo con una pequeña porción del corpus esp.testa:

- Ejemplo: primera línea del corpus esp.testa ofrecido por CoNLL2002.

Sao Paulo (Brasil), 23 may (EFECOM).

Esta línea estaría etiquetada en el corpus original (e.g. CoNLL2002) así:

```
Sao B-LOC
Paulo I-LOC
( O
Brasil B-LOC
) O
, O
23 O
may O
( O
EFECOM B-ORG
) O
. O
```

Estos corpus se encuentran anotados con la codificación BIO. La `B` indica que comienza una entidad, por lo que las etiqueta `B-LOC` indican que comienza una entidad de tipo `Localidad`. La letra `I` indica que es parte de la entidad que comenzó con la `B`. Ejemplo, la etiqueta `I-LOC` es colocada en cada una de las palabras que forman parte de la entidad, en este caso `Localidad`. De esa forma se colocan todas las demás etiquetas.

- B-PER: para indicar inicio de entidad Persona.
- B-ORG: para indicar inicio de entidad Organización.
- B-MISC: para indicar inicio de entidad Miscelánea

La etiqueta `O`, será usada para designar las palabras que no son entidad.

De esta forma la línea que se genera para el entrenamiento de los modelos de ME, para la primera entidad que aparece en el ejemplo anterior, sería la siguiente:

```
<u>1,1:Sao Paulo</u><l>#A01:=Sao
Paulo#A02:=0#A03:=0#A04:=0#A05:=0#A06:=0#A07:=0#A08:=0#A09:=0#A10:
=0#A13:=0#A14:=0#A15:=0#A16:=(#A17:=Brasil#A18:=)#A19:=1#A20:=0#A2
1:=0#A22:=0#A23:=0#A24:=1#A25:=1#A26:=0#A27:=0#A28:=0#A29:=0#A30:=
0#A31:=0#A32:=0#A33:=0#A34:=0#A35:=0#A36:=0#A37:=0#A38:=0#A39:=0#A
40:=0#A41:=0</l><c>L</c>
```


La línea que recibe ME, pero para el proceso de inferencia quedaría de la siguiente forma:

```
<u>1,1:Sao Paulo</u><l>#A01:=Sao
Paulo#A02:=0#A03:=0#A04:=0#A05:=0#A06:=0#A07:=0#A08:=0#A09:=0#A10:
=0#A13:=0#A14:=0#A15:=0#A16:=(#A17:=Brasil#A18:=)#A19:=1#A20:=0#A2
1:=0#A22:=0#A23:=0#A24:=1#A25:=1#A26:=0#A27:=0#A28:=0#A29:=0#A30:=
0#A31:=0#A32:=0#A33:=0#A34:=0#A35:=0#A36:=0#A37:=0#A38:=0#A39:=0#A
40:=0#A41:=0</l><c></c>
```

Como se puede ver, en este caso es igual a la que se usaría en el entrenamiento, con la ligera diferencia de que no se colocará la clase.

La Tabla 3.56 muestra las clases que se utilizaron para el entrenamiento de los Modelos de Máxima Entropía.

Tabla 3.56. Clases para los entrenamientos de ME

Clases	Etiqueta en el corpus	Etiqueta en ME
Persona	PER	P
Localidad	LOC	L
Organización	ORG	O
Miscelánea	MISC	M
No Entidad (en posición inicial de frase)	NOENT	N
No entidad (en otra posición diferente)	O	No será considerada clase

Para poder obtener el clasificador entrenado, se hacen varios entrenamientos con el objetivo de verificar la calidad de los atributos seleccionados; es de destacar que cada proceso de entrenamiento, con un juego de atributos seleccionados (41), dura varias horas.

Como es necesario comprobar la calidad de los atributos seleccionados, a continuación se muestra el proceso que se sigue para esta validación.

3.7.4.1 Evaluación de la influencia de los atributos seleccionados

Aquí se llevan a cabo varias pruebas para comprobar la influencia de los atributos seleccionados. Todos los experimentos se realizaron utilizando el corpus `esp.train` como conjunto de entrenamiento y los corpus `esp.testb` y `esp.testb` como conjunto de evaluación. Aunque también se hacen otras combinaciones entre ellos. Se efectuaron así, las pruebas:

- Eliminación gradual de un atributo y entrenamiento con los atributos restantes.

Esta prueba permitió validar la influencia que tendría en el proceso de entrenamiento la eliminación de cada atributo de forma independiente. Después de concluida dicha prueba, se pudo comprobar que no existen marcadas diferencias en la influencia de los distintos atributos probados. Se obtiene como promedio una precisión de 0.77 y una cobertura de 0.76 al eliminar un atributo y utilizar los restantes.

- Entrenamiento utilizando como atributo la entidad y las evidencias internas.

Utilizando sólo la entidad y las evidencias internas se obtuvieron resultados de precisión de 0,54 y cobertura de 0.52; por lo que se puede comprobar la marcada influencia que ejercen estos parámetros en el entrenamiento del sistema.

- Entrenamiento utilizando como atributo la entidad y las evidencias externas.

Utilizando sólo la entidad y las evidencias externas se obtuvieron resultados de precisión de 0.31 y cobertura de 0.31. Se puede ver que las evidencias externas no aportan lo mismo al proceso de clasificación que las internas -como era lógico de esperar- pero tampoco es despreciable su influencia, pues si se prescinde de ellas los resultados descienden significativamente.

Se puede apreciar que, para estos corpus en específico, según se muestra en los experimentos dos y tres, existe una mayor influencia de las evidencias internas que las externas a la hora de clasificar las entidades. Esto no es una regla ya que depende mucho del tipo de corpus.

- Entrenamiento utilizando solo un atributo.

Con esta prueba se puede analizar la influencia individual de un determinado atributo dentro del proceso de entrenamiento, ya que se utiliza solo este atributo para alcanzar los resultados. Esta prueba es de gran importancia para la eficiencia del método. Al hacer una evaluación individual de cada atributo se pueden optimizar el conjunto, eliminando aquellos que no ejerzan una influencia determinante, reduciendo con esto los tiempos de entrenamiento y posterior ejecución del algoritmo.

Una vez concluida la prueba, se determina que en su totalidad los atributos tienen un gran aporte en los resultados, siendo los de de mayor influencia: A1, A14, A15, A16, A17, A18, A20, A21, A22, A23, A24, A25, A26, A29, A30, A31, A32, A33, A34, A35, A36, A37, A38, A39, A40 (ver ANEXO 7)

El atributo de más influye es A1 (la entidad propiamente) y luego le siguen desde A13 hasta A18 (palabras que rodean la entidad), siendo A14 (segunda palabra a la izquierda de la entidad) el de mayor influencia entre ellos.

Una vez comprobada la influencia de cada atributo, se realizan varios experimentos para ver el funcionamiento general del método.

- Experimento usando Ten Fold Cross Validation (TFCV).

Se realiza la división del corpus en diez partes, se tomó una de estas partes para evaluar y el resto para realizar el entrenamiento. De esta forma se obtiene un conjunto de elementos de gran importancia para la investigación.

Leyenda:

E → Cantidad de palabras evaluadas (son palabras que comienzan con inicial mayúscula, por lo que son candidatas a entidad).

A → Cantidad de palabras acertadas.

F → Cantidad de palabras falladas.

Utilizado esta técnica de TFCV, se efectúan dos experimentos, uno con el corpus `esp.testa` y la otra con `esp.train`. En las tablas Tabla 3.57 y Tabla 3.58 se muestras los resultados.

Tabla 3.57. Prueba de TFCV sobre corpus `esp.testa`.

No	E	A	F	Precisión.	Cobertura
1	532	399	93	0.810976	0.75
2	532	384	99	0.795031	0.721805
3	532	389	107	0.784274	0.731203
4	532	417	72	0.852761	0.783835
5	532	407	89	0.820565	0.765038
6	532	422	68	0.861224	0.793233
7	532	374	117	0.761711	0.703008
8	532	398	93	0.810591	0.74812
9	532	398	93	0.810591	0.74812
Valor medio				0.811969	0.749373

Tabla 3.58. Prueba del TFCV sobre el corpus `esp.train`.

No	E	A	F	Precisión	Cobertura
1	2343	2124	196	0.915517	0.90653
2	2343	2165	162	0.930382	0.924029
3	2343	2017	306	0.868274	0.860862
4	2343	2132	194	0.916595	0.909945
5	2343	2128	185	0.920017	0.908237
6	2343	2071	247	0.893443	0.88391
7	2343	2145	173	0.925367	0.915493
8	2343	2112	210	0.909561	0.901408
9	2343	2152	178	0.923605	0.918481
Valor medio				0.911418	0.903211

Para finalizar se realiza un último experimento, donde se toma cada una de las partes de la prueba anterior (TFCV sobre `esp.train`) y se realiza la evaluación con el corpus `esp.testa`.

Tabla 3.59. Evaluación del corpus `esp.testa` con las diferentes partes del `esp.train`.

No	E	A	F	Precisión	Cobertura
1	5323	4305	570	0.883077	0.808754
2	5323	4294	581	0.880821	0.806688
3	5323	4288	587	0.87959	0.805561
4	5323	4303	572	0.882667	0.808379
5	5323	4308	567	0.883692	0.809318

6	5323	4305	570	0.883077	0.808754
7	5323	4298	577	0.881641	0.807439
8	5323	4278	597	0.877538	0.803682
9	5323	4296	579	0.881231	0.807064
Valor medio				0.881481	0.807293

El siguiente paso es proceder al entrenamiento de los clasificadores definitivos, los que se utilizarán para las próximas inferencias. Este proceso no es más que la iteración que hace el algoritmo por todo el corpus etiquetado, intentando obtener los pesos asociados a los enlaces entre los elementos relacionados (ver explicación detallada en el epígrafe 2.8.1).

Una vez que se ha entrenado el clasificador, se procede a la comprobación de resultados. Si todo está bien (nunca es así), entonces ya se tiene el clasificador listo para la siguiente etapa. Si no se obtienen los resultados esperados, entonces se trabaja recursivamente rectificando los posibles errores, ajustando los parámetros y regresando nuevamente al paso del entrenamiento. Este proceso se repite hasta que se obtengan los resultados deseados.

Una vez concluido el entrenamiento, ya puede ser utilizado para el proceso de clasificación de las entidades. Antes, es necesario `identificar` las entidades que posteriormente van a ser clasificadas. Este proceso, aunque se le dedica menos espacio en los artículos y publicaciones sobre REN, tiene varios detalles que son importantes para obtener un buen resultado. En el siguiente epígrafe se muestra con más detalle esta etapa. En el epígrafe 3.7.5 se muestran los diferentes experimentos realizados para el entrenamiento de los clasificadores.

3.7.5 Experimentación con los corpus del CONLL-2002 usando Máxima Entropía

Para realizar los experimentos se implementa un prototipo (Fernández y Muñoz, 2005) el que, a través de la utilización de dos recursos McxE.exe y McxEE.exe (Suárez, 2004), permite el reconocimiento de las entidades en un texto no etiquetado. Para este proceso es necesario un algoritmo que toma los corpus de CONLL-2002: `esp.train, esp.testa y esp.testb`, y los transforma (ver explicación en epígrafe 3.7.4) para generar un texto en el formato en que puede ser leído por los recursos antes mencionados.

3.7.5.1 Resultados de los entrenamientos y evaluación con Máxima Entropía

Luego de realizar todos los experimentos planificados para la comprobación del método de REN, los resultados son tabulados y se resumen a continuación.

El primer experimento se realiza entrenando y evaluando con el corpus `esp.testa`, en este caso se comprueba el método evaluando instancias que ya conoce.

Los resultados se muestran en la Tabla 3.60.

Tabla 3.60. Entrenamiento y evaluación con el corpus `esp.testa`.

Cant. Palabras	A	F	Precisión	Cobertura
5323	4874	1	0.999795	0.915649

Leyenda:

Cant. Palabras → Son todas las palabras que se identificaron como posibles entidades por poseer inicial mayúscula.

A → Cantidad de palabras acertadas.

F → Cantidad de palabras falladas.

Los resultados mostrados en la Tabla 3.60 son muy bueno, pero no pueden ser tomados en cuenta, debido a que el clasificador está haciendo inferencias sobre ejemplos con los que aprendió en la fase de entrenamiento, ya que se usa el mismo conjunto de entrenamiento como conjunto prueba. Este y el siguiente resultado, solo se usan para comprobación y valoración del correcto funcionamiento del algoritmo.

Tabla 3.61. Entrenamiento y evaluación con el corpus `esp.testb`.

Cant. Palabras	A	F	Precisión	Cobertura
4567	4128	81	0.980756	0.903876

En este caso (Tabla 3.61) ocurre lo mismo que en el caso anterior, sin embargo los resultados no son tan precisos. Estos valores tampoco son tomados en consideración.

El siguiente resultado es obtenido después de entrenar el clasificador con el corpus `esp.testa` y luego evaluar con `esp.testb`. Como se puede ver ya los valores comienzan a cambiar debido a que se están evaluando instancias con las que el clasificador no se entrenó. Esto permite apreciar su capacidad de generalización.

Tabla 3.62. Entrenamiento con `esp.testa` y evaluación con `esp.testb`.

Cant. Palabras	A	F	Precisión	Cobertura
4567	3492	717	0.829651	0.764616

A continuación se realizan las pruebas definitivas para la comprobación del método. En este caso se ha entrenado con el conjunto de entrenamiento `esp.train` y se realiza la comprobación con `esp.testa` (ver Tabla 3.63). Esta variante se utilizará para medir los resultados con los presentados en CONLL2002, pues así se midieron los equipos en esta competición.

Tabla 3.63. Entrenamiento con `esp.train` y evaluación con `esp.testa`.

Cant. Palabras	A	F	Precisión	Cobertura
5323	4298	577	0.881641	0.807439

Los resultados de la Tabla 3.64, también se obtienen usando el conjunto de entrenamiento `esp.train`, pero en este caso se comprueba con `esp.testb`. Como se puede apreciar en ambas tablas los resultados obtenido son prometedores.

Tabla 3.64. Entrenamiento con `esp.train` y evaluación con `esp.testb`.

Cant. Palabras	A	F	Precisión	Cobertura
4567	3684	525	0.875267	0.806656

En la Tabla 3.65 se hace una comparación de estos resultados con relación a los obtenidos en la competición del CONLL202 por diferentes investigadores. Se puede ver que ambas variantes obtienen los mejores resultados, ubicándose en las primeras posiciones del ranking de esta competición.

Tabla 3.65. Resultados para la prueba con español comparado con el ranking del CONLL2002.

Español	precisión	cobertura	F-Medida
Variante 1 (entrenamiento con esp.train comprobación con esp.testa)	88.16%	80.74%	84.29%
Variante 2 (entrenamiento con esp.train comprobación con esp.testb)	87.53%	80.66%	83.95%
[CMP02] (Carreras, Marquez et al., 2002)	81.38%	81.40%	81.39
[Flo02] (Florian, 2002)	78.70%	79.40%	79.05
[CY02] (Cucerzan y Yarowsky, 2002)	78.19%	76.14%	77.15
[Tjo02] (Sang y Erik, 2002)	76.00%	75.55%	75.78
[WNC02] (Wu, Ngai et al., 2002)	75.85%	77.38%	76.61
[PWM02] (Patrick, Whitelaw et al., 2002)	74.32%	73.52%	73.92
[BHM02] (Burger, Henderson et al., 2002)	74.19%	77.44%	75.78
[Jan02] (Jansche, 2002)	74.03%	73.76%	73.89
[Mal02] (Malouf, 2002)	73.93%	73.39%	73.66
[Tsu02] (Tsukamoto, Mitsuishi et al., 2002)	69.04%	74.12%	71.49
[BV02] (Black y Vasilakopoulos, 2002)	60.53%	67.29%	63.73
[MM02] (McNamee y Mayfield, 2002)	56.28%	66.51%	60.97
baseline	26.27%	56.48%	35.86

Luego de experimentar con los modelos de probabilidad condicional de ME, con el objetivo de estudiar el proceso desde otra perspectiva y realizar comparaciones, se prueban otros algoritmos de aprendizajes.

En el siguiente epígrafe se muestra en detalle el desarrollo de este proceso y toda la experimentación realizada para la comprobación de resultados.

3.7.6 Nuevo método usando otros algoritmos de aprendizaje

En esta nueva etapa se decide experimentar con otros algoritmos de aprendizaje para compáralos con los resultados obtenidos en el proceso anterior.

Para ello se entrenan y evalúan más de 10 clasificadores generados con diferentes algoritmos. También, se hace un análisis del tamaño del corpus de entrenamiento y una validación de los atributos seleccionados. Para estos experimentos se usa la herramienta WEKA, la que implementa una amplia gama de algoritmos de aprendizaje.

Los experimento realizados se centraron en Maquinas de Soporte Vectorial (MSV), Árboles de Decisión (AD) y Redes Neuronales (RN). Por razones obvias no se mostraran los resultados de todos los experimentos, pero sí de los que conllevaron a la elección final de los algoritmos usados.

Luego de un estudio de los atributos utilizados en el caso anterior (ver epígrafe 3.7.5) y pruebas preliminares realizadas con esta nueva modificación, se decide hacer un agrupamiento por categorías. Primeramente, el objetivo de este agrupamiento es reducir su cantidad de atributos, teniendo en cuenta la similitud en la naturaleza de muchos de ellos, se agrupan formando un solo atributo.

Esto se realiza de esta forma, debido a que los algoritmos de aprendizaje utilizando Máxima Entropía trabajan bastante bien al enfrentar la redundancia de información, así como la diversidad en tipo de los atributos utilizados. Los clasificadores de ME son aglomerativos, su decisión es basada en la combinación de todos los atributos activos para un ejemplo específico. Sin embargo, otros tipos de clasificadores son discriminantes, basando su decisión de clasificación solo en las pocas características activas más discriminantes en un ejemplo. Este parece ser el caso de los clasificadores utilizados en esta modificación, pues los resultados obtenidos en las pruebas preliminares, utilizando los 41 atributos del experimento anterior, fueron bajos. En el ANEXO 8 se muestra una descripción en detalles de la transformación que se ha realizado.

En esencia, la transformación fundamental consistió en eliminar los atributos: A04, A07, A10, A11, A12, A19, A20, A32, A34, A35, A38, A41. También, se decide agrupar los atributos A02 y A03, A05 y A06, A08 y A09, A22 y A23, A25 y A26, A28 y A29, A30 y A31, A36 y A37, A39 y A40.

Al final, se obtienen solo 17 atributos para realizar los nuevos experimentos. El proceso de entrenamiento y evaluación se lleva a cabo de la misma forma que se procedió en el experimento anterior. Con la diferencia que en este caso se implementa un prototipo (García y Fernández, 2009) que incorpora una API de ERAFPLN (Pérez y Fernández, 2009) para la generación de las reglas, además se utiliza el recurso Weka para generar los clasificadores entrenados y hacer las inferencias.

Una vez obtenidos todos los elementos, se realizan los experimentos utilizando varios clasificadores. Los detalles se muestran a continuación.

3.7.6.1 Resultados de los entrenamientos y evaluación utilizando diferentes clasificadores

En la Tabla 3.66 y Tabla 3.67 se muestran los resultados de precisión, cobertura y F-medida en las diferentes categorías del experimento con el algoritmo J48 (implementación de Árboles de decisión). Para todos los casos se realizó un entrenamiento con `esp.train` y se evaluó con `esp.testa` y `esp.testb`. En este caso, se han separado los resultados por clases. Esto permite comprobar cómo se comportan estos algoritmos en la clasificación de las diferentes clases.

Tabla 3.66. Experimento con J48 evaluado con `esp.testb`.

Precisión	Cobertura	F-Medida	Clase
0.758	0.744	0.751	Persona
0.726	0.655	0.689	Localidad
0.659	0.825	0.733	Organización
0.434	0.145	0.217	Misceláneas

Tabla 3.67. Experimento con J48 evaluado con `esp.testa`.

Precisión	Cobertura	F-Medida	Clase
0.776	0.696	0.734	Persona
0.622	0.70	0.659	Localidad
0.688	0.789	0.735	Organización
0.452	0.200	0.277	Misceláneas

En la Tabla 3.68 y Tabla 3.69 se muestran los resultados de precisión, cobertura y F-medida para cada clase en el experimento realizado con el algoritmo LibSVM (implementación de Máquinas de Soporte Vectorial). Se realizó un entrenamiento con `esp.train` y se evaluó con `esp.testa` y `esp.testb`.

Tabla 3.68. Experimento con LibSVM evaluado con `esp.testa`.

Precisión	Cobertura	F-Medida	Clase
0.759	0.637	0.693	Persona
0.602	0.741	0.664	Localidad
0.655	0.769	0.707	Organización
0.548	0.142	0.225	Misceláneas

Tabla 3.69. Experimento con LibSVM evaluado con `esp.testb`.

Precisión	Cobertura	F-Medida	Clase
0.696	0.687	0.692	Persona
0.722	0.636	0.676	Localidad
0.621	0.826	0.709	Organización
0.417	0.084	0.139	Misceláneas

Posteriormente, se realizan experimentos combinando varios de los algoritmos probados individualmente. Mediante un voto simple, se logran mejores resultados que los obtenidos con los clasificadores independientes.

En la Tabla 3.70 y Tabla 3.71 se muestran los resultados obtenidos al combinar SMO (Sequencial Minimal Optimization), Multilayer Perceptron y LibSVM en el entrenamiento con `esp.train` y realizando las comprobaciones con `esp.testa` y `esp.testb`.

Tabla 3.70. Experimento con votación (SMO-MLP-LibSVM) evaluado con `esp.testa`.

Precisión	Cobertura	F-Medida	Clase
0.811	0.714	0.759	Persona
0.644	0.706	0.674	Localidad
0.682	0.792	0.733	Organización
0.464	0.229	0.307	Misceláneas

Tabla 3.71. Experimento con votación (SMO-MLP-LibSVM) evaluado con `esp.testb`.

Precisión	Cobertura	F-Medida	Clase
0.767	0.756	0.761	Persona
0.742	0.674	0.706	Localidad
0.684	0.825	0.748	Organización
0.424	0.206	0.278	Misceláneas

Luego de un largo proceso de prueba, en el que se valoran una gran variedad de combinaciones de clasificadores, se decide utilizar la combinación SMO-MLP-J48, por ofrecer cada uno de estos clasificadores los de mejores resultados en la evaluación individual.

Esta combinación siempre se hace mediante un voto simple, se realiza el entrenamiento con `esp.train` y se evalúa con `esp.testa` y `esp.testb`. En las Tabla 3.72 y Tabla 3.73 se muestran los resultados obtenidos al `evaluar` la combinación descrita con `esp.testa` y `esp.testb`.

Tabla 3.72. Experimento con votación entre (SMO-MLP-J48) evaluado con `esp.testa`.

