



Universidad
Carlos III de Madrid

Departamento de Ingeniería Telemática

Sistema de Análisis Automático de
Sentimientos Basado en Procesamiento
del Lenguaje Natural

**PROYECTO FIN DE CARRERA
INGENIERÍA DE TELECOMUNICACIÓN**

Autor: Alejandro Casas García
Tutor: Julio Villena Román

Leganés, Octubre de 2014

Título: Sistema de Análisis Automático de Sentimientos
Basado en Procesamiento del Lenguaje Natural

Autor: Alejandro Casas García

Tutor: Julio Villena Román

EL TRIBUNAL

Presidente: José Alberto Hernández Gutiérrez

Vocal: Julián Moreno Schneider

Secretario: Jesús Arias Fisteus

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 22 de Octubre de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

SECRETARIO

VOCAL

PRESIDENTE

Agradecimientos

Tengo la gran suerte de estar rodeado de grandísimas personas, que enriquecen mi vida y me apoyan en todo lo que hago, por disparatado que sea en ocasiones. Es gracias a todos ellos, que finalmente estoy aquí, cerrando otra etapa de mi vida en la que tantísimo he aprendido. Por ello quiero dar las gracias a todos ellos.

A mi familia, en especial a mi madre, por darlo todo por mí y ser el verdadero apoyo que me sujeta cuando todo se tuerce.

A mis músicos, que me enseñaron a vivir con pasión y a agarrar la vida con siete manos.

A mi peque por hacerme recuperar la ilusión, que andaba un poco pálida últimamente, y aventurarse a compartirlo todo conmigo.

A mis amigos, que siempre están ahí para compartir la vida, por mucho que nos separen los kilómetros.

A todo “piso-teleco”: Antonio, Fernando y David. Los mejores compañeros de piso que se puede tener, con los que he reído, aprendido y compartido mucho. Y con los que he librado mano a mano todas las “ramitas” de la carrera.

Y por último, a mis compañeros de trabajo, sin los que jamás hubiera sido posible este proyecto, por acogerme como a uno más desde el primer día, y a mi tutor, por darme la oportunidad de ver la carrera desde otro punto de vista y ayudarme en todo lo que estaba en su mano.

Gracias.

Resumen

En este Proyecto Fin de Carrera se pretende diseñar e implementar un sistema capaz de extraer información de tipo emocional y de opinión de un texto de forma automatizada a través de técnicas de procesamiento del lenguaje natural y análisis morfosintáctico del mismo. Así, se persigue dar una solución tecnológica al problema del **análisis automático del sentimiento**, cuyo interés ha ido en alza estos últimos años, debido mayormente al crecimiento continuado en nuestra sociedad de las redes sociales, en las que los usuarios vuelcan en forma de opinión: datos muy valiosos y al mismo tiempo, difíciles de extraer, debido, paradójicamente, a la cantidad y diversidad de la información disponible.

El sistema utilizará un diccionario externo de reglas, en el que se define, entre otras cosas, palabras con polaridad asociada (sentimiento positivo o negativo) que además pueden ser modificadas mediante diferentes operadores.

Por otro lado, se pondrá interés en detectar las entidades y conceptos que aparezcan en el texto, y en cómo éstos son afectados por los sentimientos descubiertos en él. Permitiendo de esta forma recopilar en los textos analizados información de mucha utilidad, la valoración de los usuarios sobre diferentes asuntos.

Además, al final de este documento se realiza una evaluación del sistema que nos permitirá descubrir hasta qué punto puede nuestro sistema extraer de forma correcta esta información y determinar posibles mejoras futuras.