Precisión	Cobertura	F-Medida	Clase
0.837	0.725	0.777	Persona
0.661	0.732	0.694	Localidad
0.697	0.839	0.762	Organización
0.554	0.196	0.289	Misceláneas

Tabla 3.73. Experimento con votación entre (SMO-MLP-J48) evaluado con `esp.testb`.

Precisión	Cobertura	F-Medida	Clase
0.793	0.760	0.776	Persona
0.735	0.678	0.705	Localidad
0.669	0.841	0.745	Organización
0.470	0.139	0.214	Misceláneas

Para una mejor comparación, se muestran en las (Tabla 3.74, Tabla 3.75, Tabla 3.76 y Tabla 3.77) los mejores resultados por cada combinación, para la clasificación de cada una de las clases.

Tabla 3.74. Mejores resultados evaluando la clase Persona.

Algoritmo	Precisión	Cobertura	F-Medida	Clase
(SMO-MLP-J48) evaluado con esp.testa	0.837	0.725	0.777	Persona
(SMO-MLP-J48) evaluado con esp.testb	0.793	0.760	0.776	Persona

Tabla 3.75. Mejores resultados evaluando la clase Localidad.

Algoritmo	Precisión	Cobertura	F-Medida	Clase
(SMO-MLP-LibSVM) evaluado con esp.testb	0.742	0.674	0.706	Localidad
(SMO-MLP-J48) evaluado con esp.testa	0.661	0.732	0.694	Localidad

Tabla 3.76. Mejores resultados evaluando la clase Organización.

Algoritmo	Precisión	Cobertura	F-Medida	Clase
(SMO-MLP-J48) evaluado con esp.testa	0.697	0.839	0.762	Organización
(SMO-MLP-LibSVM) evaluado con esp.testb	0.684	0.825	0.748	Organización

Tabla 3.77. Mejores resultados evaluando la clase Misceláneas.

Algoritmo	Precisión	Cobertura	F-Medida	Clase
(SMO-MLP-J48) evaluado con esp.testa	0.554	0.196	0.289	Misceláneas
(SMO-MLP-J48) evaluado con esp.testb	0.470	0.139	0.214	Misceláneas

Haciendo un análisis del comportamiento de estos clasificadores, (como se muestra en las Tabla 3.74, Tabla 3.75, Tabla 3.76 y la Tabla 3.77) se puede ver que la combinación (SMO-MLP-J48), es la que mejor funciona en la mayoría de los casos, con la excepción de la clasificación de la clase `Localidad` en el corpus `esp.testb`. En este caso, la combinación (SMO-MLP-LibSVM) obtiene resultados superiores (ver Tabla 3.75).

Es evidente que los resultados obtenidos en todos estos experimentos son inferiores a los obtenidos con el clasificador de Máxima Entropía. Esto puede ser explicado desde varios puntos de vista. Primero, el hecho de reorganizar los atributos ha afectado en gran medida la clasificación. Lo que ha sucedido es que al reducir la cantidad de atributos, ahora existen varias instancias que se repiten, pero con clasificaciones distintas. Los algoritmos utilizados en estos experimentos no

También es de destacar, que ninguno de los clasificadores tiene un buen comportamiento al evaluar la clase Miscelánea. La baja precisión y cobertura en la clasificación de esta clase se debe a que engloba entidades que son muy diversas como: título de obras de teatro, nombres de películas, leyes, etc. Un elemento importante, que evitó los bajos resultados en la clasificación de esta clase, en la experimentación con ME, es el hecho de que en esa ocasión se asumió una regla en la que las

entidades que no podían ser clasificadas como Persona, Localidad u Organización fueron automáticamente designadas como Miscelánea. Estos bajos resultados también han sido reportados por otros investigadores (Gutiérrez y Martín; Troyano Jiménez, Díaz Madrigal et al., 2003; Toribio, Martínez et al., 2010).

En otro orden de cosas, es importante discutir, debido a su gran utilización en el PLN, lo referente a la generación de patrones de emparejamiento y reglas a través de las Expresiones Regulares. En el siguiente epígrafe se describe en detalle el método y los recursos utilizados en este sentido para la capturar de entidades y otros trabajos.

3.8 La generación de Expresiones Regulares y el PLN

Existe una gran cantidad de aplicaciones de PLN que hacen uso de las Expresiones Regulares para diversas tareas como: el análisis léxico, sintáctico o semántico, tokenizado, etiquetado, separación en frases, reconocimiento de patrones, creación y extracción de reglas, entre otras. En todas estas tareas son aplicadas estas expresiones con el objetivo de agilizar estos procesos.

Encontrar un patrón en un texto o generar una regla para utilizarla en un determinado sistema, no es una operación sencilla, de hecho es un proceso bastante tedioso. En estos casos, se hace necesario -al escribir la ER- especificar todos los meta-caracteres que la constituyen y esto se realiza a través de un lenguaje regular.

Además de esto, es fácil darse cuenta que el léxico de estos lenguajes es muy diferente al lenguaje humano, lo que lo hace bastante difícil para aquellos que no están familiarizados con esta temática.

En ocasiones, se hace necesario escribir patrones de emparejamiento para la validación de estructuras que están compuestas por algún elemento que es variable. Pero esa variable puede contener diferentes instancias. Esto hace que la expresión sea muy larga y por lo tanto difícil de crear y mantener.

Muestra de esto, podría ser el ejemplo que se muestra en la Tabla 3.78, en el que se intenta reconocer varios formatos de fechas. Como se puede apreciar, es necesario escribir directo en la ER todos los meses del año cuando se está construyendo la expresión. Aunque en este caso no son muchos, existen casos en los que la cantidad de elementos a escribir podría ser mucho mayor. Una variante interesante sería poder importar esto datos de un lexicón, incluso con su traducción a varios idiomas (de hecho existen lexicones así).

Tabla 3.78 Fecha con formato día/mes/año (donde el mes puede ser expresado con su nombre o con número).

```
((\d\d/)|(\d/))((((\d\d/\d)|(\d/\d))\d\d\d|((\d\d/\d)|(\d/\d))\d)|
(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octu
bre|noviembre|diciembre)((/\d\d\d\d)|(/\d\d)))
```

Para solucionar varios de esto problemas se parte de la idea, ya demostrada por otros investigadores (Brena, 2003; Formella, 2010), de que es posible obtener una Expresiones Regulares a partir del Autómata Finito que la representa.

Esta posibilidad, permitiría crear un recurso capaz de facilitar la forma de crear las ER, o por lo menos abstraerse de la complejidad de los lenguajes regulares.

La idea de este investigador, parte de que la solución es un método capaz de representar las ER a partir del autómata finito, pero es necesario comprobar se crear el autómata sería más sencillo que crear la expresión. La idea, parte entonces de la hipótesis de que es más sencillo construir una representación gráfica de un problema para darle solución, que representarlo a través de un lenguaje regular, en el que existe toda una serie de símbolo difícil de recordar e interpretar.

Este método permite que una persona con pocos o ningún conocimiento sobre ER, las pueda escribir a su conveniencia.

Es necesario entonces un algoritmo que sea capaz, primero, de construir un autómata partiendo de una representación gráfica utilizando un conjunto de símbolos. Luego es preciso generar la ER correspondiente a ese Autómata Finito. En el siguiente epígrafe se explica cómo se logra este proceso.

3.8.1 Representación del un Autómata Finito

La representación visual de un Autómata Finito, se ha definido tomando como punto de partida el diagrama de transición de estados, donde cada estado está representado por un círculo con color de fondo amarillo y borde negro, con la etiqueta en su centro (ver Figura 3.22). Se crean una serie de representaciones para cada uno de los elementos que componen un autómata. El estado inicial se distingue por una saeta que apunta desde la izquierda como aparece en la Figura 3.22, los estados finales se distinguen por un círculo más grueso en el borde (ver Figura 3.22).

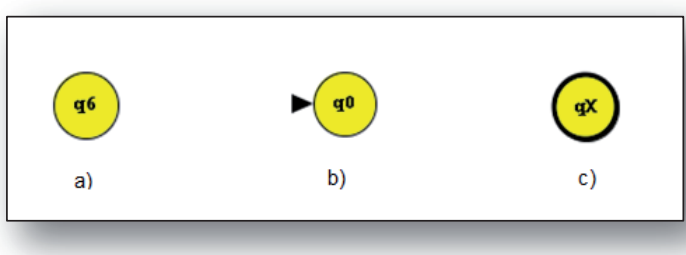


Figura 3.22. Representación de estados, a) estado ordinario, b) estado inicial, c) estado final.

Otro elemento de gran importancia a la hora de construir un autómata son los enlaces entre los estados. Con este objetivo, se han diseñado una gran variedad de enlaces predeterminados como son los alfanuméricos, dígitos, separadores, letras mayúsculas, minúsculas y variable.

- **Enlace dígitos:** construye un enlace con el símbolo `\d`.
- **Enlace separador:** construye un enlace con el símbolo `\s`.
- **Enlace letras mayúsculas:** construye un enlace con el símbolo `[A-Z]`.
- **Enlace letras minúsculas:** construye un enlace con el símbolo `[a-z]`.
- **Enlace variable:** construye un enlace que asimila una variable. Se denomina `variable` a un fragmento de la ER que es una lista de elemento del mismo tipo, por ejemplo: días de la semana, meses de año, etc. Estos datos pueden estar almacenados en cualquier soporte externo, un fichero en formato ASCII[63] con el conjunto de datos. Este fichero debe contener una palabra en cada línea.

Los enlaces o transiciones son representados con arcos cuadráticos, como se muestra en la Figura 3.23-a. Van desde el estado de origen hasta el destino. Lo que se busca con esta representación es el mayor parecido con el diagrama de transición de estados. Hay un tipo de enlace particularmente diferente, los auto-enlaces[64], que se representan por curvas (también de tipo cuadrática) de tamaño fijo donde ambos extremos están sobre el mismo estado, esto se puede apreciar en la Figura 3.23-b.

Los enlaces tendrán una etiqueta que servirá para nombrarlos, pero internamente en el algoritmo llevan el símbolo correspondiente del lenguaje regular, que al final representa la ER.

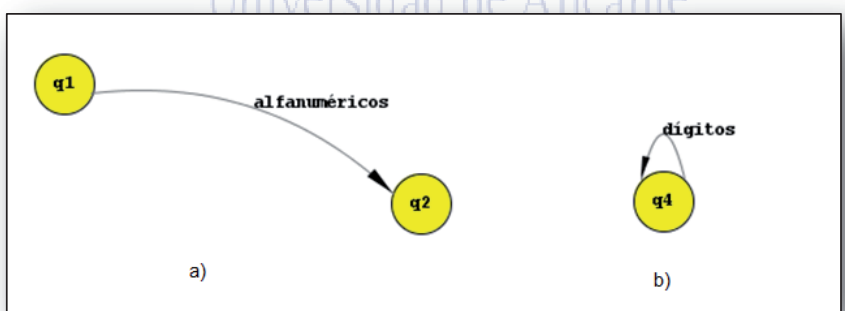


Figura 3.23. Representación de los enlaces, a) enlace, b) auto-enlace.

El objetivo final de este algoritmo es la representación de un AF mediante símbolos gráficos, que luego puedan transformarse en una representación computacional que dé lugar a su ER equivalente.

[63] American Standard Code for Information Interchange.

[64] Transiciones que dan como resultado el mismo estado de que parten

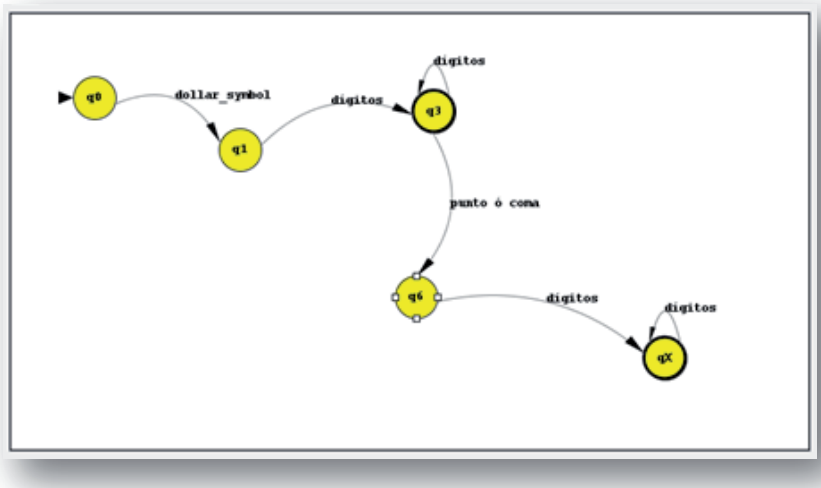


Figura 3.24. Autómata finito dibujado con los símbolos creados.

Retomando los elementos teóricos acerca de los AF, abordados en el epígrafe 2.7.2, se tiene que un AF es un quíntuplo compuesto por un conjunto de estados, el alfabeto de entrada, un estado inicial, un conjunto de estados finales y una función de transición; que determina a partir de un estado y un símbolo del alfabeto, cuál es el nuevo estado en que se encontrará el autómata. Y si es uno de los estados finales, entonces la cadena a evaluar es representada por dicho AF.

Cuando se dibuja un autómata en un ordenador, es necesario mantener en memoria una instancia de una estructura de datos que lo representa. Esta estructura, lleva el control de los estados y enlaces existentes y su interacción. Además, es la encargada de almacenar una lista de estados y de enlaces. Cada elemento de la lista de estados es una estructura de datos que representa un estado y es encargada de identificar si es inicial (condición asignada en su creación y no es modificable) o si es final, este valor es modificable debido a que pueden existir varios estados finales y puede cambiarse su tipo en el desarrollo del autómata.

De forma independiente, cada elemento de la lista de enlaces es una estructura capaz de almacenar las responsabilidades mínimas de los elementos de enlace en el autómata, como -por ejemplo- conocer qué estados conforman dicho enlace. De esta estructura heredan otras dos, la que almacena los enlaces ordinarios y los auto-enlace.

La función de transición no está expresada de forma explícita en la codificación del algoritmo, pero de la forma en que están almacenados los estados y los enlaces, se obtiene de manera muy sencilla a partir de la tabla de transiciones.

A su vez, La tabla de transiciones puede ofrecerse para estudios más profundos sobre esta temática. Para esta investigación, se ha decidido utilizar una forma diferente de mostrar la tabla de transiciones, en lugar de mostrar los símbolos del alfabeto de entrada por columnas (pueden llegar a ser demasiados, habrían muchas columnas y sería difícil su lectura), se muestran en forma de matriz de adyacencia, como en un grafo. De esta forma, por filas y columnas se muestran los estados, en la intersección fila-columna se almacena qué símbolo (ó símbolos) hay que consumir de la entrada para llegar desde el estado de la fila hasta el estado de la columna.

Ahora solo resta utilizar un algoritmo que sea capaz de hacer la transformación del AF en la ER deseada. A continuación se detalla cómo se ha realizado esta tarea.

3.8.2 Algoritmo de conversión de autómata finito a expresión regular.

El algoritmo que se describe a continuación es resultado de una compilación entre el descrito por Brena publicado en (Brena, 2003) y la traducción al español que aparece en (Hopcroft y Ullman, 2001). Se ha modificado la terminología unificando el término de estado, pues en algunos casos se refieren a éste como nodo.

Tomando esta idea, para transformar un AF en una expresión regular equivalente, un procedimiento posible para hacerlo consiste en ir eliminando gradualmente los estados del autómata, hasta que únicamente queden un estado inicial y un estado final. Este proceso comprende los siguientes pasos:

El primero paso en este procedimiento es añadir dos nuevos estados al AF, un nuevo estado inicial `i`, que hace que el antiguo estado inicial `q0` deje de ser inicial y un nuevo estado final `f`, de modo que los antiguos estados finales $q_i \in F$, dejan de ser finales; además se añade una transición vacía del nuevo estado inicial al antiguo, (i, ε, q_0), y varias transiciones de los antiguos estados finales al nuevo: $[[(q_i, \varepsilon, f)|q_i \in F]]$. Esta transformación tiene por objeto que haya un estado inicial al que no llegue ninguna transición, y un solo estado final, del que no salga ninguna transición. Esta condición es requerida para llevar a cabo el siguiente paso.

El segundo paso consiste en eliminar estados intermedios, se llama estado intermedio a aquel que se encuentra en una trayectoria entre el estado inicial y el final. El procedimiento de eliminación de estados intermedios es directo. El objetivo es que al suprimir el estado en cuestión, no se alteren las cadenas que hay que consumir para pasar de uno a otro de los estados vecinos, es decir, al suprimir dicho estado, se deben reemplazar las transiciones que antes tomaban ese estado como punto intermedio para ir de un estado vecino a otro, por otras transiciones que vayan del estado vecino original estado vecino destino, pero ahora sin pasar por el estado eliminado. En la Figura 3.25 se

observa como para pasar de p_1 a q_m y de p_n a q_1 se toma como estado intermedio a q. Así, al eliminar el estado q se verán afectados los caminos p_1-q-q_1, p_1-q-q_m, p_n-q-q_1, p_n-1-q_m. En la Figura 3.26 se muestran los caminos, ya sin el estado q donde por ejemplo: para pasar de p_1 a q_1 se conserva α_1 $(\beta_1+\ldots+\beta_k)^*\gamma_1$

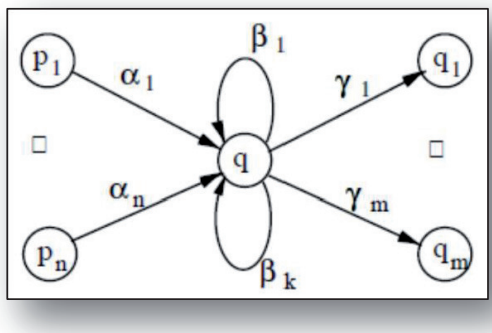


Figura 3.25. Fragmento de autómata antes de eliminar el estado q.

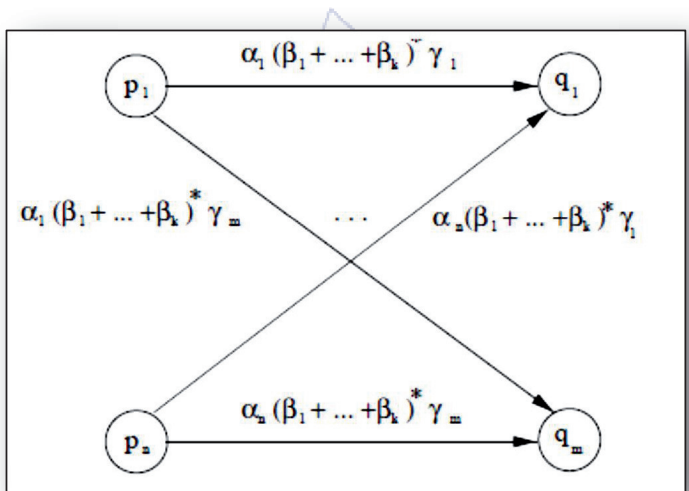


Figura 3.26. Fragmento de autómata finito después de haber eliminado el estado q.

Modificación de la conversión de autómata a ER

En el epígrafe 3.8.2 se explica cómo se convierte del autómata finito a la expresión regular, la expresión resultante contiene los símbolos del alfabeto (alfabeto del español), los operadores

asterisco (*) y barra vertical (|) y los paréntesis para agrupar subexpresiones. El operador de concatenación está implícito cuando los elementos léxicos aparecen uno a continuación del otro.

En la práctica, los motores de ER aceptan muchos más operadores que pueden variar en dependencia de la librería que se esté usando. En este caso, se ha considerado de gran importancia que la ER como resultado de aplicar la conversión desde el autómata, tenga además de los citados operadores, el operador + (ver ejemplo de la Figura 3.28. El significado de dicho operador es invariante entre las diferentes implementaciones y es el cuantificador de repetición de una o más veces, quiere decir que el fragmento de expresión regular afectado por el signo +, debe aparecer al menos una vez en el texto. Esto contribuiría a disminuir la longitud de la cadena que da como resultado, siendo más compacta. Con el operador + se cumple que: (X)+ = (X)(X)* = (X)*(X).

La Figura 3.27 muestra el algoritmo de conversión de autómata finito a expresión regular por el método de eliminación de estados, donde se ha desglosado en partes el paso de eliminar cada estado. Entre los pasos necesarios para eliminar un estado, se han destacado en rojo aquellos que se han modificado para obtener el operador +. Cuando se tiene un caso como el de la Figura 3.28, en el que hay un enlace y un auto-enlace consecutivos, se puede ver que si se eliminar el estado q_1, los enlaces anotados como dígitos cumplen esta condición. Teniendo en cuenta esto, en el mencionado paso, se ha programado de la siguiente manera:

Para cada enlace e_1 que llega y e_2 que parte del estado a eliminar e, hacer:

- Si existe un auto-enlace sobre e
- y
- El final de la etiqueta de e_1 coincide con la etiqueta del auto-enlace, entonces hacer:
  - Temporal e_3 = la parte que no coincide de e_1 concatenado con la etiqueta del auto-enlace y el signo +.
  - Nuevo enlace e_4 = e_3 concatenado con e_2

De esta forma en el ejemplo de la Figura 3.28 se obtiene como resultado `(dígitos)+(letras minúsculas)`, considerando que los motores de expresiones regulares abrevian dígitos como \d y las letras minúsculas comprenden el intervalo `[a-z]`, la expresión regular que representa este autómata quedaría `(\d)+[a-z]`.

Por otra parte, como los motores de ER utilizan para evaluar internamente un AFN, cuando aparece una unión (`|`) el evaluador trata de tomar la vía más corta para llegar a un estado final. Como resultado, si una de las subexpresiones que separan la barra es subcadena de la otra, entonces la subexpresión más larga debe quedar delante, es decir, a la izquierda del operador de unión.

De no ser así, el evaluador no detectará las variantes más cortas, siendo esto un gran problema, pues no se cumpliría la completez de la solución. Por este motivo se ha modificado el proceso de generación teniendo en cuenta la longitud de ambos operandos, para así colocar siempre el más largo a la izquierda.

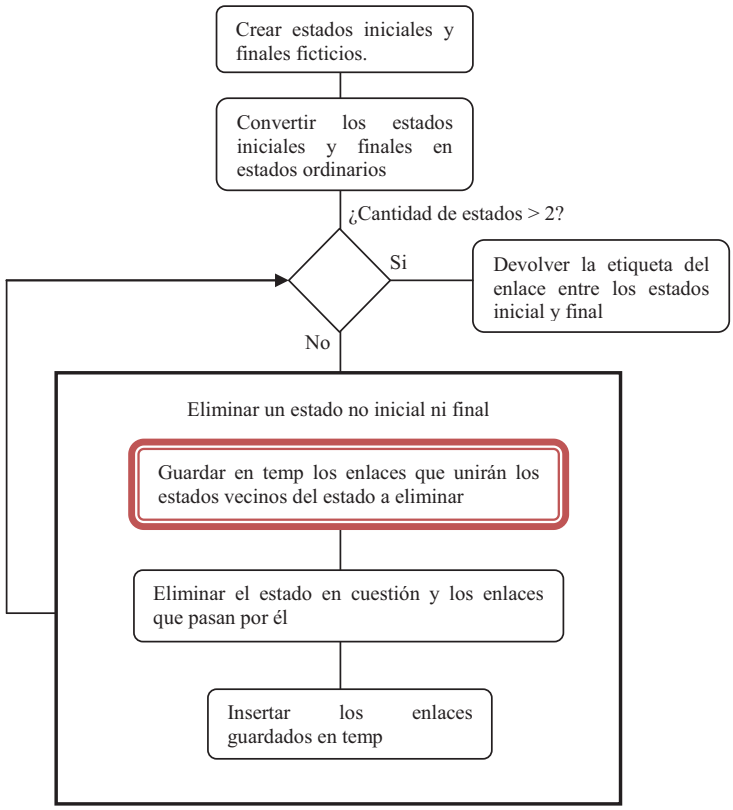


Figura 3.27. Diagrama de bloques del algoritmo de conversión de AF a ER

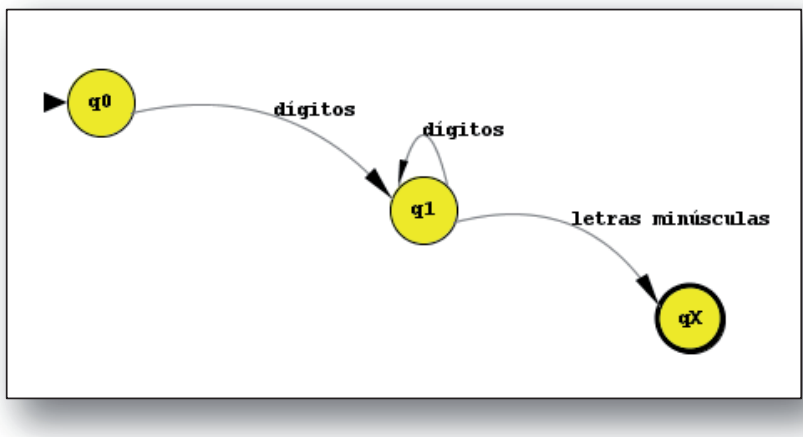


Figura 3.28 Autómata de ejemplo para obtener un operador +.

Especialización de las funcionalidades para uso de PLN

Como se ha mencionado en el epígrafe 3.8.1, el algoritmo desarrollado implementa un tipo de enlace especial que permite obtener valores de un diccionario, los que serán usados en la posición de una variable. Con esto se evita tener que introducir, una de cada vez, las opciones en caso de alternativas múltiples, como por ejemplo: artículos, días de la semana, meses del año, entre otras.

En trabajos de PLN, el uso de estos diccionarios es bastante frecuente, por lo que esta opción contribuye a esta rama de la investigación, aportando una manera cómoda de construir expresiones regulares donde existen múltiples alternativas (ver ejemplo de ER en la Tabla 3.78).

Otro elemento, que has sido de amplio uso en las investigaciones en PLN es el etiquetado de determinados patrones del texto que se desea analizar posteriormente o simplemente destacar. Muchas aplicaciones de PLN realizan esta tarea, pero desafortunadamente, tanto los elementos que etiquetan como las etiquetas propiamente, son fijos y no se pueden alterar; a menos que se pueda entrar en el código fuente de estas aplicaciones. Basado en este elemento, se ha diseñado un algoritmo que soluciona este inconveniente.

Se implementan dos opciones, la primera es el remplazo, que es fundamental para el trabajo con expresiones regulares. La segunda como un caso particular del remplazo, es el etiquetado de textos. Para ello, solo son necesarios el texto de antes y el de después de las coincidencias (texto buscado) a etiquetar.

Se considera etiquetado, al proceso de insertar etiquetas (cadenas de texto que muchos casos toman la estructura de los tags de XML (Extensible Markup Language), ejemplo:

<etiqueta_antes>texto coincidente</etiqueta_después>) antes y después de cada coincidencia de un patrón dado. Esto puede resolverse con un simple replace que aporta cualquier motor de ER. Pero esto trae como consecuencia que, si se aplica más de una vez sobre el mismo texto, se multiplican las etiquetas.

Para resolver esta problemática se ha implementado en el algoritmo el método siguiente:

1. Aplicar la expresión regular actual al texto.
2. Inicializar resultado como ε (cadena vacía) y `ultima_cuenta` en `0`.
3. Para cada una de las coincidencias m, hacer:
   - `Temporal_anterior = cadena_comprendida_entre ultima_cuenta y posición_comienzo_patrón`
   - `Temporal_despues = cadena_comprendida_entre posición_fin_patrón y fin_del_texto`
   - Considerar ya etiquetado si:
     - `Temporal_anterior` termina con `etiqueta_antes`
     - y
     - `Temporal_despues` termina con `etiqueta_después`
   - Si no está etiquetado, entonces:
     - Añadir al resultado `etiqueta_antes + patron_encontrado + etiqueta_después`
   - Si ya lo está, entonces:
     - Añadir al resultado solo `patron_encontrado`
   - Hacer `ultima_cuenta = posición_fin_patrón`
4. Remplazar el texto anterior por lo que quedó acumulado en resultado.

3.8.3 Experimentación

El objetivo primordial del método creado es ofrecer una vía `más simple` para la creación de reglas en los sistemas de PLN. Para validar está hipótesis se hicieron varias pruebas al método. Todas en su mayoría encaminadas a verificar la factibilidad de utilizar el AF para la creación de ER, que serán el lenguaje que se utilizará en la creación de las reglas.

Para poder experimentar con el método creado, fue necesario implementar un prototipo (Pérez y Fernández, 2009) que permitiera realizar todas las operaciones. Con este recurso creado, se somete a prueba con dos grupos de 30 estudiantes cada uno, procedentes del tercer año de la Facultad de Informática de la Universidad de Matanzas. Durante una actividad de laboratorio se le aplican tres ejercicios a cada estudiante.

La resolución de los ejercicios consiste en el planteamiento de un problema donde se necesita encontrar el AFD y la ER que represente el lenguaje regular descrito en el texto del problema. Con la ayuda del prototipo se puede comprobar, en tiempo real, si la respuesta es satisfactoria.

3.8.3.1 Experimento con dos grupos de 30 estudiantes

1. Se pide desarrollen un autómata finito capaz de capturar una dirección de correo electrónico (sencilla, sin tener en cuenta la validez de los nombres de dominio ni la cantidad de estos). Posteriormente se genera la expresión correspondiente y se compara con la solución.

Solución: `\w+@\w+(\.\w+)+`

2. Se pide desarrollen un autómata finito capaz de capturar una cantidad numérica que representa precios de artículos (con el signo $ incluido y la coma como separador de cifras decimales). Posteriormente se genera la expresión correspondiente y se compara con la solución.

Solución: `((\$\d(\d)*[.,]\d(\d)*)|(\$\d(\d)*))`

3. Se pide desarrollen un autómata finito capaz de capturar una fecha con el formato día/mes/año (donde el mes puede ser expresado con su nombre o con número). Posteriormente se genera la expresión correspondiente y se compara con la solución.

Solución:

```
((\d\d/)|(\d/))((((\d\d/\d)|(\d/\d))\d\d\d|((\d\d/\d)|(\d/\d))\d)|
(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octu
bre|noviembre|diciembre)((/\d\d\d\d)|(/\d\d)))
```


A través de la observación por dos investigadores, se midieron tres parámetros: desenvolvimiento con la interfaz gráfica, habilidad de creación utilizando el prototipo para la confección del autómata finito, así como pruebas y puesta a punto de expresiones regulares. No se enfocó en la creación de las expresiones regulares, porque el prototipo es capaz de obtenerlas a partir del autómata.

3.8.3.2 Resultados y análisis del experimento con 30 estudiantes.

Los resultados arrojados fueron los siguientes:

1. En el desenvolvimiento con la interfaz gráfica en general.

   a) El 92% logró conducirse con gran facilidad.

   b) El 5% logró conducirse realizando algunas preguntas.

   c) El restante 3% tuvo dificultades para realizar los ejercicios.

En la habilidad de creación utilizando el prototipo para la confección del autómata finito, a continuación en la Tabla 3.79, resumen los resultados de este parámetro.

Tabla 3.79 Resultados del uso del prototipo en los tres ejercicios

Ejercicios	Trabajaron sin problemas	Tuvieron algunos problemas	No pudieron resolver el ejercicio
1	83%	11%	6%
2	79%	7%	4%
3	72%	8%	9%

2. En las pruebas y puesta a punto de expresiones regulares.

   a) El 93% logró desenvolverse con gran facilidad.

   b) El 3% logró desenvolverse realizando algunas preguntas.

   c) El restante 4% tuvo dificultades para realizar los ejercicios.

De las cifras anteriores y en particular de las reflejadas en la Tabla 3.79, donde se muestran los resultados más importantes obtenidos del experimento anterior, se puede considerar que aunque la mayoría de los estudiantes pudieron utilizar a plenitud el prototipo, hubo algunos estudiantes que tuvieron problemas.

Se observaron las dificultades que presentaron a la hora de manipular el editor gráfico para la creación de los autómatas, de estas surgen modificaciones en la metodología que contribuyen a mejorar la interacción con este recurso. También se detectaron algunos bugs[65], aunque no de la metodología sí de la programación, en la generación de la expresión regular, los que fueron debidamente reparados.

La mayoría de los problemas que se presentan están asociados a la interfaz del prototipo, no a la metodología creada.

En cuanto a los bugs (errores) encontrados, fue precisamente mediante esta prueba que se detecta que cuando hay un operador de unión (|) y un operando es subcadena del otro, entonces hay que colocar el de mayor longitud a la izquierda, como se explica en el epígrafe 2.7.

3.8.3.3 Experimento con cinco estudiantes

Intentando demostrar la hipótesis de que la representación del autómata finito permitirá la creación de expresiones regulares, sin necesidad de poseer amplios conocimientos sobre su estructura y

[65] Errores de programación

sintaxis; se realiza un experimento encaminando a corroborar está afirmación. Para comprobarla se seleccionan al azar cinco estudiantes de segundo año de la carrera de Ingeniería Informática de la Universidad de Matanzas. Dichos estudiantes no tienen ningún conocimiento, ni de autómatas, ni de ER.

El experimento puede ser clasificado por el tiempo de duración como breve, pues duró menos de dos horas. En función de las condiciones para la realización se clasifica como de laboratorio, debido a que los participantes fueron escogidos teniendo en cuenta si conocían o no, acerca de los autómatas y expresiones regulares. Se escogen, precisamente de ese curso, porque no han recibido la asignatura Programación IV, donde se abordan los contenidos relacionados con las expresiones regulares y autómatas finitos.

Antes de comenzar se les da una pequeña panorámica de los elementos básicos de expresiones regulares y de autómatas finitos.

En el caso de autómatas finitos consistió en describir qué es, cómo funciona y un ejemplo paso a paso. En el caso de las expresiones regulares se les explican los elementos que las conforman con un ejemplo para cada parte, dejando escrito en la pizarra los caracteres reservados y un resumen de lo planteado.

Esta información proporcionada a los estudiantes es lo que se entiende como conocimientos mínimos en el marco de esta investigación, es decir, que el único conocimiento que tienen acerca del tema es lo que se les enseñó brevemente, minutos antes.

Para obtener el tamaño adecuado de la muestra según la población (60 estudiantes entre los dos grupos de segundo año de la carrera de informática) se ha utilizado la siguiente fórmula según (Pita Fernández y Pértega Díaz, 2001)

$$n = \frac{N \cdot Z_\alpha^2 \cdot p \cdot q}{d^2 \cdot (N-1) + Z_\alpha^2 \cdot p \cdot q} \qquad 3.36$$

Z_α = 1.962 (nivel de confianza 95%)

p = 0.5 (proporción esperada)

q = 1 – p (en este caso 1 – 0.5 = 0.5)

d = 0.05 (precisión, en este caso se espera un 0.5% de error)

N= 60 estudiantes

La muestra que se debe tomar es de 52 estudiantes. La muestra utilizada no es representativa, pues se toman solo diez estudiantes. No obstante se disidió que si al menos un estudiante podía resolver los ejercicios más rápido, con una diferencia de al menos un minuto, por la vía del autómata que por la expresión regular; con los escasos conocimientos sobre ambos métodos, ya se estaría obteniendo un resultado relevante, que corroboraría la validez de la hipótesis planteada.

Se seleccionan los estudiantes entre los que tuvieran un promedio académico entre 3.7 y 4. Se les pide resolver tres ejercicios que consisten en detectar patrones de texto, mediante el autómata finito y la expresión regular, directamente utilizando el prototipo para comprobar sus resultados en ambos casos.

Las tareas son asignadas en orden de menor a mayor complejidad, para que ganen en práctica y así enfrentar problemas más complejos. Los ejercicios propuestos son los siguientes:

1. Obtener el autómata finito y la expresión regular para detectar cifras numéricas correspondientes a cantidad de dinero, ejemplos de cadenas válidas: $1.00, $0.50, destacando que el símbolo de pesos es obligatorio al inicio de la cadena.

2. Obtener el autómata finito y la expresión regular para detectar la hora en el formato Horas : Minutos `am` ó `pm`, sin tener en cuenta que sea una hora válida, es decir, las cadenas `25:99am` y `19:85pm` serían aceptadas como válidas, como también lo son `1:1am`, `01:01am`, `11:58PM`. Lo importante en este ejercicio es que sean capaces de detectar dígitos separados por dos puntos y al final las constantes `am` y `pm` en sus variantes mayúsculas y minúsculas.

3. Obtener el autómata finito y la expresión regular para detectar la hora en formato militar, es decir, horas de `0` a `23`, dos puntos y minutos de `0` a `59`. Esta vez sí es necesario que validen que la hora sea correcta. En este caso cadenas como: `25:88` no son consideradas como válidas, sólo desde las `00:00` hasta las `23:59`.

En cada caso se miden los tiempos que tardan en resolver cada uno de los ejercicios propuestos, si la solución es incorrecta se les notifica y se sigue intentando hasta dar con la respuesta correcta.

3.8.3.4 Resultados y análisis del experimento con cinco estudiantes.

En la Tabla 3.80 se muestran los valores de los tiempos de cada estudiante, en cada ejercicio, así como el promedio por ejercicio. Solo se refleja el tiempo en minuto, los segundos no se tuvieron en cuenta, se redondean los valores.

Tabla 3.80 Tiempos en minutos de duración en resolver los ejercicios propuestos.

Estudiantes	AF 1	ER 1	ER 2	AF 2	AF 3	ER 4
1	3	4	13	9	13	11
2	4	5	10	9	9	15
3	4	6	11	8	7	12
4	4	6	4	5	8	10
5	4	4	6	6	9	11
6	3	5	10	8	10	13
7	3	5	10	8	8	11
8	5	6	11	6	11	15
9	4	6	9	6	7	13
10	4	6	9	5	9	13
Tiempo promedio	3.8	5.3	9.3	7	9.1	12.4

Leyenda: AF – Autómata Finito, ER – Expresión Regular

Como se pueden apreciar las diferencias de tiempo promedio entre la resolución de cada ejercicio mediante el autómata (columnas AF), respecto a la resolución utilizando directamente las expresiones regulares (columnas ER), en todos los casos fueron significativas (ver Tabla 3.80, tiempos promedio). Demostrándose que se utiliza menos tiempo en la generación del autómata finito que en la generación de la expresión regular.

Al aumentar la complejidad del ejercicio, aumenta también la diferencia en tiempo entre ambos métodos. Además se alteró el orden entre ambas formas de resolución para evitar favorecer alguna en específico.

Los errores más frecuentes se registran en el caso de las expresiones regulares, se pueden citar algunos que aparecieron de forma reiterada, por ejemplo:

a) Omitir el carácter de escape (\) delante del punto para interpretarlo como punto y no como cualquier carácter, que es su significado en el lenguaje regular.

b) No agrupar correctamente los fragmentos de expresión regular para aplicar cuantificadores como: (+) y (?).

c) No considerar exactamente la cantidad de repeticiones para un carácter determinado.

El mayor tiempo se consume en diseñar y poner a punto la expresión regular, se comprobó que para comenzar a generar la expresión regular se demoraban más tiempo que para comenzar a crear el autómata. También sucede lo mismo con la puesta a punto de ambos elementos.

A continuación se muestran las soluciones a los ejercicios propuestos, cada uno con su autómata finito y la expresión regular correspondiente. Es importante destacar que las soluciones que se proponen no son las únicas, cada problema tiene múltiples soluciones, y sólo quien posea amplios conocimientos podrá dar las soluciones óptimas.

Una expresión regular que da solución al ejercicio 1 es:

```
((\$(\d)+[.,](\d)+)|(\$(\d)+))
```

El autómata para esta expresión sería:

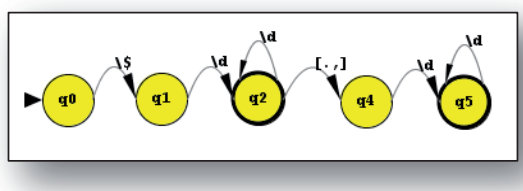


Figura 3.29 Autómata finito que da solución al ejercicio 1.

Una expresión regular que da solución al ejercicio 2 es:

```
(((\d\d)|(\d)):\d\d|((\d\d)|(\d)):\d)(AM|am|PM|pm|m|M)
```


El autómata sería:

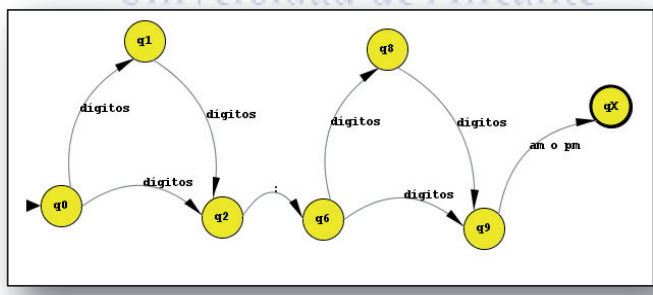


Figura 3.30 Autómata finito que da solución al ejercicio 2.

Una expresión regular que da solución al ejercicio 3 es:

```
((((1\d)|(\d))|(0\d)):|(2(0|1|2|3):))(0|1|2|3|4|5)\d|((((1\d)|(\d
))|(0\d)):|(2(0|1|2|3):))\d)
```

El autómata en este caso sería:

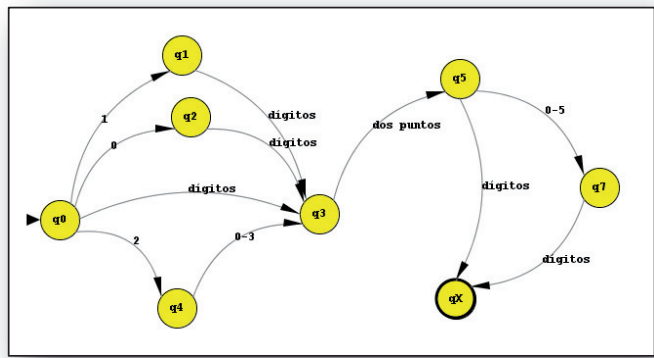


Figura 3.31 Autómata finito que da solución al ejercicio 2

Una vez concluida la explicación de las ocho tareas desarrolladas en este capítulo, solo resta hacer las conclusiones a las que se ha llegado en cada uno de estos temas.

3.9 Conclusiones parciales

Las conclusiones de este capítulo se hacen partir de cada una de las tareas investigadas.

Extensión de la Distancia de edición

La conclusión fundamental, extraída de la experimentación con la Extensión de la Distancia de edición, es que la incorporación de la penalización por la realización de permutaciones en la raíz de la palabra, así como la colocación de los pesos a los caracteres, influye positivamente en la determinación de la distancia entre dos palabras. Además, se ha podido comprobar que la DEx tiene un comportamiento superior, en la medición de distancias entre palabras, así como en el agrupamiento en familias, con respecto a otras métricas con las que ha sido comparada.

Además, al utilizar esta distancia como un atributo más en el Alineamiento Léxico-Semántico, se ha comprobado que tiene una marcada influencia en este proceso. Como se puede ver en el ANEXO 10, esta métrica obtiene un coeficiente de correlación de 0.596, por lo que resulta ser de gran importancia para este método de estimación de similitud. Estos resultados la convierten en un recurso importante para muchas tareas del PLN.

Polaridad Sentimental en el Reconocimiento de la Implicación Textual

Por otra parte, a través del experimento de comprobación sobre la influencia de la Polaridad Sentimental en el Reconocimiento de la Implicación Textual, se ha podido comprobar la factibilidad de la premisa de la que se partió. Si un par texto-hipótesis posee relaciones de polaridad diferente, específicamente opuesta (negativo-positivo), no debe existir una relación de implicación textual entre ellos. También se ha podido corroborar que se obtienen buenos resultados en la identificación de la implicación, partiendo de la premisa de que ambas frase deben poseer polaridades iguales.

Con relación a este mismo tema, es bueno aclarar que la mayoría de los recursos semánticos utilizados obtienen una relación muy parecida entre Polaridad Sentimental e Implicación Textual. Sin embargo, combinando los resultados de cada recurso mediante una votación se logran resultados superiores. Lo que permite inferir que una óptica multidimensional favorece los resultados en esta problemática.

Contradictoriamente, al utilizarla como un atributo en el Alineamiento Léxico-Semántico, no ofrece estos resultados que se vieron en los experimentos descritos en el epígrafe 3.2.1.2. Este es un aspecto que debe ser estudiado con mayor detalle, pues contradice totalmente lo alcanzado anteriormente.

Alineamiento Léxico-Semántico

Analizando otro elemento de los investigados, específicamente las pruebas realizadas al Alineamiento Léxico-Semántico, se puede apreciar que al usar atributos léxicos con ligero soporte semántico se obtienen valores de correlación alrededor del 75%. Pero también, al unir este resultado con todos los atributos que incluyen otros aportes léxicos, como los reflejados en los grupos F2 y F3 (medidas del alineamiento léxico y medidas léxicas obtenidas con SimMetrics Library) se obtiene una correlación de aproximadamente un 80%, una mejoría de un 5% con relación al resultado obtenido anteriormente. Esto corrobora que la utilización de un alineamiento léxico, reforzado con las distancias léxicas y medidas de similitud, desde una perspectiva multidimensional, permite obtener resultados importantes en la determinación de similitud entre frases. Estos resultados se confirman en las dos competiciones internacionales en las que se validó este método, SemEval-2012 y SemEval-2013.