Índice general

| | | |
|--------|---|----|
| 1. | Introducción | 1 |
| 1.1. | Motivación del proyecto | 1 |
| 1.2. | Objetivos del proyecto..... | 2 |
| 1.3. | Contenido de la memoria | 3 |
| 2. | Estado del arte | 5 |
| 2.1. | Historia, interés y actualidad del análisis automático de sentimientos..... | 5 |
| 2.2. | Problemática del análisis de sentimiento..... | 9 |
| 2.2.1. | Problemática del procesamiento del lenguaje natural | 9 |
| 2.2.2. | Problemática del análisis de sentimientos | 12 |
| 2.3. | Procesamiento del lenguaje natural..... | 14 |
| 2.3.1. | Técnicas del procesamiento del lenguaje natural..... | 14 |
| 3. | Diseño del sistema | 16 |
| 3.1. | Introducción..... | 16 |
| 3.2. | Input: Entrada del sistema..... | 17 |
| 3.3. | Output: Salida del sistema | 17 |
| 3.3.1. | score_tag | 19 |
| 3.3.2. | Agreement, Subjectivity, Irony..... | 20 |
| 3.3.3. | entity_list y concept_list..... | 21 |
| 3.3.4. | sentence_list..... | 21 |
| 3.3.5. | segment_list | 22 |
| 3.4. | POS, Parsing y Topic extraction | 22 |
| 3.5. | Diccionario de reglas..... | 25 |
| 3.5.1. | Definición de macros | 26 |
| 3.5.2. | Definición de reglas | 27 |
| 3.5.3. | Otros detalles de la implementación del diccionario..... | 34 |
| 3.5.4. | Anexo, ejemplo de reglas de diccionario | 34 |
| 3.6. | Búsqueda y etiquetado | 35 |
| 3.6.1. | Carga del diccionario | 36 |
| 3.6.2. | Búsqueda en el diccionario..... | 37 |

| | | |
|--------|--|----|
| 3.6.3. | Selección de reglas | 38 |
| 3.7. | Procesamiento | 40 |
| 3.7.1. | Introducción | 40 |
| 3.7.2. | Aplicación de negadores..... | 42 |
| 3.7.3. | Creación de segmentos | 44 |
| 4. | Implementación del sistema | 53 |
| 4.1. | Implementación del diccionario | 53 |
| 4.2. | Implementación del motor | 53 |
| 4.2.1. | Estructura del programa..... | 54 |
| 4.2.2. | Clases implementadas | 54 |
| 4.2.3. | Implementación..... | 57 |
| 5. | Evaluación | 61 |
| 5.1. | Introducción..... | 61 |
| 5.2. | Evaluación del sentimiento a nivel global | 61 |
| 5.2.1. | Corpus de pruebas..... | 61 |
| 5.2.2. | Contenido del corpus | 62 |
| 5.2.3. | Preparación del diccionario | 63 |
| 5.2.4. | Batería de pruebas | 63 |
| 5.2.5. | Resultados de la prueba | 64 |
| 5.3. | Evaluación del sentimiento a nivel de entidad..... | 68 |
| 5.4. | Análisis de respuestas y errores del sistema..... | 70 |
| 6. | Conclusiones y Trabajos futuros | 76 |
| 6.1. | Conclusiones | 76 |
| 6.2 | Trabajo futuro | 77 |
| 7. | Bibliografía | 80 |

Índice de ilustraciones

| | |
|--|----|
| Ilustración 1: Arquitectura del sistema | 16 |
| Ilustración 2: Separación por frases. | 23 |
| Ilustración 3: Segmentación en árbol..... | 23 |
| Ilustración 4: Lematización | 23 |
| Ilustración 5: Identificación de entidades y conceptos..... | 24 |
| Ilustración 6: Detección de relaciones | 24 |
| Ilustración 7: Salida del preprocesado | 25 |
| Ilustración 8: Trie | 36 |
| Ilustración 9: Carga del diccionario | 37 |
| Ilustración 10: Búsqueda de reglas..... | 38 |
| Ilustración 11: Selección por prioridad..... | 39 |
| Ilustración 12: Prioridad en elección de reglas | 39 |
| Ilustración 13: Información del preprocesado | 40 |
| Ilustración 14: Ejemplos de etiquetado..... | 41 |
| Ilustración 15: Negación sobre el árbol morfosintáctico | 43 |
| Ilustración 16: Negación de polaridades | 43 |
| Ilustración 17: Algoritmo creador de segmentos..... | 45 |
| Ilustración 18: Agregación de etiquetas..... | 47 |
| Ilustración 19: Relación entre Topics y Segmentos..... | 49 |
| Ilustración 20: Algoritmo de cruce | 50 |
| Ilustración 21: Ejemplo de segmento creado por cruce | 51 |
| Ilustración 22: Ejemplo de segmento creado por cruce con desacuerdo..... | 52 |
| Ilustración 23: Esquema de clases..... | 56 |
| Ilustración 24: Precisión del sistema | 64 |
| Ilustración 25: Resultados del TASS 2013..... | 67 |
| Ilustración 26: Matriz de confusión..... | 68 |