Alineamiento Semántico

Luego de efectuados los experimentos con el Alineamiento Semántico, se llega a la conclusión de que al incorporar un aporte semántico multinivel más profundo, ofrecido a través de la combinación de diferentes recursos, se consigue mejorar los resultados de la medición de similitud entre frases con respecto al Alineamiento Léxico-Semántico hasta un 22%. Especialmente se observa la mejoría cuando el corpus posee un fuerte componente semántico, en el que el método léxico obtiene pobres resultados (ver Tabla 3.31).

Aunado a esto, al combinar ambos métodos se puede apreciar una mejoría de casi un 10%, incluso en aquellos corpus con poca carga semántica. Estos resultados se pueden apreciar en la Tabla 3.81.

Tabla 3.81. Resultados y ranking de los tres métodos en la competición de SemEval-2012

Método	ALL	Rank	ALLnrm	RankNrm	Mean	RankMean
AS+ALS	0.7213	18	0.8239	14	0.6158	15
ALS	0.6630	26	0.7922	46	0.5560	49
AS	0.6529	29	0.8115	23	0.6116	16

Reconocimiento de la Paráfrasis

Para concluir sobre los resultados obtenidos en la experimentación en el Reconocimiento de la Paráfrasis, a través de los métodos no supervisados, se puede plantear que aplicando técnicas relativamente sencillas se logran resultados comparables a los obtenidos internacionalmente por otros investigadores. Se demostró, a través de los resultados de la variante 3 del método de reconocimiento de la paráfrasis, que utilizando solamente el recurso Freeling y el sentido más

frecuente, se logran resultados comparables y hasta superiores en exactitud y cobertura que con la combinación con Freeling y WordNet. Esto permitiría disminuir considerablemente la carga computacional del método.

Continuando con esta temática, se puede apreciar que el método NESP supera ligeramente a las variantes uno, dos y la tres del método no supervisado, excepto en la precisión alcanzada. Lo más importante en esta versión es la cobertura que logra alcanzar. Es de destacar que de las 1147 instancias positiva que tiene el corpus con que se trabajó, NESP solo deja de encontrar 51. Esto indica que la utilización de la semántica multinivel influye positivamente en captura de este fenómeno lográndose un 95.5% de detección de casos positivos.

Evaluating Phrasal Semantics

Como se ha podido ver en este capítulo, el método NESP también introduce un framework de gran utilidad para la tarea Evaluating Phrasal Semantics. Es importante destacar que este método fue el único que evaluó sus resultados para el corpus del italiano. Esto es debido a la facilidad de adaptar la arquitectura de este método a diferentes idiomas.

Otro elemento a favor del Alineamiento Semántico propuesto para el método NESP, es que proporciona una variante prometedora, basado simplemente en las relaciones de WordNet. Esté método alcanza la posición número seis, para el idioma inglés, en el ranking de SemEval-2013. Destacándose, con una pequeña diferencia de 0.07 puntos respecto a la exactitud, comparado con el trabajo de mejores resultados de este ranking. A pesar del problema ya señalado de la pobre clasificación de instancias positivas, como se ha visto este método obtiene resultados muy similares a los alcanzados por el mejor equipo de esta competición[66]; lo que indica lo acertado del proceder empleado.

El Stemming

Los experimentos realizados con relación al agrupamiento de familias de palabras, representan una muestra de las posibilidades de utilización del algoritmo de la Distancia Extendida en este tipo de agrupamiento de palabras y por consecuencia constituye un aporte al modelo de generación del stem que se ha propuesto. Debido a su basamento matemático, el algoritmo no necesita definir reglas para realizar el proceso de stemming, lo que le permite operar en diferentes idiomas sin necesidad de grandes cambios.

[66] Christian_wartena. Team HsH.

Se debe tener en cuenta que en algunos casos el stem propuesto por este tipo de proceder, no es gramaticalmente correcto, pero a efectos de su función es útil porque permite agrupar las palabras en familias y obtener los lexemas. También se puede utilizar para la creación de bancos léxicos muy útiles en varias tareas del PLN.

Los algoritmos clásicos de stemming, aunque muy refinados para un idioma específico, no dejan de ser un procedimiento de tratamiento individual de palabras, lo que los hace dependientes del lenguaje.

Reconocimiento de Entidades

En el Reconocimiento de Entidades, es importante acotar que la combinación de atributos seleccionados, aunque pudieran seguirse refinando, es acertada, teniendo en cuenta que se obtienen resultados relevantes en la mayoría de los experimentos realizados.

Para este investigador, el hecho de que en la mayoría de los trabajos realizados en esta temática, solo se reflejen los resultados de la fase de clasificación, hace que se minimice la importancia que la fase de identificación posee para el perfeccionamiento de los sistemas encargados del reconocimiento de entidades.

Al dividir y evaluar por separado las fase de identificación y clasificación en el REN, se puede tener un mayor dominio del correcto funcionamiento de los métodos implementados. Permitiéndose de esta manera perfecciona, evaluar y comparar mucho mejor los resultados de los sistemas dedicados a esta tarea.

Los modelos de probabilidad condicional de Máxima Entropía obtienen valores superiores comparados con los clasificadores MSV, AD y RN. Esta técnica fue la más utilizada en el CONLL-2003 y los equipos que obtuvieron los primeros lugares en esta competición, para el Inglés y el Alemán, utilizaron ME (Tjong Kim Sang y De Meulder, 2003).

Generación de Expresiones Regulares

Teniendo en cuenta la diferencia evidente entre ambas vías de solución, como se puede observar en los ejemplos mostrados anteriormente (Figura 3.29, Figura 3.30 y Figura 3.31), se considera que queda demostrado, no sólo que el autómata es la vía más simple e intuitiva para resolver este tipo de problemas, sino también que es posible -a través de un recurso informático- obtener la expresión regular partiendo de la modelación gráfica del Autómata Finito. Esto sería un aporte importante a la escritura de patrones para los sistemas de PLN, pues evitaría tener que recordar la sintaxis exacta del lenguaje formal que se utiliza para representar las reglas.

Capítulo 4 Conclusiones, trabajos futuros y producción científica

4.1 Conclusiones generales

1. ¿Es posible prescindir de los recursos para el análisis morfo-léxico, sintáctico y semántico en el reconocimiento de entidades, para con ello evitar los errores acarreados por los recursos dedicados a estas tareas y, aún así, obtener resultados a la altura de los alcanzados a nivel internacional?

Analizando los resultados obtenidos por el método expuesto en el epígrafe 3.7.5, a partir de los clasificadores de Máxima Entropía, se puede ver que esta variante se ubica en la primera posición entre los trabajos que han medido su desenvolvimiento con los corpus del CONLL-2002. También otros clasificadores con los que se experimenta (SVM, AD, RN), obtienen resultados por encima del promedio de los obtenidos en esta competición. Estos resultados son obtenidos para los corpus en español, un lenguaje con una gramática bastante compleja. Esto hace que se pueda afirmar que la opción de prescindir de la fase de pre-procesamiento de los corpus, evitando así el acarreo de errores y la carga computacional que representa el pre-procesamiento (el análisis léxico-sintáctico o semántico) en este tipo de tarea, es factible.

2. ¿Puede lograrse un método que permita abstraer las complejidades de los lenguajes formales en la creación de patrones de extracción, a partir de expresiones regulares para su utilización en diferentes tareas del PLN?

A través de una representación gráfica como el autómata finito, se ha visto en los experimentos reflejados en el epígrafe 3.8.3 que resulta más simple crear un patrón a través de una representación simbólica, que utilizando un lenguaje regular. La dificultad para usar una sintaxis como esta, no solo influye a la hora de la creación de las expresiones, sino también para interpretarlas y modificarlas. También corrobora este resultado la utilización exitosa de este método en otros trabajos como son (Miranda, 2008; Fernández, Gutiérrez et al., 2011; García, 2011; León, 2011; Pérez, Fernández et al., 2011). Finalmente, una vez concluido todos los experimentos, se puede afirmar -partiendo de los resultados- que se hace más fácil obtener el patrón deseado a través del autómata finito que directamente con la expresión regular. Por tanto, este sería un posible método para la creación de patrones de extracción en aplicaciones de PLN, en el que se evitaría tener que lidiar directamente con el lenguaje regular.

3. ¿De qué forma mejorar la distancia de edición, utilizada en la medición de similitud, de manera que al comparar palabras con diferencias en la raíz -que provocan cambio en el significado- sean penalizadas consecuentemente?

Si se tienen en cuenta no solo las operaciones a realizar para transformar una palabra en otra, si no también el lugar donde se producen, dándole una importancia que estará en función de cuán cerca ocurren del inicio de las palabras y además se valida la significación del carácter involucrado en la operación, se pueden obtener resultados que superan a los alcanzados por otras métricas que con regularidad se utilizan en esta tarea.

4. ¿Se podrá utilizar la modificación de la distancia de edición en la generación automática de familias de palabras y obtener resultados por encima de un 90% de precisión en esta tarea?

A partir de la extensión realizada a la distancia de edición, no solo se puede utilizar esta métrica en el agrupamiento de familias de palabras, si no que se obtienen resultados superiores al 90% de precisión y exactitud en esta tarea. De esta forma, se logra un método de gran utilidad para el agrupamiento de bolsas de palabras que pertenecen al mismo campo semántico, lo que permite su utilización en otras tareas del PLN.

5. Mediante la modificación de la distancia de edición, ¿se podrá obtener un stemmer independiente del lenguaje, a partir del agrupamiento de familias de palabras, que obtenga resultados superiores al 95% de precisión, cobertura y exactitud?

Mediante el agrupamiento de familias utilizando la DEx, se logra identificar el lexema común de las palabras, por lo tanto se puede identifica, con buena exactitud, el stem que agrupa bajo un mismo concepto a dichas palabras. Además, esto se logra con una precisión, exactitud y cobertura superior al 95%.

6. ¿Qué influencias tiene la utilización de un alineamiento léxico y el semántico en reconocimiento de la similitud textual desde una perspectiva multidimensional?

Se ha podido ver, A través de los experimentos realizados, que el alineamiento léxico aporta buenos resultados en la tarea de determinar la similitud entre frases. Además, la incorporación de varias métricas como atributos en un sistema de aprendizaje automático, mejora considerablemente estos resultados. Así también, al incorporar un aporte semántico a este alineamiento, a partir de las relaciones de WordNet, mejora significativamente el resultado alcanzado. Pero, al incorporar la semántica proveniente de un análisis `multidimensional`, aumentan las posibilidades de alcanzar el objetivo previsto. También a quedado demostrado que no pueden desligarse el análisis léxico y el semántico, pues ambos forman el complemento que permite realizar esta tarea.

7. ¿Podrá mejorar la polaridad sentimental la determinación de la Implicación Textual?

Analizando los experimentos realizados con los corpus para el Reconocimiento del la Implicación Textual, se ha demostrado que la Polaridad Sentimental ejerce una marcada influencia en este proceso. De esta forma, se puede afirmar que si dos frases se encuentran en una relación de implicación deben compartir una misma polaridad o lo que es lo mismo: si dos frases no poseen la misma polaridad, entonces no debe tener una relación de implicación entre ellas.

4.2 Trabajos futuros

Toda la investigación realizada ha abierto las puertas a nuevas interrogantes y nuevos trabajos de investigación. A continuación se describen, dividiéndose por las diferentes tareas, los que serán foco de atención en próximos trabajos.

Para el método de Alineamiento Semántico, es preciso modificar el algoritmo de selección de costo mínimo entre los grupos. Por ejemplo, cuando se está valorando el caso del grupo `wale`, con todos los grupos de la frase mayor, se selecciona la distancia Dist=2 entre `wale` y `bear`; y luego el grupo `bear` sale de la búsqueda. O sea, en las siguientes iteraciones no se va a tener en cuenta este grupo. Esto trae consigo, que si en próximas iteraciones hay otra palabra -de la frase menor- que tenga una mejor relación con las del grupo de la frase mayor, nunca será seleccionada. Este es el caso del ejemplo en la Tabla 3.22, para los grupos `dogs` y `bear` su distancia es menor que la comparación entre `wale` y `bear`. Lo correcto sería hacer todos los cálculos y luego tomar la decisión sobre qué grupo de la frase menor se empareja mejor con los de la frase mayor.

Haciendo un análisis de los resultados obtenidos por las tres variantes no supervisadas de Reconocimiento de la Paráfrasis, se podría intentar aplicar una técnica de votación entre ella, ya que quedó demostrado en lo experimentos que cada una es mejor que la otra en un determinado aspecto. Por ejemplo, la variante 3 es la que más casos positivos de paráfrasis detecta, pero la variante 1 ofrece una mejor precisión. Por otra parte, la variante 2 es la que captura más casos en los que no existe paráfrasis. De aquí que podrían funcionar mejor si la decisión se tomara teniendo en cuenta los resultados de las tres.

Para la tarea EPS en particular, el trabajo futuro en este sentido debe estar encaminado a resolver el problema con las instancias mal clasificadas. Además, se pretende introducir sustituciones léxica (synonyms) para expandir el corpus; así como aplicar similitud semántica conceptual, usando los árboles semánticos relevantes (Gutiérrez, Fernández et al., 2010a; 2011).

En cuanto al Reconocimiento de Entidades, aunque en esta investigación no se ha comprobado, se podría pensar a priori que las distintas clases (Localidad, Persona, Organización y Miscelánea), llamadas comúnmente entidades fuertes, pueden ser clasificadas mejor usando diferentes combinaciones de clasificadores para cada una de ella. También se propone experimentar con el uso de clasificadores de voting y con meta-clasificadores.

Después de desarrollada la modificación a la distancia extendida y la comprobación de sus buenos resultados, solo resta enfocarse en algunos elementos susceptibles a mejorar en el algoritmo. En este sentido, sería necesario analizar la posibilidad de modificar el algoritmo para obtener la subsecuencia común más larga, de forma tal que se obtenga la subsecuencia óptima. Esto implicaría el cálculo de todas las subsecuencia y una vez obtenido, seleccionar la mejor.

Para la utilización de la DEx en la obtención del stem, existen dos tareas pendientes a desarrollar, primeramente se necesita crear un método de obtención de un umbral de corte dinámico y no un valor fijo obtenido a través de la experimentación, esto permitiría reducir los errores por inclusión o exclusión de palabras en un determinado concepto. En segundo lugar, que planteado el estudio de una variante en la que puedan agruparse, o al menos quedar enlazadas bajo el mismo concepto, aquellas palabras que han sido excluidas por que constituyen flexiones en las que se modifica el lexema. Este es el caso, para el español por ejemplo, de los verbos irregulares. Esta modificación estaría encaminada en valorar la posible utilización de un recurso como WordNet o un lematizador.

Para continuar la investigación sobre la influencia de la Polaridad Sentimental en el Reconocimiento de la Implicación Textual, se propone utilizar una versión modificada del algoritmo de ranking (RA-SR) utilizando bigramas, y una nueva propuesta que utiliza un puntuador de skipgrams (Fernández, Gutiérrez et al., 2013).

4.3 Producción científica

En este epígrafe se resume toda la labor científica desarrollada a lo largo de esta investigación. Se recogen los trabajos realizados desde el año 2004 hasta la fecha y se divide en dos grupos, publicaciones y participación en competiciones internacionales.

4.3.1 Publicaciones

Año 2013

1. Vila Katia, **Fernández Antonio C.**, Gómez José M., Ferrández Antonio, Díaz Josval: Noise-tolerance feasibility for restricted-domain Information Retrieval systems. Data Knowl. Eng. 86: 276-294 (2013)
2. Chávez Alexander, **Fernández Antonio C.**, Dávila Héctor, Gutiérrez Yoan, et al. UMCC_DLSI: Textual Similarity based on Lexical-Semantic features. Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task, pages 109–118, Atlanta, Georgia, June 13-14, 2013. Association for Computational Linguistics.
3. Dávila Héctor, **Fernández Antonio C.**, Chávez Alexander, Gutiérrez Yoan. UMCC_DLSI-(EPS): Paraphrases Detection Based on Semantic Distance. Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 93–97, Atlanta, Georgia, June 14-15, 2013. Association for Computational Linguistics.
4. Gutiérrez Yoan, Castañeda Yenier, González Andy, Estrada, Rainel, Piug, Dennys, Abreu Jose I., Pérez Roger, **Fernández, Antonio C.,** et al. UMCC_DLSI: Reinforcing a Ranking Algorithm with Sense Frequencies and Multidimensional Semantic Resources to solve Multilingual Word Sense Disambiguation. Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 241–249, Atlanta, Georgia, June 14-15, 2013. Association for Computational Linguistics.
5. Collazo Armando, Ceballo Alberto, Puig Dennys, Gutiérrez Yoan, **Fernández Antonio C.**, et al. UMCC_DLSI: Semantic and Lexical features for detection and classification Drugs in biomedical texts. Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 636–643, Atlanta, Georgia, June 14-15, 2013. Association for Computational Linguistics.

Año 2012

6. **Fernández Antonio C.**, Gutiérrez Yoan, et al. (2012). Approaching Textual Entailment with Sentiment Polarity. Proceeding of ICAI'12 - The 2012 International Conference on Artificial Intelligence, Las Vegas, Nevada, USA.
7. **Fernández Antonio C.**, Gutiérrez Yoan, et al. (2012). UMCC-DLSI: multidimensional lexical-semantic textual similarity. Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM), pages 608–616, Montréal, Canada, June 7-8, 2012. 2012 Association for Computational Linguistics.
8. Díaz Héctor, **Fernández Antonio C.**, Gutiérrez Yoan, et al. (2012). Método de Extracción de Información Semántica en ontologías. SEPLN. XXVIII Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural, España.

Año 2011

9. **Fernández Antonio C.**, Díaz Josval, Gutiérrez Yoan and Muñoz Rafael (2011). An Unsupervised Method to Improve Spanish Stemmer. *Natural Language Processing and Information Systems*: Springer Berlin / Heidelberg) pp 221-4.
10. **Fernández Antonio C.**, Gutiérrez Yoan, Pérez Abel, García Yaniseth and Miranda Lisandra (2011). Extracción de Información en Documentos no Etiquetados del Entorno Educacional. In: *Xii Simposio Internacional De Comunicación Social. Centro de Linguistica Aplicada Santiago de Cuba,* (Cuba), pp 970-4.
11. **Fernández Antonio C.**, Chávez Alexander (2011). Métodos de Alineamiento y extracción de atributos para la detección de implicación textual usando aprendizaje automático. CIUM'2011: V convención científica internacional de la Universidad de Matanzas. Matanzas, Cuba, 7-11 noviembre 2011. ISBN: 9591613997, 9789591613998.
12. **Fernández Antonio C.,** Cobarrubia Ángel (2011). Un recurso para el preprocesamiento e indexado de documentos digitales. CIUM'2011: V convención científica internacional de la Universidad de Matanzas. Matanzas, Cuba, 7-11 noviembre 2011. ISBN: 9591613997, 9789591613998.
13. **Fernández Antonio C.**, Collazo Armando (2011). Módulo de segmentación y análisis morfológico de documentos digitales. CIUM'2011: V Convención Científica Internacional de la Universidad de Matanzas. Matanzas, Cuba, 7-11 noviembre 2011. ISBN: 9591613997, 9789591613998.
14. **Fernandez Antonio**, Gutiérrez Yoan, Pérez Abel, et al. (2011). Extracción de Información en documentos no etiquetados del entorno educacional. In: XII Simposio Internacional De Comunicación Social. Centro de Linguistica Aplicada Santiago de Cuba, (Cuba), Actas II, pp 881-10.
15. Pérez Abel, **Fernández Antonio C.**, Gutiérrez Yoan and Alfonso Yeiniel (2011). Generación de Expresiones Regulares para La Creación de Reglas en Aplicaciones de PLN. In: XII Simposio Internacional De Comunicación Social. Centro de Linguistica Aplicada Santiago de Cuba, (Cuba), Actas II, pp 881-5.
16. Gutiérrez Yoan, **Fernández Antonio C.**, Montoyo Andrés and Vázquez Sonia 2011 Enriching the Integration of Semantic Resources based on WordNet *Procesamiento del Lenguaje Natural* **47** 249-57.
17. Vila Katia, Díaz Josval, **Fernández Antonio C.**, et al. (2010). An approach for adding noise-tolerance to restricted-domain information retrieval. Natural Language Processing and Information Systems, Springer**:** 1-12.

Año 2010

18. Gutiérrez Yoan, **Fernández Antonio C.**, Montoyo Andrés and Vázquez Sonia 2010 Integration of semantic resources based on WordNet. In: *XXVI Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural,* ed S 2010 (Universidad Politécnica de Valencia, Valencia: SEPLN 2010) pp 161-8.
19. Gutiérrez Yoan, **Fernández Antonio C.**, Montoyo Andrés and Vázquez Sonia 2010 UMCC-DLSI: Integrative resource for disambiguation task. In: *Proceedings of the 5th International Workshop on Semantic Evaluation,* (Uppsala, Sweden: *Association for Computational Linguistics*) pp 427-32.

Año 2009

20. **Fernández Antonio C.**, Díaz Josval, Fundora Alfredo, Muñoz Rafael (2009). Un Algoritmo para la extracción de características lexicográficas en la comparación de Palabras. CIUM'2009:

VI Convención Científica Internacional de la Universidad de Matanzas. Matanzas, Cuba. junio 16-18.

Año 2008

21. **Fernández Antonio C.**, Calderín Yanoski, García Yaniseth (2006). Un sistema de extracción de información de los Programas de Disciplinas. Universidad 2006. Matanzas, Cuba, 2008.

Año 2005

22. **Fernández Antonio C.**, García Yaniseth (2005). Las Expresiones Regulares. Ejemplo de su aplicación en un sistema de Extracción de Información. MateCompu'2005: VII Congreso Internacional de Matemática y Computación. Revista Científica de la Universidad de Matanzas. Reporte in extenso. CD. Matanzas, Cuba. Instituto Superior Pedagógico Juan Marinello. ISSN:1682-2749. RNPS:1842.
23. Vila Katia, **Fernández Antonio C.** (2005). Interfaz de Acceso en Lenguaje Natural a la Información de la Universalización de la Enseñanza Superior. Monografías 2005. Universidad de Matanzas, Cuba. http://monografias.umcc.cu/.

Año 2004

24. **Fernández Antonio C.**, Muñoz Rafael, Suárez Armando (2005). Conference Proceeding: REME: Reconocedor de Entidades basado en Máxima Entropía. III Jornadas en Tecnología del Habla, Valencia. España; 11/2004.