Índice de tablas

| | |
|--|----|
| Tabla 1: Ejemplos de uso del lenguaje en las redes sociales..... | 12 |
| Tabla 2: Campos de salida del sistema | 19 |
| Tabla 3: score_tag..... | 20 |
| Tabla 4: Atributos de una oración | 21 |
| Tabla 4: Atributos de un segmento | 22 |
| Tabla 5: Operadores modificadores | 29 |
| Tabla 6: Operadores Variables | 30 |
| Tabla 7: Etiquetas morfosintácticas | 32 |
| Tabla 8: Operadores de contexto | 33 |
| Tabla 9: Caracteres escapados | 34 |
| Tabla 10: Prioridad de los complementos | 46 |
| Tabla 11: Estructura de ficheros..... | 54 |
| Tabla 12: Resultados generales de la evaluación | 65 |
| Tabla 13: Clasificación de errores..... | 65 |
| Tabla 14: Resultados para el sistema de tres niveles | 66 |
| Tabla 15: Respuestas correctas | 71 |
| Tabla 16: Errores leves | 72 |
| Tabla 17: Falsos positivos y falsos negativos..... | 73 |
| Tabla 18: Cambios de polaridad | 74 |

Índice de ejemplos

| | |
|--|----|
| Ejemplo 1: Salida del programa..... | 18 |
| Ejemplo 2: Definición de una macro | 26 |
| Ejemplo 3: Creación de una macro..... | 26 |
| Ejemplo 4: Sintaxis general de una regla del diccionario | 27 |
| Ejemplo 5: Sintaxis del campo regla | 27 |
| Ejemplo 6: Agrupación de términos | 28 |
| Ejemplo 7: Etiquetado básico de sentimiento | 29 |
| Ejemplo 8: Uso del operador igualdad | 31 |
| Ejemplo 9: Uso del operador de categoría..... | 32 |
| Ejemplo 10: Uso de contextos | 34 |
| Ejemplo 11: Uso del diccionario | 35 |
| Ejemplo 12: Contenido del corpus | 62 |

1. Introducción

1.1. Motivación del proyecto

Las opiniones y sentimientos de las personas siempre han sido una valiosa fuente de información. Por ejemplo, los políticos necesitan conocer el impacto público que producen y mantener día a día controlada la opinión pública, que será el medidor número uno del éxito de sus campañas electorales. También en el mundo de los negocios, desde las empresas más pequeñas a las más grandes, se quiere saber qué opinan los usuarios de los productos y servicios que ofrecen, para así poder tomar las decisiones adecuadas que lleven su negocio hacia el éxito. Las empresas de marketing siempre han tomado la opinión social como el centro de sus actividades. Los estudios de mercado, tan valorados por las grandes empresas, no son otra cosa que una gran recolección de datos (de opinión entre otros), por medio de distintas herramientas, como entrevistadores, formularios y encuestas telefónicas, todo ello con el mismo fin, obtener una buena base de conocimiento sobre la opinión social en algún aspecto concreto.

Además de la importancia que de por sí tienen, actualmente, la sociedad informatizada en la que nos hemos adentrado en el último siglo, ha hecho que el interés por el análisis de la opinión crezca mucho en los últimos años. Prueba de ello son los estudios, charlas, software, investigaciones desarrolladas recientemente sobre el tema.