4.3.2 Participación en competiciones científicas

1. Competición de Semeval-2010. Association for Computational Linguistics para la tarea "Word Sense Disambiguation on Specific Domain". Sweden, 2010.
2. Competición de SemEval-2012. Association for Computational Linguistics para la "tarea Lexical and Computational Semantics", Montréal, Canada, June 7-8, 2012. 2012.
3. Competición de SemEval-2013. Association for Computational Linguistics para la tarea "Lexical and Computational Semantics", Atlanta, Georgia, June 13-14, 2013.
4. Competición de SemEval-2013 Task 9: Extraction of Drug-Drug Interactions from Biomedical Texts (DDIExtraction 2013), Atlanta, Georgia, June 13-14, 2013.
5. Competición de SemEval-2013 task 5: Evaluating phrasal semantics. Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), Atlanta, Georgia, June 13-14, 2013.

Referencias Bibliográficas

Academia Española, R. (2001). "Diccionario de la Lengua Española 22 a ed." Edición. Madrid: RAE.

Adams, R., G. Nicolae, et al. (2007). Textual entailment through extended lexical overlap and lexico-semantic matching. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, Association for Computational Linguistics.

Agirre, E., D. Cer, et al. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (* SEM 2012).

Agirre, E., D. Cer, et al. (2013a). *Sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. SEM 2013: The Second Joint Conference on Lexical and Computational Semantics.

Agirre, E., D. Cer, et al. (2013b). *SEM 2013 Shared Task: Semantic Textual Similarity including a Pilot on Typed-Similarity. *SEM 2013: The Second Joint Conference on Lexical and Computational Semantics, Association for Computational Linguistics.

Al-Shalabi, R., G. Kannan, et al. (2005). "Experiments with the successor variety algorithm using the cutoff and entropy methods." Information Technology Journal **4**(1): 55-62.

Allan, J., B. Croft, et al. (2012). Frontiers, challenges, and opportunities for information retrieval: Report from swirl 2012 the second strategic workshop on information retrieval in lorne. ACM SIGIR Forum, ACM.

Appel, A. W., J. S. Mattson, et al. (1989). "A lexical analyzer generator for Standard ML." Distributed with Standard ML of New Jersey.

Appelt, D. E., J. R. Hobbs, et al. (1993). FASTUS: A finite-state processor for information extraction from real-world text. INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, LAWRENCE ERLBAUM ASSOCIATES LTD.

Arevalo Rodriguez, M., M. C. Torruella, et al. (2004). "MICE: a module for Named Entities Recognition and Classification." International Journal of Corpus Linguistics **9**(1): 53-68.

Asahara, M. and Y. Matsumoto (2003). Japanese named entity extraction with redundant morphological analysis. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, Association for Computational Linguistics.

Atserias, J., B. Casas, et al. (2006). FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06).

Ávila, D., Y. García, et al. (2010a). Baseline 1.1. Matanzas, Universidad de Matanzas.

Ávila, D., Y. García, et al. (2010b). Baseline 2.0. Matanzas, Universidad de Matanzas.

Baeza-Yates, R. and B. Ribeiro-Neto (1999). Modern information retrieval, ACM press New York.

Balahur, A. and A. Montoyo (2010). Semantic approaches to fine and coarse-grained feature-based opinion mining. Natural Language Processing and Information Systems, Springer**:** 142-153.

Bar-Haim, R., I. Dagan, et al. (2006). The Second Pascal Recognising Textual Entailment challenge. Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment.

Bar-Haim, R., I. Dagan, et al. (2007). Semantic Inference at the Lexical-Syntactic Level. Proceedings of AAAI-07.

Bar-Haim, R., I. Szpektor, et al. (2005). Definition and Analysis of Intermediate Entailment Levels. Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Association for Computational Linguistics.

Bär, D., C. Biemann, et al. (2012). Ukp: Computing semantic textual similarity by combining multiple content similarity measures. Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics.

Barzilay, R. and L. Lee (2005) "Learning to Paraphrase An Unsupervised Approach Using Multiple-Sequence Alignment." 8.

Barzilay, R. and K. R. McKeown (2001). Extracting paraphrases from a parallel corpus. Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics.

Beltrán, C. (2007). "Comparación de Sistemas para la Detección de Límites de Oraciones." Revista INFOSUR. Nro **1**.

Bender, O., F. J. Och, et al. (2003). Maximum entropy models for named entity recognition. Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics.

Bentivogli, L., P. Clark, et al. (2010). "The Sixth PASCAL Recognizing Textual Entailment Challenge." Proceedings of TAC.

Bentivogli, L., P. Clark, et al. (2011). "The Seventh Pascal Recognizing Textual Entailment Challenge." Proceedings of TAC.

Bentivogli, L., I. Dagan, et al. (2009). "The Fifth PASCAL Recognizing Textual Entailment Challenge." Proceedings of TAC **9**: 14-24.

Berger, A. L., S. A. D. Pietra, et al. (1996). A maximum entropy approach to natural language processing.**:** 39-71.

Berkhin, P. (2006). A survey of clustering data mining techniques. Grouping multidimensional data, Springer**:** 25-71.

Berry, M. W. (2004). Survey of Text Mining I: Clustering, Classification, and Retrieval, Springer.

Bhole, A., B. Fortuna, et al. (2007). "Extracting named entities and relating them over time based on Wikipedia." INFORMATICA-LJUBLJANA- **31**(4): 463.

Bikel, D. M., S. Miller, et al. (1997). Nymble: a high-performance learning name-finder. Proceedings of the fifth conference on Applied natural language processing, Association for Computational Linguistics.

Bilenko, M. and R. J. Mooney (2003). "Adaptive duplicate detection using learnable string similarity measures." KDD: 39.

Black, W. J., F. Rinaldi, et al. (1997). FACILE: Description of the ne system used for MUC-7. Manchester, Department of Language Engineering UMIST**:** 10.

Black, W. J. and A. Vasilakopoulos (2002). Language independent named entity classification by modified transformation-based learning and by decision tree induction. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Blum-Kulka, S. (1996). "Introducción a la pragmática del interlenguaje." La competencia pragmática: elementos lingüísticos y psicosociales: 155-175.

Borthwick, A., J. Sterling, et al. (1998). Description of the MENE Named Entity System used for MUC-7. Proceedings of the Seventh Message Understanding Conference (MUC-7).

Borthwick, A., J. Sterling, et al. (1998.). Description of the MENE Named Entity System used for MUC-7. Proceedings of the Seventh Message Understanding Conference (MUC-7).

Braschler, M. and P. Schäuble (2000). Experiments with the Eurospider REtrieval System for CLEF 2000. Cross-Language Information Retrieval an Evaluation, Workshop of the Cross-Aln guage Evaluation Forum, CLEF 2000.

Breiman, L. (1996). "Bagging predictors." Machine Learning **24**(2): 123-140.

Brena, R. (2003). Autómatas y Lenguajes. Monterrey, Tecnológico de Monterrey.

Brockett, C. (2007) "Aligning the RTE 2006 Corpus." Microsoft Research Technical Report MSR-TR-2007-77.

Budanitsky, A. and H. Graeme (2001). Semantic distance inWordNet: An experimental, application-oriented evaluation of five measures. Workshop on wordnet and other lexical resources, second meeting of the North American Chapter of the Association for Computational Linguistics.

Budi, I. and S. Bressan (2003). Association rules mining for name entity recognition. Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on, IEEE.

Burger, J. D., J. C. Henderson, et al. (2002). Statistical named entity recognizer adaptation. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Burkard, R., M. Dell'Amico, et al. (2012). Assignment problems, Cambridge University Press.

Callison-Burch, C., P. Koehn, et al. (2006) "Improved Statistical Machine Translation Using Paraphrases." 8.

Carreras, X., L. Marquez, et al. (2002). Named entity extraction using AdaBoost. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Carroll, J., T. Briscoe, et al. (1998). Parser evaluation: a survey and a new proposal. Proceedings of the 1st International Conference on Language Resources and Evaluation.

Chang, A. X. and C. Manning (2012). SUTime: A library for recognizing and normalizing time expressions. LREC.

Chapman, S. and C. Parkinson (2006). SimMetrics library v 1.5 for .NET 2.0 System and Reference Manual. U. o. Sheffield. United Kingdom.

Charniak, E. and D. McDermott (1985). Introduction to artificial intelligence, Pearson Education India.

Chávez, A. and A. C. Fernández (2012). AlinRTE V 1.0. Matanzas, Universidad de Matanzas**:** Prototipo para el estudio de la similitud textual.

Chieu, H. L. and H. T. Ng (2002). Named entity recognition: a maximum entropy approach using global information. Proceedings of the 19th international conference on Computational linguistics-Volume 1, Association for Computational Linguistics.

Chinchor, N. A. (1999). Overview of muc-7/met-2. Message Understanding Conference MUC-7.

Chinchor, N. A., P. Robinson, et al. (1998). "Hub-4 Named Entity task definition version 4.8." Available by ftp from www. nist. gov/speech/hub4_98.

Chomsky, N. (1956). "Three models for the description of language." Information Theory, IRE Transactions on **2**(3): 113-124.

Chomsky, N. (1957). Syntactic Structures, Mouton and Co., The Hague.(1965) Aspects of the Theory of Syntax, The MIT Press.

Chong Tat Chua, F. and S. Asur (2013). "Automatic Summarization of Events From Social Media."

Christensen, J., S. S. Mausam, et al. (2013). Towards Coherent Multi-Document Summarization. Proceedings of NAACL-HLT.

Cimiano, P. and J. Völker (2005). "Towards large-scale, open-domain and ontology-based named entity classification."

Cohen, W. W., P. D. Ravikumar, et al. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. IIWeb.

Contreras, H. Y. (2001). Procesamiento del lenguaje natural basado en una" gramática de estilos" para el idioma español. Facultad de Ingeniería. Colombia, Universidad de los Andes **Propuesta de tesis**.

Corley, C. and R. Mihalcea (2005). Measuring the semantic similarity of texts. Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Association for Computational Linguistics.

Cormen, T. H., C. E. Leiserson, et al. (2001). Introduction to algorithms, MIT press.

Cucerzan, S. and D. Yarowsky (1999.). Language independent named entity recognition combining morphological and contextual evidence. In Proceedings of 1999 Joint SIGDAT Conference on EMNLP and VLC, University of Maryland.

Cucerzan, S. and D. Yarowsky (2002). Language independent NER using a unified model of internal and contextual evidence. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Curran, J. R. and S. Clark (2003). Language independent NER using a maximum entropy tagger. Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics.

Daelemans, W., J. Zavrel, et al. (2004). "Timbl: Tilburg memory-based learner." Tilburg University.

Dagan, I., O. Glickman, et al. (2005). The PASCAL Recognizing Textual Entailment Challenge. Proceedings of the Workshop on Recognising Textual Entailment, Southampton, UK.

Dagan, I., O. Glickman, et al. (2006). "The pascal recognizing textual entailment challenge." Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: 177-190.

Dagan, I., L. Lee, et al. (1999). "Similarity-based models of word cooccurrence probabilities." Machine Learning **34**(1): 43-69.

Dakka, W. and S. Cucerzan (2008). Augmenting wikipedia with named entity tags. Proceedings of the 3rd International Joint Conference on Natural Language Processing.

Dalvi, B. B., W. W. Cohen, et al. (2012). Websets: Extracting sets of entities from the web using unsupervised information extraction. Proceedings of the fifth ACM international conference on Web search and data mining, ACM.

Dávila, H., Y. Gutierrez, et al. (2012). SemanticAlin V 1.0. Matanzas, Universidad de Matanzas**:** Determinador se similitud mediante Alineamiento Semántico.

Dawson, J. L. (1974) "Suffix Removal and Word Conflation." ALLC Bulletin, Michaelmas, 33-46.

De la Cueva, O., A. M. González, et al. (2005). Manual de Gramática Española I. La Habana, Editorial Félix Varela.

De la Vega, A., A. Pérez, et al. (2012). Prototipo para el reconocimiento de Expresiones Regulares. Matanzas, Universidad de Matanzas.

Delmonte, R., S. Tonelli, et al. (2006). Venses–a linguistically-based system for semantic evaluation. Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, Springer**:** 344-371.

Deza, M.-M. and E. Deza (2006). Dictionary of distances, Elsevier.

Díaz, J. and A. C. Fernández (2009). Estemizador V 1.0. Matanzas, Universidad de Matanzas**:** Extractor de características lexicográficas.

Dice, L. R. (1945). "Measures of the amount of ecologic association between species." Ecology **26**(3): 297-302.

Doan, A., R. Ramakrishnan, et al. (2006). Managing information extraction: state of the art and research directions. Proceedings of the 2006 ACM SIGMOD international conference on Management of data, ACM.

Dolan, B., C. Quirk, et al. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. Proceedings of the 20th international conference on Computational Linguistics, Association for Computational Linguistics.

Elmasri, R. and S. Navathe (2009). "Fundamentals of Database Systems."

Esuli, A. and F. Sebastiani (2006). SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. Fifth international conference on Languaje Resources and Evaluation

Fernández, A. C. (2005). Reconocimiento de Entidades utilizando la combinación de Técnicas Basadas en Conocimiento y Modelos de Probabilidad Condicional de Máxima Entropía. Departamento de Lenguajes y Sistemas Informáticos. Alicante, Universidad de Alicangte. **DEA**.

Fernández, A. C., J. Díaz, et al. (2009). Un algoritmo para la extracción de características lexicográficas en la comparación de palabras. IV Convención Científica Internacional CIUM, Matanzas, Cuba.

Fernández, A. C., Y. Gutiérrez, et al. (2012a). UMCC-DLSI: multidimensional lexical-semantic textual similarity. Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. Montreal, Canada, Association for Computational Linguistics.

Fernández, A. C., Y. Gutiérrez, et al. (2012b). Approaching Textual Entailment with Sentiment Polarity. proceeding of ICAI'12 - The 2012 International Conference on Artificial Intelligence, Las Vegas, Nevada, USA.

Fernández, A. C., Y. Gutiérrez, et al. (2011). Extracción de Información en documentos no etiquetados del entorno educacional. XII Simposio Internacional de Comunicación Social, Santiago de Cuba, Centro de Lingüística Aplicada.

Fernández, A. C. and R. Muñoz (2005). REME V 1.0. Alicante, Universidad de Alicante**:** Reconocedor de Entidades Nombradas.

Fernández, A. C., R. Muñoz, et al. (2004). REME: Reconocedor de entidades basado en Máxima Entropía. III Jornadas en Tecnología del Habla, Valencia

Fernández, J., Y. Gutiérrez, et al. (2013). Sentiment Analysis of Spanish Tweets Using a Ranking Algorithm and Skipgrams. Proc. of the TASS workshop at SEPLN 2013. IV Congreso Español de Informática.

Fernández Orquín, A. (2005). Reconocimiento de Entidades utilizando la combinación de Técnicas Basadas en Conocimiento y Modelos de Probabilidad Condicional de Máxima Entropía. Departamento de Lenguajes y Sistemas Informáticos. Alicante, Universidad de Alicangte. **DEA**.

Fernández Orquín, A., R. Muñoz Guillena, et al. (2004). REME: Reconocedor de entidades basado en Máxima Entropía. III Jornadas en Tecnología del Habla, Valencia

Fernando, S. (2007). Paraphrase Identification. Department of Computer Science, University of Sheffield. **Master of Science:** 56.

Fernando, S. and M. Stevenson (2008). "A semantic similarity approach to paraphrase detection." Computational Linguistics UK (CLUK 2008) 11th Annual Research Colloqium.

Ferrández, O., Z. Kozareva, et al. (2005). "Nerua: sistema de detección y clasificación de entidades utilizando aprendizaje automático." Procesamiento del lenguaje natural **35**(37-44): 94.

Florian, R. (2002). Named entity recognition as a house of cards: Classifier stacking. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Formella, A. (2010). "Teoría de Autómatas y Lenguajes Formales."

Frakes, W. B. (1992). "Stemming algorithms." Information Retrieval: data structures and algorithms: 131-160.

Freire, N., J. Borbinha, et al. (2012). An analysis of the named entity recognition problem in digital library metadata. Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries, ACM.

Friedl, J. (2006). Mastering regular expressions, O'Reilly Media, Incorporated.

Gaizauskas, R., K. Humphreys, et al. (1995). University of Sheffield: description of the LaSIE system as used for MUC-6. Proceedings of the 6th conference on Message understanding, Association for Computational Linguistics.

García, I. (2011). RERAS: Sistema de Reconocimiento de Entidades combinando Expresiones Regulares y Aprendizaje Supervisado. Departamento de Informática. Matanzas, Camilo Cienfuegos. **Trabajo de Diploma en opción al Título de Ingeniería Informática:** 72.

García, I. and A. C. Fernández (2009). Rec_Ent-JApp V 1.0. Matanzas, Universidad de Matanzas**:** Reconocedor de Entidades Nombradas.

García Vasconcelos, Y., A. Fernández Orquín, et al. (2005). Las expresiones regulares. Ejemplo de su aplicación en un sistema de Extracción de Información. Congreso internacional de matemáticas y computación (MATECOMPU 2005), Matanzas, Cuba.

García, Y. and A. C. Fernández (2008). Baseline 1.0. Matanzas, Universidad de Matanzas.

García, Y., A. C. Fernández, et al. (2005). Las expresiones regulares. Ejemplo de su aplicación en un sistema de Extracción de Información. MateCompu-2005. Evento Internacional de Matemática y Computación, Matanzas, Instituto Superior Pedagógico "Jaun Marinello".

Gelbukh, A. (2010) "Procesamiento de lenguaje natural." 9.

Giampiccolo, D., H. T. Dang, et al. (2008). The fourth Pascal Recognizing Textual Entailment Challenge. Proceedings of the First Text Analysis Conference (TAC 2008).

Giampiccolo, D., H. T. Dang, et al. (2009). The fourth Pascal Recognizing Textual Entailment Challenge. Proceedings of the First Text Analysis Conference (TAC 2008).

Giampiccolo, D., B. Magnini, et al. (2007). The Third Pascal Recognizing Textual Entailment Challenge. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, Association for Computational Linguistics.

Giannakidou, A. (2002). "Licensing and sensitivity in polarity items: from downward entailment to nonveridicality." CLS **38**: 29-53.

Glickman, O., I. Dagan, et al. (2006). A lexical alignment model for probabilistic textual entailment. Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, Springer**:** 287-298.

Glickman, O., E. Shnarch, et al. (2006). Lexical reference: a semantic matching subtask. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Goldsmith, J. (2001). "Unsupervised learning of the morphology of natural language." Computational Linguistics **27(2)**: 153–198.

Gómez Guinovart, J. (2007). "Fundamentos de Lingüística Computacional: bases teóricas, líneas de investigación y aplicaciones." Bibliodoc: anuari de biblioteconomia, documentació i informació: 135-146.

Gómez, J. (2007). "Fundamentos de Lingüística Computacional: bases teóricas, líneas de investigación y aplicaciones." Bibliodoc: anuari de biblioteconomia, documentació i informació: 135-146.

Gossen, N. T. G., N. Kanhabua, et al. (2012). "NEER: An Unsupervised Method for Named Entity Evolution Recognition."

Gotoh, O. (1982). "An improved algorithm for matching biological sequences." Journal of molecular biology **162**(3): 705-708.

Graña Gil, J. (2002). "Técnicas de análisis sintáctico robusto para la etiquetación del lenguaje natural." Procesamiento del lenguaje natural(28): 117-118.

Gravano, L., P. G. Ipeirotis, et al. (2001). "Using q-grams in a DBMS for Approximate String Processing." IEEE Data Eng. Bull. **24**(4): 28-34.

Grishman, R. and B. Sundheim (1996). Message understanding conference-6: A brief history. Proceedings of COLING.

Guerrero, M., M. J. García, et al. (2010). "Propuesta para la integración de expresiones temporales procedentes de patrimonio documental en un SIG." Revista Catalana de Geografía **15**(40).

Gupta, V. and G. S. Lehal (2010). "A survey of text summarization extractive techniques." Journal of Emerging Technologies in Web Intelligence **2**(3): 258-268.

Gutiérrez, Á. B. and J. J. S. Martín "SPNER–Reconocedor de entidades nombradas para el español."

Gutierrez, Y. (2010). Resolución de ambiguedad semántica mediante el uso de vectores de conceptos relevantes. Departamento de Informática. Matanzas, Camilo Cienfuegos. **Tesis de Maestría**.

Gutiérrez, Y., Y. Castañeda, et al. (2013). UMCC_DLSI: Reinforcing a Ranking Algorithm with Sense Frequencies and Multidimensional Semantic Resources to solve Multilingual Word Sense Disambiguation. Second Joint Conference on Lexical and Computational Semantics (*SEM), Atlanta, Georgia, Association for Computational Linguistics.

Gutiérrez, Y., A. C. Fernández, et al. (2010a). Integration of semantic resources based on WordNet. XXVI Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural, Universidad Politécnica de Valencia, Valencia, SEPLN 2010.

Gutiérrez, Y., A. C. Fernández, et al. (2010b). UMCC-DLSI: Integrative resource for disambiguation task. Proceedings of the 5th International Workshop on Semantic Evaluation, Uppsala, Sweden, Association for Computational Linguistics.

Gutiérrez, Y., A. C. Fernández, et al. (2011). "Enriching the Integration of Semantic Resources based on WordNet." Procesamiento del Lenguaje Natural **47**: 249-257.

Gutiérrez, Y., S. Vázquez, et al. (2011). Sentiment Classification Using Semantic Features Extracted from WordNet-based Resources. Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011), Portland, Oregon., Association for Computational Linguistics.

Hafer, M. A. and S. F. Weiss (1974). "Word segmentation by letter succesor varieties." Information Storage and Retrieval **10**: 371-385.

Hammarström, H. (2007). Unsupervised Learning of Morphology:Survey, Model, Algorithm and Experiments. Department of Computing Science Sweden, Chalmers University of Technology and Göteborg University. **Master Thesis for the Degree of Licentiate of Engineering:** 81.

Harabagiu, S. and A. Hickl (2006). Methods for using textual entailment in open-domain question answering. ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS.

Harris, Z. S. (1954). "Distributional structure." Word.

Hasegawa, T., S. Sekine, et al. (2005) "Unsupervised Paraphrase Acquisition via Relation Discovery." 8.

Hasegawa, T. S., Satoshi y Grishman, Ralph (2005) "Unsupervised Paraphrase Acquisition via Relation Discovery." 8.

Haya, S. P. (2006). "Edad y etapas en el aprendizaje de la negación en inglés como L3 en contextos formales." Revista española de lingüística aplicada(19): 143-162.

Hays, D. G. (1967). Introduction to computational linguistics, American Elsevier New York.

Herrera de la Cruz, J. (2005). Un Modelo Fundamentado en Análisis de Dependencias y WordNet para el Reconocimiento de Implicación Textual. Departamento de Lenguajes y Sistemas Informáticos. Madrid, España, Universidad Nacional de Educación a Distancia**:** 63.

Hickl, A. and J. Bensley (2007). A discourse commitment-based framework for recognizing textual entailment. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, Association for Computational Linguistics.

Hickl, A., J. Williams, et al. (2006). Recognizing Textual Entailment with LCC's GROUNDHOG System Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment.

Hirschberg, D. S. (1975). "A linear space algorithm for computing maximal common subsequences." ACM **18**(6): 341.

Hirschberg, D. S. (1977). "Algorithms for the longest common subsequence problem." J. ACM **24**: 664–675.

Ho, T. K. (1998). "The random subspace method for constructing decision forests." Pattern Analysis and Machine Intelligence, IEEE Transactions on **20**(8): 832-844.

Holzinger, A., C. Stocker, et al. (2013). Combining HCI, Natural Language Processing, and Knowledge Discovery-Potential of IBM Content Analytics as an assistive technology in the biomedical field.

Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data, Springer**:** 13-24.

Hontoria, H. R. (2003). "SIMILITUD SEMÁNTICA." Lexicografía computacional y semántica **64**: 119.

Hopcroft, J. E. and J. D. Ullman (2001). Introduction to Automata Theory, Languages, and Computation, Second Edition. Addison-Wesley Publishing Company.

Huenerfauth, M. (2003). "A survey and critique of american sign language natural language generation and machine translation systems." Computer and Information Sciences, University of Pennsylvania, Philadelphia.

Humphreys, K., R. Gaizauskas, et al. (1998). University of Sheffield: Description of the LaSIE-II system as used for MUC-7. Proceedings of the Seventh Message Understanding Conferences (MUC-7).

Hurtado Carmona, D. and J. Sarabia Agámez (2008). Construcción de expresiones regulares más cortas a partir de un autómata finito. Generación Digital. Puerto Colombia, Fundación Universitaria San Martín. **7:** 26.

Ibrahim, A. K., Boris y Lin, Jimmy (2003) "Extracting Structural Paraphrases from Aligned Monolingual Corpora." 8.

Iglesias, E., A. Seara Vieira, et al. (2013). "An HMM-based over-sampling technique to improve text classification." Expert Systems with Applications.

Izquierdo, R., A. Suárez, et al. (2007). "A Proposal of Automatic Selection of Coarse-grained Semantic Classes for WSD." Procesamiento del Lenguaje Natural **39**: 189-196.

Jaccard, P. (1908). Nouvelles recherches sur la distribution florale.

James, M. and M. Paul (2003). "Single N-gram stemming." Proceedings of the 16th annual international ACM SIGIR conference on Research and develomepment in information retrieval: 415-416.

Jansche, M. (2002). Named entity extraction with conditional markov models and classifiers. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Jaro, M. (1989). "Advances in record linking methodology as applied to the 1985 census of tampa Florida." Journal of the American Statistical Association **84**.

Jaro, M. (junio 1989). "Advances in record linking methodology as applied to the 1985 census of tampa Florida." Journal of the American Statistical Association **84**.

Jaynes, E. T. (1990). Notes on present status and future prospects.

In W.T. Grandy and L.H. Schick, editors, Maximum Entropy and Bayesian Methods**:** 1-3.

Jiang, J. J. and D. W. Conrath (1997). "Semantic similarity based on corpus statistics and lexical taxonomy." arXiv preprint cmp-lg/9709008.

Jijkoun, V. and M. d. Rijke (2005). Recognizing textual entailment using lexical similarity. Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment.

Jinxi, X. and C. Bruce (1998). "Corpus-based stemming using co-ocurrence of word variants." ACM Transactions on Information Systems **16**(1): 61-81.

Jivani, A. G. (2011). "A Comparative Study of Stemming Algorithms." Int. J. Comp. Tech. Appl **2**(6): 1930-1938.

Jonker, R. and A. Volgenant (1987). "A shortest augmenting path algorithm for dense and sparse linear assignment problems." Computing **38**(4): 325-340.

Kaesler, R. L. (1966). "Quantitative re-evaluation of ecology and distribution of recent Foraminifera and Ostracoda of Todos Santos Bay, Baja California, Mexico."

Kauchak, D. and R. Barzilay (2006) "Paraphrasing for Automatic Evaluation." 8.

Kaufmann, A. and R. Cruon (1967). La programación dinámica, Compañía Editorial Continental.

Kazarov, D. and S. Manandhar (2001). Unsupervised Learning ofWord Segmentation Rules with Genetic Algorithms and Inductive Logic Programming. Machine Learning. C. D. P. a. S. W. Luc De Raedt, Kluwer Academic Publishers. **43:** 121–162.

Kleene, S. C., N. de Bruijn, et al. (1971). Introduction to metamathematics, Wolters-Noordhoff Groningen.

Knuth, D. E. (2007). MiKTeX 2.6. http://www.miktex.org.

Koehn, P. and K. Knight (2009). Statistical machine translation, Google Patents.

Korkontzelos, I., T. Zesch, et al. (2013). SemEval-2013 task 5: Evaluating phrasal semantics. Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013).

Kraaij, W. and R. Pohlmann (1996). Viewing Stemming as Recall Enhancement. Proceeding of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzeland.

Krause, E. (1987). Taxicab geometry: An adventure in non-Euclidean geometry, DoverPublications. com.

Krovetz, R. (1993). Viewing morphology as an inference process. Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, ACM.

Kuhn, H. W. (1955). "The Hungarian Method for the assignment problem." Naval Research Logistics Quarterly **2**: 83–97.

Lafferty, J., A. McCallum, et al. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. Proc. 18th Internacional Conf. on Machine Learning, San Francisco.

Lalín, M. d. R. (2012). "Alineamiento y validación de terminologías a gran escala en el ámbito médico."

Lau, R. (1994). Adaptative statistical language modeling. MIT.

Lau, R., R. Rosenfeld, et al. (1993). Adaptative statistical language modeling using the maximum entropy principle. Proceedings of the Human Language Technology Workshop, ARPA.

Leacock, C. and M. Chodorow (1998). "Combining local context and WordNet similarity for word sense identification." WordNet: An electronic lexical database **49**(2): 265-283.

Lee, K.-J., Y.-S. Hwang, et al. (2003). Two-phase biomedical NE recognition based on SVMs. Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13, Association for Computational Linguistics.

Lee, L. (1999). Measures of distributional similarity. Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, Association for Computational Linguistics.

León, L. (2011). Herramienta Web para la Extracción de Información de los Programas de Asignatura. Departamento de Informática. Mantanzas, Camilo Cienfuegos. **Trabajo de Diploma en opción al Título de Ingeniería Informática:** 72.

Lepage, Y. and E. Denoual (2005) "Automatic generation of paraphrases to be used as translation references in objective evaluation measures of machine translation." 8.

Lepage, Y. y. D., Etienne (2005) "Automatic generation of paraphrases to be used as translation references in objective evaluation measures of machine translation." 8.

Lesk, M. E. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. Proceedings of the ACM SIGDOC Conference, Toronto, Ontario.

Lesk, M. E. and E. Schmidt (1975). Lex: A lexical analyzer generator, Bell Laboratories Murray Hill, NJ.

Levenshtein, V. I. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. Problems of information Transmission.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics - Doklady

Lewandowski, T. (2000). Diccionario de Lingüística. Madrid, Cátedra.

Li, H., A. Kumaran, et al. (2009). Report of NEWS 2009 machine transliteration shared task. Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, Association for Computational Linguistics.

Li, W., T. Liu, et al. (2005). Automated generalization of phrasal paraphrases from the web. Proceedings of IWP, Jeju Island, South Korea.

Li, Y. and K. Joshi (2012). "The State of Social Computing Research: A Literature Review and Synthesis using the Latent Semantic Analysis Approach."

Lin, C.-H., C.-T. Huang, et al. (2006). Optimization of regular expression pattern matching circuits on FPGA. Design, Automation and Test in Europe, 2006. DATE'06. Proceedings, IEEE.

Lin, D. (1998). An information-theoretic definition of similarity. Proceedings of the 15th international conference on Machine Learning, San Francisco.

Lin, D. and P. Pantel. (2002). "Discovery of Inference Rules for Question Answering." Retrieved junio, 2008.

Llopis, F. (1988). “EXIT:Propuesta de un sistema de extracción de información de textos notariales”. Novática.

Llorens Piñana, D. (2000). Suavizado de autómatas y traductores finitos estocásticos, Universitat Politècnica de València.

Lovins, J. B. (1968). Development of a stemming algorithm, Mechanical Translation and Computational Linguistics.

MacCartney, B., M. Galley, et al. (2008). A phrase-based alignment model for natural language inference. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

MacCartney, B., T. Grenager, et al. (2006). Learning to Recognize Features of Valid Textual Entailments Proceedings of NAACL-06, New York.

Magnini, B. and G. Cavaglia (2000). Integrating Subject Field Codes into WordNet. LREC.

Malouf, R. (2002). Markov models for language-independent named entity recognition. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Manning, C. D., P. Raghavan, et al. (2008). Introduction to information retrieval, Cambridge University Press Cambridge.

Mansouri, A., L. S. Affendey, et al. (2008). "Named entity recognition approaches." International Journal of Computer Science and Network Security **8**: 339-344.

Marsi, E. and E. Krahmer (2005). Classification of semantic relations by humans and machines. ACL-05 Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor.

Martí, F. J., J. F. Quesada, et al. (2011) "Procesamiento del Lenguaje Natural." 76.

Martí, M. A., A. Fernández, et al. (2003). Lexicografía computacional y semántica. Barcelona, Universidad de Barcelona.

Matsuo, Y., H. Tomobe, et al. (2004). Finding social network for trust calculation. ECAI.

McCallum, A., D. Freitag, et al. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. Machine Learning. Proceedings of the Seventeenth International Conference (ICML2000), Stanford, California.

McCallum, A. and W. Li (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics.

McDonald, D. (1996). Internal and external evidence in the identification and semantic categorization of proper names.

McNamee, P. and H. T. Dang (2009). Overview of the TAC 2009 knowledge base population track. Text Analysis Conference (TAC).

McNamee, P. and J. Mayfield (2002). Entity extraction without language-specific resources. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Meya, M. (1991). "Semántica del grupo preposicional." Procesamiento del lenguaje natural **9**.

Mihalcea, R., C. Banea, et al. (2012). Multilingual subjectivity and sentiment analysis. Tutorial Abstracts of ACL 2012, Association for Computational Linguistics.

Mihalcea, R., C. Corley, et al. (2006). Corpus-based and knowledge-based measures of text semantic similarity. Proceedings of the national conference on artificial intelligence, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Mikheev, A., C. Grover, et al. (1998). Description of the LTG system used for MUC-7. Proceedings of 7th Message Understanding Conference (MUC-7), Fairfax, VA.

Miller, G. A., R. Beckwith, et al. (1990). "Introduction to WordNet: An On-line Lexical Database." International Journal of Lexicography, 3(4):235-244.

Milone, D., A. Rubio, et al. (2001). Modelos de lenguaje variantes en el tiempo. Memorias del XXIV Congreso Nacional de Ingeniería Biomédica, Oaxtepec, México.

Miranda, L. (2008). Asistente para la generación de conferencias virtuales para la Educación Superior cubana. Departamento de Informática. Matanzas, Camilo Cienfuegos. **Trabajo de Diploma para optar por el título de Ingeniería Informática:** 93.

Moldovan, D. I. and V. Rus (2001). "Explaining Answers with Extended WordNet." ACL.

Monge, A. E. and C. Elkan (1996). The Field Matching Problem: Algorithms and Applications. KDD.

Montoyo, A. (2009). "Resolución de la ambigüedad semántica mediante métodos basados en conocimiento y su aportación a tareas de PLN."

Montoyo, A., P. Martínez-Barco, et al. (2012). "Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments." Decision Support Systems **53**(4): 675-679.

Monz, C. and M. de Rijke (2001). Light-weight entailment checking for computational semantics. Proc. of the third workshop on inference in computational semantics (ICoS-3).

Mooi, E. A. and M. Sarstedt (2011). A concise guide to market research: The process, data, and methods using IBM SPSS statistics, Springer.

Moore, E. F. (1956). Gedanken-experiments on Sequential Machines, Estudios de Autómatas, Anales de los Estudios Matemáticos Princeton University Press. Princeton, N. J. **no. 34:** 129 - 153.

Moreno, L., M. Palomar, et al. (1999). "Introducción al procesamiento del lenguaje natural." Servicio de Publicaciones Universidad de Alicante). Universidad de Alicante.

Moulinier, I., J. A. McCulloh, et al. (2001). "West Group at CLEF 2000: Non-English Monolingual Retrieval." Cross-Language Information Retrieval and Evaluation, Workshop of Cross-Language Evaluation Forum, CLEF 2000: 253-260.

Muñoz, R., P. Martinez-Barco, et al. (1999). "Método para la resolución de correferencias de sintagmas nominales definidos incluyendo alias y acrónimos en el sistema de extracción de información EXIT." Procesamiento del lenguaje natural **25**: 143-149.

Nadeau, D. and S. Sekine (2007). "A survey of named entity recognition and classification." Lingvisticae Investigationes **30**(1): 3-26.

Nadkarni, P. M., L. Ohno-Machado, et al. (2011). "Natural language processing: an introduction." J Am Med Inform Assoc **18**: 544e551.

Nairn, R., C. Condoravdi, et al. (2006). "Computing relative polarity for textual inference." Inference in Computational Semantics (ICoS-5): 20-21.

Navigli, R. (2009). "Word sense disambiguation: A survey." ACM Computing Surveys (CSUR) **41**(2): 10.

Neeedleman, S. and C. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." Mol. Biol **48**(443): 453.

Niles, I. and A. Pease (2001). Towards A Standard Upper Ontology. Proceedings of FOIS 2001, Ogunquit, Maine, USA. .

NTCIR-8_MOAT. (2010). "NTCIR-8 Workshop Meeting." NTCIR-8 Workshop Meeting, from http://research.nii.ac.jp/ntcir/ntcir-ws8/meeting/.

Och, F. J. and H. Ney (2004). "The alignment template approach to statistical machine translation." Computational Linguistics **30**(4): 417-449.

Och, F. J., C. Tillmann, et al. (1999). Improved alignment models for statistical machine translation. Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora.

Oliveira, O., M. Nunes, et al. (1998). "Por qué aún no podemos hablar con una computadora." Boletín de la Sociedad Mexicana de Física **12**(3): 145-150.

Ontiveros, N. J. and M. Pérez (2013). El lenguaje entre las computadoras y los humanos. Hypatia-Revista de Divulgación Científico Tecnológica. Estado de Morelos. México. **43**.

Paice, C. D. (1990). Another stemmer. ACM SIGIR Forum, ACM.

Paice, C. D. (1994). An evaluation method for stemming algorithm, Proceedings of ACM-SIGIR94.

Palmer, D. D. and D. S. Day (1997). A Statistical Profile of the Named Entity Task. Proceedings of Fifth ACL Conference for Applied Natural Language Processing (ANLP-97), Washington.

Patrick, J., C. Whitelaw, et al. (2002). Slinerc: The sydney language-independent named entity recogniser and classifier. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Peñas, A., Á. Rodrigo, et al. (2007). Overview of the answer validation exercise 2006. Evaluation of Multilingual and Multi-modal Information Retrieval, Springer**:** 257-264.

Peñas, A., Á. Rodrigo, et al. (2008). Overview of the answer validation exercise 2007. Advances in Multilingual and Multimodal Information Retrieval, Springer**:** 237-248.

Peng, F., N. Ahmed, et al. (2007). "Context sensitive stemming for web search." Proceedings of the 3oth annual international ACM SIGIR conference on Research and development in information retrieval: 639-646.

Pérez, A. and A. C. Fernández (2009). ERAFPLN. V 1.0. Matanzas, Universidad de Matanzas**:** Generador de Expresiones Regulares mediante Autómatas.

Pérez, A., A. C. Fernández, et al. (2011). Generación de expresiones regulares para la creación de reglas en aplicaciones de PLN. XII Simposio Internacional de Comunicación Social, Santiago de Cuba, Centro De Lingüística Aplicada.

Pérez Martínez, A., A. C. Fernández Orquín, et al. (2011). Generación de expresiones regulares para la creación de reglas en aplicaciones de PLN. XII SIMPOSIO INTERNACIONAL DE COMUNICACIÓN SOCIAL. CENTRO DE LINGUISTICA APLICADA SANTIAGO DE CUBA, Santiago de Cuba, Cuba.

Pita Fernández, S. and S. Pértega Díaz (2001). "Estadística descriptiva de los datos." Unidad de Epidemiología Clínica y Bioestadística. Complexo Hospitalario Juan Canalejo. A Coruña.

Poibeau, T. (2004). Automatic extraction of paraphrastic phrases from medium size corpora. Proceedings of the 20th international conference on Computational Linguistics, Association for Computational Linguistics.

Popovic, M. F. and P. Willet (1992). "The effectiveness of stemming for natural language accesess to Slovene textual data." Journal of American Society for Information Science **3 (5)**: 384-390.

Porter, M. (1980) "An Algorithm for suffix stripping. Program 14 (3)." 130-137.

Progovac, L. (1993). "Negative polarity: Entailment and binding." Linguistics and Philosophy **16**(2): 149-180.

Qiu, L., M.-Y. Kan, et al. (2006). Paraphrase recognition via dissimilarity significance classification. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Ratnaparki, A., A. Ittycheriah, et al. (2001). Question answering using maximum entropy components. of the NAACL Conference, Pittsburgh.

Rau, L. F. (1991). Extracting company names from text. Proceedings of the Seventh Conference on Artificial Intelligence Applications, IEEE.

Resnik, P. (1995). "Using information content to evaluate semantic similarity in a taxonomy." arXiv preprint cmp-lg/9511007.

Roche, E. and Y. Schabes (1997). Finite-state language processing, The MIT Press.

Rodrigo, Á., A. Peñas, et al. (2009). Overview of the answer validation exercise 2008. Evaluating Systems for Multilingual and Multimodal Information Access, Springer**:** 296-313.

Rodríguez Hontoria, H. (2001). Extracción de Información Tutorial. Sevilla, TALP Research Center Dep. Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya.

Saggion, H. and T. Poibeau (2013). Automatic text summarization: Past, present and future. Multi-source, Multilingual Information Extraction and Summarization, Springer**:** 3-21.

Sang, E. F. T. K. and F. D. Meulder. (2003, octubre 27, 2003). "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition." 2005, from http://www.cnts.ua.ac.be/conll2003.

Sang, T. K. and F. Erik (2002). Memory-based named entity recognition. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Santos, D., N. Seco, et al. (2006). Harem: An advanced NER evaluation contest for Portuguese. Proceedings of LREC.

Sanz, M. P., R. M. Guillena, et al. (2004). Extracción de Información en bibliotecas digitales. Alicante.

Savoy, J. (1999). "A stemming procedure and stopword list for general French corpora." Journal of American Society for Information Science **50 (10)**: 944-952.

Schmerling, S. F. (1971). "A note on negative polarity."

Sekine, S. (1998). NYU: Description of the Japanese NE system used for MET-2. Proceedings of the Seventh Message Understanding Conference (MUC-7).

Sekine, S. and H. Isahara (2000). IREX: IR and IE Evaluation project in Japanese. Proceedings of the 2nd International Conference on Language Resources and Evaluation.

Serra Sepúlveda, S. (2009). "Las restricciones de selección en los diccionarios generales de lengua española." Boletín de filología **44**: 187-213.

Shannon, C. E. (2001). "A mathematical theory of communication." ACM SIGMOBILE Mobile Computing and Communications Review **5**(1): 3-55.

Sheridan, P. and J. P. Ballerini (1996). Experiment in Multilingual Information Retrieval. ACM SIGIR Conference on Research and Development in Informational Retrieval, Zurich, Switzerland.

Shinyama, Y. (2005) "Using Repeated Patterns across Comparable Articles for Paraphrase Acquisition." 8.

Shinyama, Y. and S. Sekine (2005). "Using Repeated Patterns across Comparable Articles for Paraphrase Acquisition." New York University. Proteus Technical Report.

Shinyama, Y., S. Sekine, et al. (2002). Automatic paraphrase acquisition from news articles. Proceedings of the second international conference on Human Language Technology Research, Morgan Kaufmann Publishers Inc.

Smirnov, I. (2008). Overview of Stemming Algorithms, DePaul University.

Smith, T. F. and M. S. Waterman (1981). "Comparison of biosequences." Advances in Applied Mathematics **2**(4): 482-489.

Sokal, R. R. and C. D. Michener (1958). A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bull. **V.38:** p. 1409-1438.

Srihari, R., C. Niu, et al. (2000). A hybrid approach for named entity and sub-type tagging. Proceedings of the sixth conference on Applied natural language processing, Association for Computational Linguistics.

St-Onge, D. and U. o. T. C. S. R. Institute (1995). Detecting and correcting malapropisms with lexical chains, Citeseer.

Stein, B. and M. Potthast (2007). Putting successor variety stemming to work. Advances in Data Analysis, Springer**:** 367-374.

Suárez, A. (2004). Resolución de la ambigüedad semántica de las plabras mediante Modelos de Probabilidad de Máxima Entropía. Departamento de Lenguajes y Sistemas Informáticos. Alicante, Universidad de Alicante. **Tesis Doctoral:** 230.

Suárez, A. (2005). "Resolución de la Ambigüedad Semántica de las palabras mediante Modelos de Probabilidad de Máxima Entropía." Procesamiento de Lenguaje Natural **34**.

Sundheim, B. M. (1992). Overview of the fourth message understanding evaluation and conference. Proceedings of the 4th conference on Message understanding, Association for Computational Linguistics.

Tatu, M., B. Iles, et al. (2006). COGEX at the Second Recognizing Textual Entailment Challenge. Proceedings of the Second PASCAL Recognising Textual Entailment Challenge Workshop, Venice, Italy.

Tatu, M. and D. Moldovan (2007). Cogex at RTE3. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, Association for Computational Linguistics.

Thelwall, M. and K. Buckley (2013). "Topic-based sentiment analysis for the social web: The role of mood and issue-related words." Journal of the American Society for Information Science and Technology.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Tjong Kim Sang, E. F. and F. De Meulder (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics.

Toral, A. (2005). DRAMNERI: a free knowledge based tool to Named Entity Recognition. Proceedings of the 1st Free Software Technologies Conference, Citeseer.

Toral, A. and R. Muñoz (2006). "A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia." NEW TEXT Wikis and blogs and other dynamic text sources: 56.

Toribio, R., P. Martínez, et al. (2010). "Evaluación de la Extracción de Entidades Nombradas de OpenCalais en castellano." Procesamiento del lenguaje natural **45**: 287-290.

Troyano, J. A. (2004). "Named Entity Recognition through Corpus Transformation and System Combination."

Troyano Jiménez, J. A., V. J. Díaz Madrigal, et al. (2003). "Identificación de entidades con nombre basada en modelos de Markov y árboles de decisión." Procesamiento del lenguaje natural, nº 31 (septiembre 2003); pp. 235-242.

Tsukamoto, K., Y. Mitsuishi, et al. (2002). Learning with multiple stacking for named entity recognition. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Tsuruoka, Y. and J. i. Tsujii (2004). "Improving the performance of dictionary-based approaches in protein name recognition." Journal of biomedical informatics **37**(6): 461-470.

Valdivia, M. and M. Teresa (2004). Algoritmo LVQ aplicado a tareas de procesamiento del lenguaje natural, PhD thesis, Universidad de Málaga.

Valitutti, A., C. Strapparava, et al., Eds. (2004). Developing Affective Lexical Resources. ITC-irst, Trento, Italy, PsychNology Journal.

Vázquez, S., A. Montoyo, et al. (2004). Using Relevant Domains Resource for Word Sense Disambiguation. IC-AI'04. Proceedings of the International Conference on Artificial Intelligence, Ed: CSREA Press. Las Vegas, E.E.U.U.

Venegas, R. (2006). "La similitud léxico-semántica en artículos de investigación científica en español: Una aproximación desde el Análisis Semántico Latente." Revista signos **39**(60): 75-106.

Voorhees, E. (2001, Marzo 8, 2005). "SAIC Information Extraction." 2005.

Vossen, P. (1997). EuroWordNet: a multilingual database for information retrieval. Proceedings of the DELOS workshop on Cross-language Information Retrieval.

Wagner, R. A. and M. J. Fisher (1974). "The string-to-string correction problem." Journal of the ACM **Vol. 21**: pp. 168-173.

Wang, S. and C. D. Manning (2012). Baselines and bigrams: Simple, good sentiment and topic classification. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, Association for Computational Linguistics.

Wang, Y. and I. H. Witten (1997). Inducing model trees for continuous classes. Poster Papers of the 9th European Conference on Machine Learning (ECML 97), Prague, Czech Republic.

Watanabe, Y., M. Asahara, et al. (2007). A graph-based approach to named entity categorization in Wikipedia using conditional random fields. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

Watanabe, Y., Y. Miyao, et al. (2013). Overview of the Recognizing Inference in Text (RITE-2) at NTCIR-10. Proceedings of the 10th NTCIR Conference.

Wilbur, W. J. (2007). Unsupervised Learning of the Morpho-Semantic Relationship in MEDLINE. Biological, translational, and clinical language processing. BioNLP 2007, Prague.

Wimalasuriya, D. C. and D. Dou (2010). "Ontology-based information extraction: An introduction and a survey of current approaches." Journal of Information Science **36**(3): 306-323.

Winkler, W. (1999) "The state of record linkage and current research problems."

Witten, I. H. and E. Frank (2005). Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann.

Wu, D., G. Ngai, et al. (2002). Boosting for named entity recognition. proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics.

Wu, Z. and M. Palmer (1994). Verbs semantics and lexical selection. Proceedings of the 32nd annual meeting on Association for Computational Linguistics, Association for Computational Linguistics.

Yamamoto, K. (2002) "Acquisition of Lexical Paraphrases from Texts." 7.

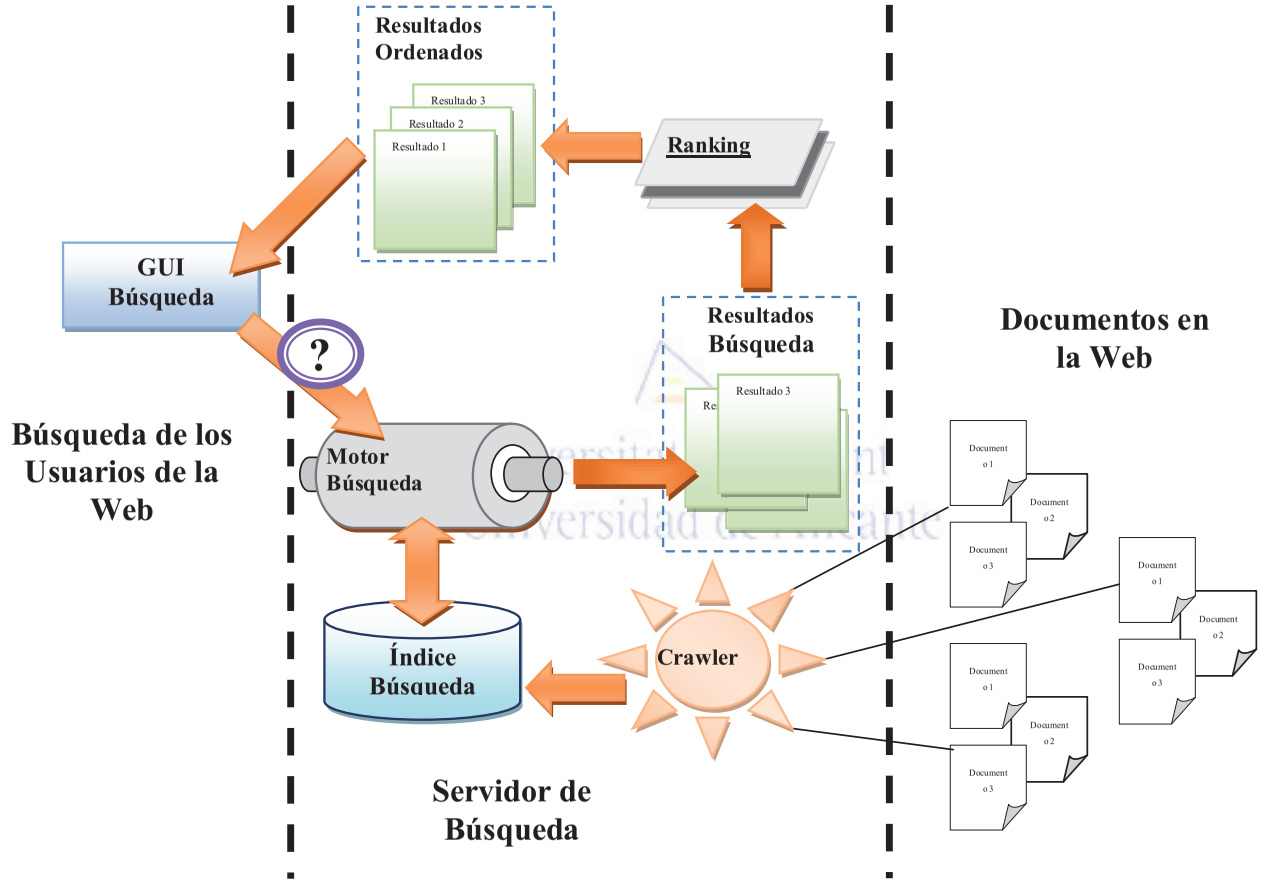
Zhang, B., A. Marin, et al. (2013). "Learning Phrase Patterns for Text Classification." IEEE Transactions on Audio, Speech & Language Processing **21**(6): 1180-1189.

Zhang, Y. and J. Patrick (2005). Paraphrase identification by text canonicalization. Proceedings of the Australasian language technology workshop.

Zheng, S. and J. Yu (2012). Automatic Summarization of Web Page Based on Statistics and Structure. Knowledge Discovery and Data Mining, Springer**:** 643-649.

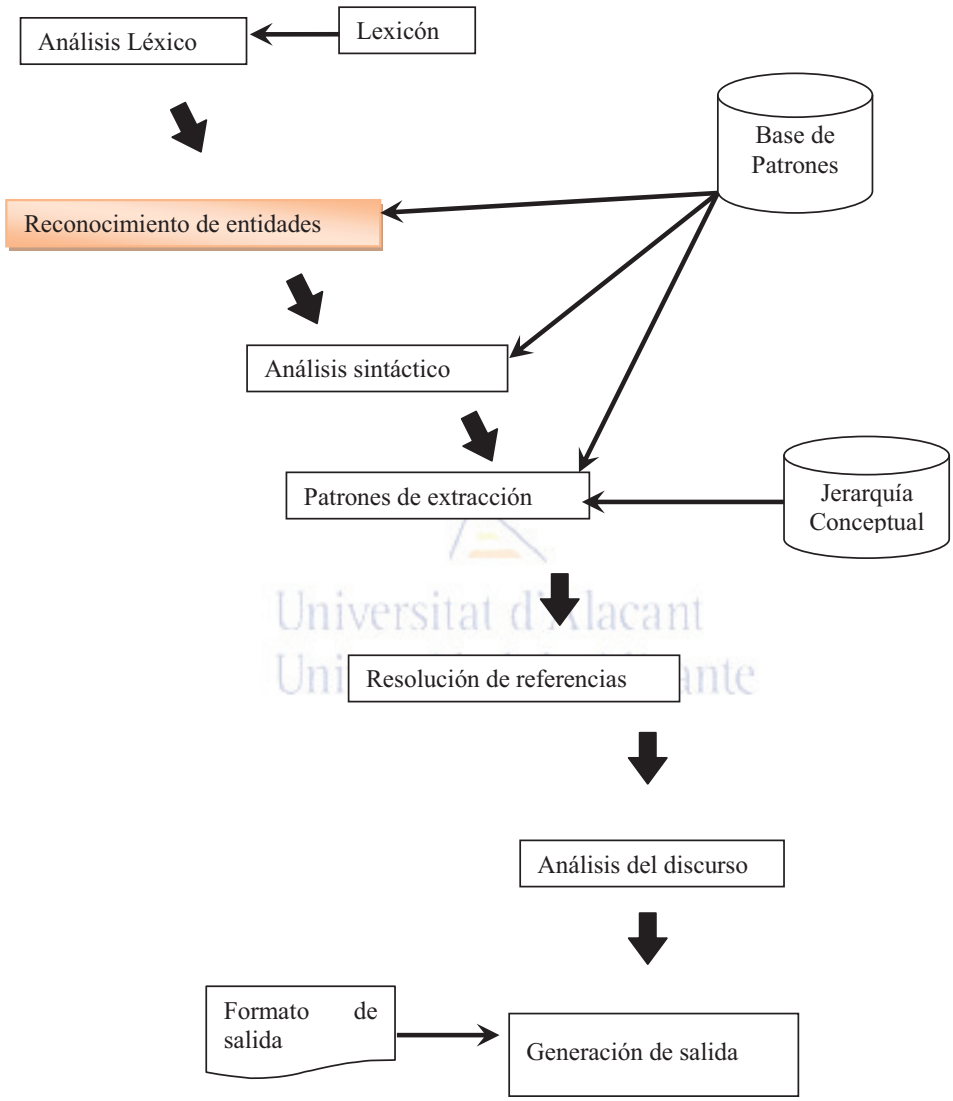
Zubaryeva, O. and J. Savoy (2010). Opinion Detection by Combining Machine Learning & Linguistic Tools Proceedings of NTCIR-8 Workshop Meeting, Tokyo, Japan.

ANEXO 1. Esquema de general de un sistema de RI.

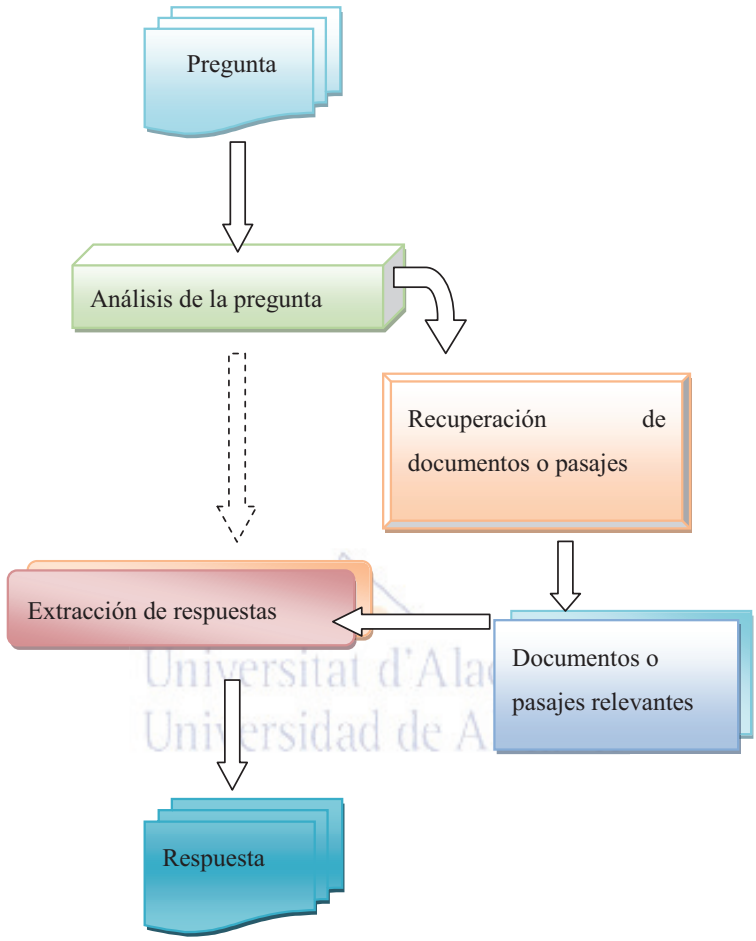


ANEXO 2. Estructura general de un sistema de EI.

(idea extraída de (Rodríguez Hontoria, 2001)).



ANEXO 3. Arquitectura básica de un sistema de BR.



ANEXO 4. Patrones a detectar. (Frase: objetivos educativos).

Patrones de búsqueda		
educativo	objetivos educativo	objetivos general educativo
educativo:	objetivos educativo:	objetivos general educativo:
educativo.	objetivos educativo.	objetivos general educativo.
educativos	objetivos educativos	objetivos general educativos
educativos:	objetivos educativos:	objetivos general educativos:
educativos.	objetivos educativos.	objetivos general educativos.
generales educativo	objetivo educativo	objetivo generales educativo
generales educativo:	objetivo educativo:	objetivo generales educativo:
generales educativo.	objetivo educativo.	objetivo generales educativo.
generales educativos	objetivo educativos	objetivo generales educativos
generales educativos:	objetivo educativos:	objetivo generales educativos:
generales educativos.	objetivo educativos.	objetivo generales educativos.
general educativo	objetivos generales educativo	objetivo general educativo
general educativo:	objetivos generales educativo:	objetivo general educativo:
general educativo.	objetivos generales educativo.	objetivo general educativo.
general educativos	objetivos generales educativos	objetivo general educativos
general educativos:	objetivos generales educativos:	objetivo general educativos:
general educativos.	objetivos generales educativos.	objetivo general educativos.

ANEXO 5. Código para emular la ER de la Tabla 1.1.

```
for ($i = 0; $i < count($Programa _Disciplina); $i++)
{$linea=trim($Programa _Disciplina[$i]); $linea_mayuscula= strtoupper($linea);
switch ($linea_mayuscula) {
   case "<P>EDUCATIVO</P>":
   case "<P>EDUCATIVO:</P>":
   case "<P>EDUCATIVO.</P>":
   case "<P>EDUCATIVOS</P>":
   case "<P>EDUCATIVOS:</P>":
   case "<P>EDUCATIVOS.</P>":
   case "<P>GENERALES EDUCATIVO</P>":
   case "<P>GENERALES EDUCATIVO:</P>":
   case "<P>GENERALES EDUCATIVO.</P>":
   case "<P>GENERALES EDUCATIVOS</P>":
   case "<P>GENERALES EDUCATIVOS:</P>":
   case "<P>GENERALES EDUCATIVOS.</P>":
   case "<P>GENERAL EDUCATIVO</P>":
   case "<P>GENERAL EDUCATIVO:</P>":
   case "<P>GENERAL EDUCATIVO.</P>":
   case "<P>GENERAL EDUCATIVOS</P>":
   case "<P>GENERAL EDUCATIVOS:</P>":
   case "<P>GENERAL EDUCATIVOS.</P>":
   case "<P>OBJETIVOS EDUCATIVO</P>":
   case "<P>OBJETIVOS EDUCATIVO:</P>":
   case "<P>OBJETIVOS EDUCATIVO.</P>":
   case "<P>OBJETIVOS EDUCATIVOS</P>":
   case "<P>OBJETIVOS EDUCATIVOS:</P>":
   case "<P>OBJETIVOS EDUCATIVOS.</P>":
   case "<P>OBJETIVO EDUCATIVO</P>":
   case "<P>OBJETIVO EDUCATIVO:</P>":
   case "<P>OBJETIVO EDUCATIVO.</P>":
   case "<P>OBJETIVO EDUCATIVOS</P>":
   case "<P>OBJETIVO EDUCATIVOS:</P>":
   case "<P>OBJETIVO EDUCATIVOS.</P>":
   case "<P>OBJETIVOS GENERALES EDUCATIVO</P>":
   case "<P>OBJETIVOS GENERALES EDUCATIVO:</P>":
   case "<P>OBJETIVOS GENERALES EDUCATIVO.</P>":
   case "<P>OBJETIVOS GENERALES EDUCATIVOS</P>":
   case "<P>OBJETIVOS GENERALES EDUCATIVOS:</P>":
   case "<P>OBJETIVOS GENERALES EDUCATIVOS.</P>":
   case "<P>OBJETIVOS GENERAL EDUCATIVO</P>":
   case "<P>OBJETIVOS GENERAL EDUCATIVO:</P>":
   case "<P>OBJETIVOS GENERAL EDUCATIVO.</P>":
   case "<P>OBJETIVOS GENERAL EDUCATIVOS</P>":
   case "<P>OBJETIVOS GENERAL EDUCATIVOS:</P>":
   case "<P>OBJETIVOS GENERAL EDUCATIVOS.</P>":
   case "<P>OBJETIVO GENERALES EDUCATIVO</P>":
   case "<P>OBJETIVO GENERALES EDUCATIVO:</P>":
   case "<P>OBJETIVO GENERALES EDUCATIVO.</P>":
   case "<P>OBJETIVO GENERALES EDUCATIVOS</P>":
   case "<P>OBJETIVO GENERALES EDUCATIVOS:</P>":
   case "<P>OBJETIVO GENERALES EDUCATIVOS.</P>":
   case "<P>OBJETIVO GENERAL EDUCATIVO</P>":
   case "<P>OBJETIVO GENERAL EDUCATIVO:</P>":
   case "<P>OBJETIVO GENERAL EDUCATIVO.</P>":
   case "<P>OBJETIVO GENERAL EDUCATIVOS</P>":
   case "<P>OBJETIVO GENERAL EDUCATIVOS:</P>":
   case "<P>OBJETIVO GENERAL EDUCATIVOS.</P>":
$file_ClearTable[$i]="<p><Obj_Educ>".substr($linea,3,strlen($linea)-7)."</p>";}
}
```

ANEXO 6. Porción de una ER para detectar expresiones temporales

```
((((hoy|ahora|anteayer|mañana|anoche|anteanoche|pasado mañana|antes de ayer|antes de anoche|al
mediodía|por la noche|hoy en día|hoy
día)\s|(en)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciem
bre)\s+)|(el|la|los|las)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|sep
tiembre|octubre|noviembre|diciembre)\s+)|(el|la|los|las)\s+(lunes|martes|miércoles|jueves|viernes|sá
bado|domingo)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|oct
ubre|noviembre|diciembre)\s+)a\s+(el|la|los|las)\s+(1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|
20|21|22|23|24|1h|2h|3h|4h|5h|6h|7h|8h|9h|10h|11h|12h|13h|14h|15h|16h|17h|18h|19h|20h|21h|22h|23h|24
h|1 hora|2 horas|3 horas|4 horas|5 horas|6 horas|7 horas|8 horas|9 horas|10 horas|11 horas|12
horas|13 horas|14 horas|15 horas|16 horas|17 horas|18 horas|19 horas|20 horas|21 horas|22 horas|23
horas|24
horas|1pm|2pm|3pm|4pm|5pm|6pm|7pm|8pm|9pm|10pm|11pm|12pm|13pm|14pm|15pm|16pm|17pm|18pm|19pm|20pm|21p
m|22pm|23pm|24pm|1am|2am|3am|4am|5am|6am|7am|8am|9am|10am|11am|12am|13am|14am|15am|16am|17am|18am|19
am|20am|21am|22am|23am|24am)\s+(del|de)\s+(el|la|los|las)\s+(mañana|tarde|noche|mediodía|medianoche|
madrugada|momento|período|actualidad|temporada|actualmente)|((((((((((\d\d?\s+(del|de|en|por)\s+(ener
o|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre)|((\d\d\d\d\s+\
d\d?\s+\d\d?)|(\d\d\d\d\d\d?\d\d?))))|(\d\d\s+(del|de|en|por)\s+\d+))|(\d\d\d\d(/|-)\d\d?(/|-
)\d\d?))|(este|esta)\s+(mañana|tarde|noche|mediodía|medianoche|madrugada|momento|período|actualidad|
temporada|actualmente))|(este|esta)\s+(años|año|Año|Años|días|día|Días|Día|meses|mes|Meses|Mes))|(es
te|esta)\s+(lunes|martes|miércoles|jueves|viernes|sábado|domingo))|(en)\s+(enero|febrero|marzo|abril
|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre)\s+(del|de)\s+\d+)|(el|la|los|las)\s
+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|
diciembre)\s+(del|de)\s+\d+)|(el|la|los|las)\s+(lunes|martes|miércoles|jueves|viernes|sábado|domingo
)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviemb
re|diciembre)\s+(del|de)\s+\d+))|((((hoy|ahora|anteayer|mañana|anoche|anteanoche|pasado mañana|antes
de ayer|antes de anoche|al mediodía|por la noche|hoy en día|hoy
día)\s|(en)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciem
bre)\s+)|(el|la|los|las)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|sep
tiembre|octubre|noviembre|diciembre)\s+)|(el|la|los|las)\s+(lunes|martes|miércoles|jueves|viernes|sá
bado|domingo)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|oct
ubre|noviembre|diciembre)\s+)a\s+(el|la|los|las)\s+(1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|
20|21|22|23|24|1h|2h|3h|4h|5h|6h|7h|8h|9h|10h|11h|12h|13h|14h|15h|16h|17h|18h|19h|20h|21h|22h|23h|24
h|1 hora|2 horas|3 horas|4 horas|5 horas|6 horas|7 horas|8 horas|9 horas|10 horas|11 horas|12
horas|13 horas|14 horas|15 horas|16 horas|17 horas|18 horas|19 horas|20 horas|21 horas|22 horas|23
horas|24
horas|1pm|2pm|3pm|4pm|5pm|6pm|7pm|8pm|9pm|10pm|11pm|12pm|13pm|14pm|15pm|16pm|17pm|18pm|19pm|20pm|21p
m|22pm|23pm|24pm|1am|2am|3am|4am|5am|6am|7am|8am|9am|10am|11am|12am|13am|14am|15am|16am|17am|18am|19
am|20am|21am|22am|23am|24am)\s+(del|de)\s+(hoy|ahora|anteayer|mañana|anoche|anteanoche|pasado
mañana|antes de ayer|antes de anoche|al mediodía|por la noche|por la tarde|por la mañana|en la
tarde|en la mañana|en la noche|hoy en día|hoy
día))|((((hoy|ahora|anteayer|mañana|anoche|anteanoche|pasado mañana|antes de ayer|antes de anoche|al
mediodía|por la noche|hoy en día|hoy
día)\s|(en)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciem
bre)\s+)|(el|la|los|las)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|sep
tiembre|octubre|noviembre|diciembre)\s+)|(el|la|los|las)\s+(lunes|martes|miércoles|jueves|viernes|sá
bado|domingo)\s+\d\d?\s+(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|oct
ubre|noviembre|diciembre)\s+)a\s+(el|la|los|las)\s+(1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|
20|21|22|23|24|1h|2h|3h|4h|5h|6h|7h|8h|9h|10h|11h|12h|13h|14h|15h|16h|17h|18h|19h|20h|21h|22h|23h|24
h|1 hora|2 horas|3 horas|4 horas|5 horas|6 horas|7 horas|8 horas|9 horas|10 horas|11 horas|12
horas|13 horas|14 horas|15 horas|16 horas|17 horas|18 horas|19 horas|20 horas|21 horas|22 horas|23
horas|24
horas|1pm|2pm|3pm|4pm|5pm|6pm|7pm|8pm|9pm|10pm|11pm|12pm|13pm|14pm|15pm|16pm|17pm|18pm|19pm|20pm|21p
m|22pm|23pm|24pm|1am|2am|3am|4am|5am|6am|7am|8am|9am|10am|11am|12am|13am|14am|15am|16am|17am|18am|19
am|20am|21am|22am|23am|24am)\s+(del|de)\s+(mañana|tarde|noche|mediodía|medianoche|madrugada|momento|
período|actualidad|temporada|actualmente))|(el|la|los|las)\s+año\s+\d+)|(((el|la|los|las)\s+\d\d?\s+
(del|de)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre
)|(en)\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre))
|(el|la|los|las)\s+(lunes|martes|miércoles|jueves|viernes|sábado|domingo)\s+\d\d?\s+(del|de)\s+(ener
o|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre)))
```

ANEXO 7. Atributos para entrenamiento con ME

Atributo	Descripción
A01	Contiene la entidad.
A02	(Evidencia externa) si la 1ª palabra hacia la izquierda es disparador de persona.
A03	(Evidencia externa) si la 2ª palabra hacia la izquierda es disparador de persona.
A04	(Evidencia externa) si la 3ª palabra hacia la izquierda es disparador de persona.
A05	(Evidencia externa) si la 1ª palabra hacia la derecha es disparador de persona.
A06	(Evidencia externa) si la 2ª palabra hacia la derecha es disparador de persona.
A07	(Evidencia externa) si la 3ª palabra hacia la derecha es disparador de persona.
A08	(Evidencia externa) si la 1ª palabra hacia la izquierda es disparador de localidad.
A09	(Evidencia externa) si la 2ª palabra hacia la izquierda es disparador de localidad.
A10	(Evidencia externa) si la 3ª palabra hacia la izquierda es disparador de localidad.
A11	No está en uso
A12	No está en uso
A13	1ª palabra hacia la izquierda de la entidad.
A14	2ª palabra hacia la izquierda de la entidad.
A15	3ª palabra hacia la izquierda de la entidad.
A16	1ª palabra hacia la derecha de la entidad.
A17	2ª palabra hacia la derecha de la entidad.
A18	3ª palabra hacia la derecha de la entidad.
A19	Posición de la no entidad.
A20	Si una parte de la entidad aparece en el diccionario de no entidades.
A21	Si toda la entidad aparece en el diccionario de nombres de personas.
A22	Si una parte de la entidad aparece en el diccionario de nombres de personas
A23	(Evidencia Interna) Si la entidad contiene un disparador de persona.
A24	Si toda la entidad aparece en el diccionario de localidades.
A25	Si una parte de la entidad aparece en el diccionario de localidades.
A26	(Evidencia Interna) Si la entidad contiene un disparador de localidad.
A27	Si toda la entidad aparece en el diccionario de organizaciones.
A28	Si una parte de la entidad aparece en el diccionario de organizaciones.
A29	(Evidencia Interna) Si la entidad contiene un disparador de organización.
A30	Si toda la entidad aparece en el diccionario de Miscelánea
A31	Si una parte de la entidad aparece en el diccionario de Miscelánea
A32	(Evidencia Interna) si la entidad contiene un disparador de Miscelánea

A33	(Evidencia externa) si la 1ª palabra hacia la derecha es disparador de localidad.
A34	(Evidencia externa) si la 2ª palabra hacia la derecha es disparador de localidad.
A35	(Evidencia externa) si la 3ª palabra hacia la derecha es disparador de localidad
A36	(Evidencia externa) si la 1ª palabra hacia la izquierda es disparador de organización.
A37	(Evidencia externa) si la 2ª palabra hacia la izquierda es disparador de organización.
A38	(Evidencia externa) si la 3ª palabra hacia la izquierda es disparador de organización.
A39	(Evidencia externa) si la 1ª palabra hacia la derecha es disparador de organización.
A40	(Evidencia externa) si la 2ª palabra hacia la derecha es disparador de organización.
A41	(Evidencia externa) si la 3ª palabra hacia la derecha es disparador de organización.

ANEXO 8. Nueva combinación de atributos.

Nuevo Atrib.	Propuesta de cambios	Atrib.	Descripción
A1	Permanece	A01	Contiene la entidad.
A2	Agrupar	A02	(Evidencia externa) si la 1ª palabra hacia la izquierda es disparador de persona.
		A03	(Evidencia externa) si la 2ª palabra hacia la izquierda es disparador de persona.
	Eliminar	A04	(Evidencia externa) si la 3ª palabra hacia la izquierda es disparador de persona.
A3	Agrupar	A05	(Evidencia externa) si la 1ª palabra hacia la derecha es disparador de persona.
		A06	(Evidencia externa) si la 2ª palabra hacia la derecha es disparador de persona.
	Eliminar	A07	(Evidencia externa) si la 3ª palabra hacia la derecha es disparador de persona.
A4	Agrupar	A08	(Evidencia externa) si la 1ª palabra hacia la izquierda es disparador de localidad.
		A09	(Evidencia externa) si la 2ª palabra hacia la izquierda es disparador de localidad.
	Eliminar	A10	(Evidencia externa) si la 3ª palabra hacia la izquierda es disparador de localidad.
	Eliminar	A11	No está en uso
	Eliminar	A12	No está en uso
A5	Permanece	A13	1ª palabra hacia la izquierda de la entidad.
A6	Permanece	A14	2ª palabra hacia la izquierda de la entidad.
A7	Permanece	A15	3ª palabra hacia la izquierda de la entidad.
A8	Permanece	A16	1ª palabra hacia la derecha de la entidad.
A9	Permanece	A17	2ª palabra hacia la derecha de la entidad.
A10	Permanece	A18	3ª palabra hacia la derecha de la entidad.
	Eliminar	A19	Posición de la no entidad.
	Eliminar	A20	Si una parte de la entidad aparece en el diccionario de no entidades.
A11	Agrupar	A21	Si toda la entidad aparece en el diccionario de nombres de personas.
		A22	Si una parte de la entidad aparece en el diccionario de nombres de personas
		A23	(Evidencia Interna) Si la entidad contiene un disparador de persona.
A12	Agrupar	A24	Si toda la entidad aparece en el diccionario de localidades.
		A25	Si una parte de la entidad aparece en el diccionario de localidades.
		A26	(Evidencia Interna) Si la entidad contiene un disparador de localidad.
A13	Agrupar	A27	Si toda la entidad aparece en el diccionario de organizaciones.
		A28	Si una parte de la entidad aparece en el diccionario de organizaciones.
		A29	(Evidencia Interna) Si la entidad contiene un disparador de organización.
A14	Agrupar	A30	Si toda la entidad aparece en el diccionario de Miscelánea
		A31	Si una parte de la entidad aparece en el diccionario de Miscelánea
	Eliminar	A32	(Evidencia Interna) si la entidad contiene un disparador de Miscelánea
A15	Agrupar	A33	(Evidencia externa) si la 1ª palabra hacia la derecha es disparador de localidad.
		A34	(Evidencia externa) si la 2ª palabra hacia la derecha es disparador de localidad.
	Eliminar	A35	(Evidencia externa) si la 3ª palabra hacia la derecha es disparador de localidad
A16	Agrupar	A36	(Evidencia externa) si la 1ª palabra hacia la izquierda es disparador de organización.
		A37	(Evidencia externa) si la 2ª palabra hacia la izquierda es disparador de organización.
	Eliminar	A38	(Evidencia externa) si la 3ª palabra hacia la izquierda es disparador de organización.
A17	Agrupar	A39	(Evidencia externa) si la 1ª palabra hacia la derecha es disparador de organización.
		A40	(Evidencia externa) si la 2ª palabra hacia la derecha es disparador de organización.
	Eliminar	A41	(Evidencia externa) si la 3ª palabra hacia la derecha es disparador de organización.

ANEXO 9. Abreviaturas

affectDist= Distancia entre dominios del texto y la hipótesis ofrecida por el recurso WorNet Affect

BlockD= Block Distance

CLDeviation= Distancia de ChapmanLengthDeviation

CMLength= Distancia de ChapmanMeanLength.

CosineS= Cosine Similarity

Cost= Distancia total de macheo óptimo

CostA= Distancia total de macheo óptimo Adjetivos

CostN= Distancia total de macheo óptimo para los sustantivos

CostR= Distancia total de macheo óptimo para los Adverbios

CostV= Distancia total de macheo óptimo Verbos

CPA= Cantidad de Palabras Alineadas.

CPNA= Cantidad de Palabras no Alineadas.

DEx= Distancia Extendida

DiceS= Dice Similarity

Dif= Número de palabras que no se emparejan

DifA= Número de adjetivos que no se emparejan

DifN= Número de Sustantivos que no se emparejan

DifR= Número de advervios que no se emparejan

DifV= Número de verbos que no se emparejan

DistP= Distancia entre las polaridades del texto y la hipótesis

DLED= Distancia de Levenshtein Doble

DomainDist= Distancia entre dominios del texto y la hipótesis ofrecida por el recurso WND.

DomainHipot= Dominio de la hipótesis ofrecido por el recurso WND.

DomainText= Dominio del texto ofrecido por el recurso WND.

EuclideanD= Distancia de Euclideana

JaccardS= Jaccard Similarity

Jaro= Distancia de Jaro

JaroW= Distancia de JaroWinkler

LED= Distancia de Levenshtein

LevLema= Distancia de Levenshtein Lema.

MaCoef= Matching Coefficient

Min= Número de lemas de la frase más corta

MinA= Número de adjetivos de la frase más corta

MinN= Número de Sustantivos de la frase más corta

MinR= Número de advervios de la frase más corta

MinV= Número de lemas de la frase más corta

MongeElkan= Distancia de MongeElkan

NormAlinLevF= Normalización del Alineamiento por Levenshtein y la grafía de las palabras.

NormAlinLevL= Normalización del Alineamiento por Levenshtein y el Lema de las palabras.

NormDLED= Distancia Normalizada de Levenshtein Doble.

NormLED= Distancia de Levenshtein Normalizada.

NWunch= Distancia de NeedlemanWunch.

OverlapCoef= OverlapCoefficient.

PolHipótesis= Polaridad de la hipótesis.

PolTexto= Polaridad del texto.

QGramsD= QGrams Distance.

Same= Número de coincidencias exactas.

SameA= Número de coincidencias exactas de Adjetivos.

SameN= Número de coincidencias exactas Sustantivos.

SameR= Número de coincidencias exactas Advervios.

SameV= Número de coincidencias exactas Verbos.

scDist = Distancia entre dominios del texto y la hipótesis ofrecida por el recurso Semantic Class.

SemanticClassHipot= Dominio de la hipótesis ofrecido por el recurso SC.

SemanticClassText= Dominio del texto ofrecido por el recurso SC.

SIM= Similitud final del alineamiento.

sumoDist= Distancia entre dominios del texto y la hipótesis ofrecida por el recurso Sumo.

SumoHipot= Dominio de la hipótesis ofrecido por el recurso Sumo.

SumoText= Dominio del texto ofrecido por el recurso Sumo.

SWaterman= Distancia de SmithWaterman.

SWGotoh= Distancia de SmithWatermanGotoh.

SWGWAffine= Distancia de SmithWatermanGotohWindowedAffine.

WordNetDist= Distancia entre dominios del texto y la hipótesis ofrecida por el recurso WN.

ANEXO 10. Influencia de atributos (correlación de Pearson).

Tabla 4.1. Pearson para atributos con el conjunto de entrenamiento de SemEval 2012, variante de ALS.

Atributo	**Correlación**	**Atributo**	**Correlación**	**Correlación usando todos los atributos**
DEx	0.596	NWunch	0.2431	0.8519
QGramsD	0.5749	JaroW	0.2366	
CMLength	0.5588	WordNetDist	0.2098	
CPA	0.5753	CPNA	0.2588	
BlockD	0.5259	NormLED	0.4349	
SIM	0.5064	CLDeviation	0.4035	
OverlapCoef	0.4983	EuclideanD	0.4025	
MongeElkan	0.4971	Jaro	0.3611	
DiceS	0.4951	SWGWAffine	0.29103	
MaCoef	0.4913	SWGotoh	0.2893	
CosineS	0.4859	SWaterman	0.2803	
JaccardS	0.4849	NormDLED	0.4457	
DLED	0.4782	DistP	0.0342	
LED	0.4572			

ANEXO 11. Influencia de atributos (correlación de Pearson).

Tabla 4.2. C. de Pearson para atributos con el conjuntos de entrenamiento de SemEval 2012, variante de AS.

<table>
<tr><td>Feature</td><td colspan="9">Correlación de Pearson (MSRpar, MSRvid y SMTeuroparl)</td></tr>
<tr><td>sumoDist</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td rowspan="30">0.8509</td></tr>
<tr><td>affectDist</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>WordNetDist</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>scDist</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>DEx</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td rowspan="29">0.8507</td></tr>
<tr><td>DistP</td><td></td><td></td><td></td><td></td><td rowspan="2"></td><td></td><td></td></tr>
<tr><td>QGramsD</td><td></td><td></td><td></td><td></td><td></td><td rowspan="28">0.8491</td></tr>
<tr><td>CMLength</td><td></td><td></td><td rowspan="28">0.8075</td><td></td><td></td><td></td></tr>
<tr><td>Same</td><td rowspan="4">0.7043</td><td rowspan="28">0.795</td><td rowspan="28">0.829</td><td rowspan="28">0.8302</td><td rowspan="28">0.8228</td></tr>
<tr><td>Min</td></tr>
<tr><td>Dif</td></tr>
<tr><td>Cost</td></tr>
<tr><td>SameV</td><td rowspan="4">0.576</td></tr>
<tr><td>MinV</td></tr>
<tr><td>DifV</td></tr>
<tr><td>CostV</td></tr>
<tr><td>SameN</td><td rowspan="4">0.5975</td></tr>
<tr><td>MinN</td></tr>
<tr><td>DifN</td></tr>
<tr><td>CostN</td></tr>
<tr><td>SameA</td><td rowspan="4">0.4285</td></tr>
<tr><td>MinA</td></tr>
<tr><td>DifA</td></tr>
<tr><td>CostA</td></tr>
<tr><td>SameR</td><td rowspan="4">0.3778</td></tr>
<tr><td>MinR</td></tr>
<tr><td>DifR</td></tr>
<tr><td>CostR</td></tr>
</table>

Nota: las celdas en gris significan que los atributos no fueron tomados en consideración.

ANEXO 12. Comparativa entre diferentes stemmers.

stemmer	Ventajas	Desventajas
Lovins	1. Rápido, algoritmo de una sola pasada. 2. Maneja la eliminación de las letras dobles, ejemplo getting que se transforma en get. 3. Manejan plurales irregulares como mouse y mice, etc.	1. Consumidor de tiempo. 2. No disponible todos los sufijos. 3. No es muy confiable y con frecuencia deja de formar palabras a partir de tallos. 4. Dependiente del vocabulario técnico utilizado por el autor.
Porter	1. Produce la mejor salida comparado con otros. 2. Menos rango de errores. 3. Comparado con Lovins es un stemmer ligero. 4. El framework Snowball es independiente del lenguaje	1. Los stem producidos no siempre son palabras reales. 2. Tiene al menos 5 pasos y 6 reglas que consumen tiempo. 3. Puede eliminar terminaciones que coincidan con sufijos.
Paice/Husk	1 Forma simple 2 Cada iteración se ocupa del borrado y remplazo	1. Algoritmo pesado. 2. Puede ocurrir overstemming.
Dawson	1 Cubre más sufijos que Lovins 2 Más rápido	1 Algoritmo muy complejo 2 Carece de implementación estándar.
Krovetz	1 Es un stemmer ligero 2 Puede ser usado como pre-stemmer para otros stemmers.	1 No es eficiente para grandes documentos. 2 No puede cubrir palabras que no estén en el lexicón. 3 No produce buenos resultados de precisión y cobertura. 4 El lexicón debe ser creado con antelación.
N-Gram	1 Basado el concepto de n-grama y la comparación de cadenas. 2 Independiente del lenguaje.	1 No es eficiente en tiempo 2 Requiere de un espacio significativo para crear el índice de n-grmas. 3 No es un método práctico.
HMM	1 Basado en el concepto de las cadenas ocultas de Markov. 2 No supervisado e independiente del lenguaje.	1 Método complejo para su implementación. 2 Puede ocurrir overstemming.
Xerox	1 Trabaja bien en documentos grandes 2 Remueve los prefijos siempre que aplique. 3 Todos los stem son palabras válidas.	1 No puede cubrir palabras fuera del lexicón. 2 No se desempeña bien en lenguajes distintos del inglés. Puede ocurrir overstemming. 3 Dependiente del lenguaje.
YASS	1 Basado en cluster jerárquico distancias métricas. 2 Basado en corpus. 3 Puede ser usado en cualquier lenguaje si se conoce su morfología.	1 Dificultad para elegir el umbral para crear el cluster. 2 Requiere de poder computacional.

Nota: Tabla original en (Jivani, 2011)

ANEXO 13. Aparición de los términos Pragmática, Semántica y Sintaxis.

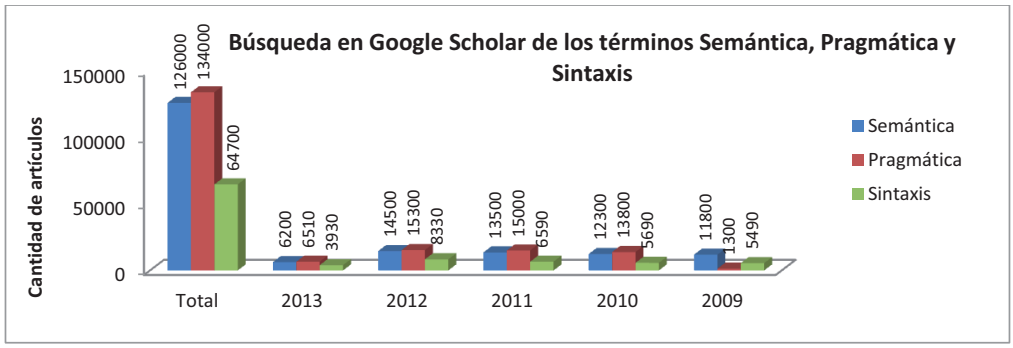


Figura 4.1. Búsqueda en Google Scholar de los términos: Semántica, Pragmática y Sintaxis.

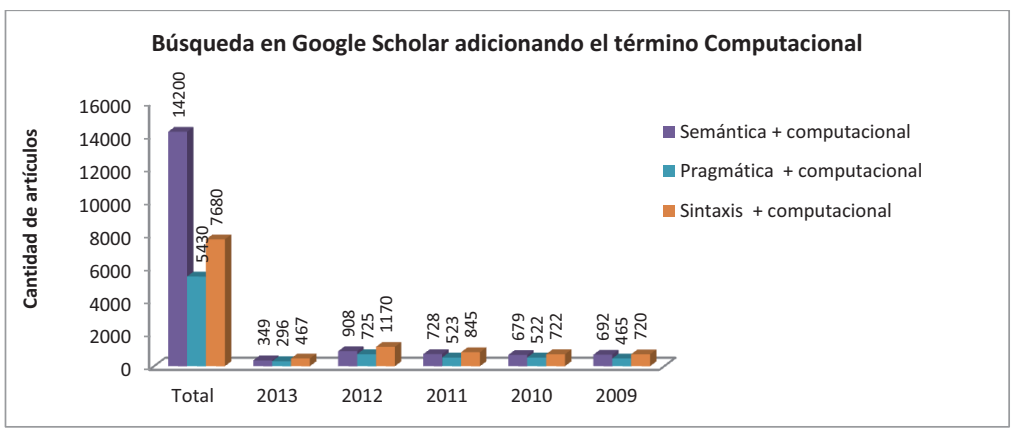


Figura 4.2. Búsqueda en Google Scholar de los términos: Semántica, Pragmática y Sintaxis, adicionando computacional.

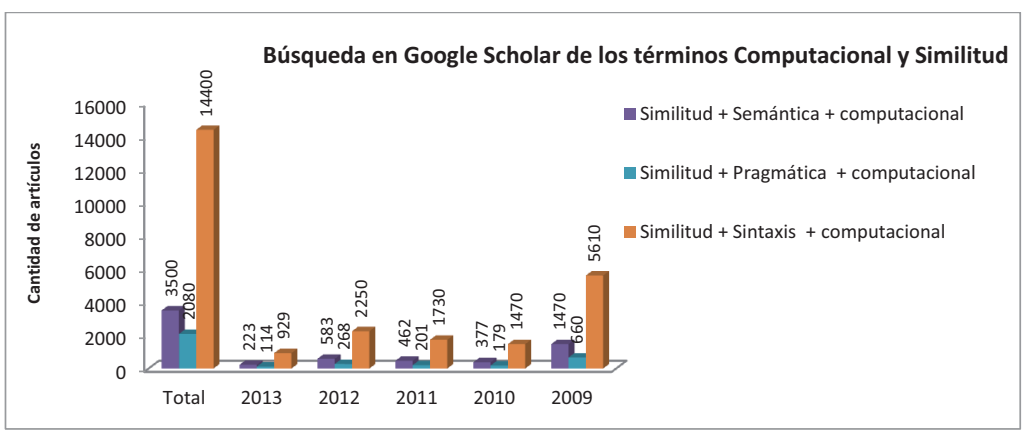


Figura 4.3. Búsqueda en Google Scholar de: Semántica, Pragmática y Sintaxis, adicionando computacional y similitud.