Vivimos en una sociedad en la que la evolución de los sistemas electrónicos e informáticos, junto con el continuo crecimiento de las redes de datos, ha supuesto un cambio enorme en nuestra forma de vida. Nos hemos rodeado de sistemas capaces de obtener, crear, transportar todo tipo de información y compartirla de forma inmediata con el resto de usuarios de la red. No cabe duda que la información se ha convertido pues, en el eje central de nuestra sociedad. Esta llamada “era digital” ha hecho posible el fluir de la información prácticamente sin límites a unas velocidades que jamás se hubieran imaginado hace apenas cien años y ha transformado profundamente las bases de nuestro mundo, cambiado de forma radical nuestra forma de trabajar, de divertirnos e incluso nuestra forma de relacionarnos con los demás.

Con el auge de las redes sociales en la última década, los usuarios de servicios web como Twitter, Facebook o Tumblr no paran de crecer [1], y en ellos los usuarios no paran de volcar y compartir, opiniones y sentimientos sobre cualquier tema, creando una gran cantidad de información valiosa y pública, pero difícilmente analizable.

Si antes el problema de las empresas de marketing era conseguir la información, ahora el problema es manejarla. La información ya está ahí, pero desordenada y en cantidades inmanejables. El reto ahora es encontrar entre toda esa cantidad de datos,

sólo lo que necesitamos, y hacerlo de una forma estructurada, que sea útil, que nos permita analizar los resultados de forma que éstos nos lleven a conclusiones beneficiosas.

Aquí es donde entra en juego el objetivo del presente proyecto. Las técnicas de análisis automático de sentimiento pueden ayudar mucho en este campo, permitiendo el análisis de grandes cantidades de datos de forma rápida, eficiente y con un alto índice de acierto. Por ello están despertando un enorme interés entre la comunidad científica.

Gracias a estas técnicas, se podría por ejemplo procesar un gran lote de tweets (pequeños textos publicados por la famosa red de microblogging Twitter), y nos permitiría conocer a “tiempo real” los sentimientos que se están volcando en los últimos momentos en la red. Pudiendo analizar y controlar, si un nuevo producto está generando rechazo, o está gustando, si un programa televisivo está siendo entretenido, o si un famoso deportista, está obteniendo el apoyo o genera rechazo a los internautas, por ejemplo.

1.2. Objetivos del proyecto

El objetivo de este proyecto es diseñar, implementar y evaluar un sistema capaz de analizar el sentimiento en textos escritos.

Por un lado se intentará que el sistema sea capaz de detectar el sentimiento global del texto, pudiendo ser muy positivo, positivo, neutral, negativo o muy negativo. Este resultado intentará identificar de alguna manera el estado emocional de la persona que lo escribió, es decir, si se encontraba feliz, emocionada, animada (positivo o muy positivo) si se encontraba enfada, angustiada, triste (negativo o muy negativo) o si expresaba una emoción que fuera una mezcla a partes iguales entre las dos anteriores (neutral).

Por otro lado, se debe hacer hincapié en conseguir que el sistema sea capaz de detectar los conceptos y entidades que aparezcan en el texto, y en que sea capaz de relacionarlos con los sentimientos encontrados, dando una polaridad (o puntuación) a cada entidad y concepto, que expresará el nivel de afinidad de la persona con esa entidad o concepto.

Al primer tipo de análisis nos referiremos como “Análisis de sentimiento global”, mientras que al segundo lo nombraremos como “Análisis orientado a aspectos”.

Por ejemplo, en una ejecución normal del sistema, a éste se le podría dar como entrada el siguiente texto “Me arrepiento muchísimo de haberme comprado un iPhone. Ojalá tuviera aquí mi Samsung” y el sistema de forma ideal debería ser capaz de respondernos aproximadamente así:

- Sentimiento global: muy negativo
- Aspectos:
 - o iPhone: Negativo
 - o Samsung: Positivo

Para realizar estas tareas, el sistema se diseñará como un sistema orientado a reglas sobre las capacidades que nos da el software Textalytics, un potente motor de procesamiento del lenguaje natural de la empresa Daedalus [2], capaz de realizar tareas como el análisis sintáctico del texto, de forma que podamos basar nuestras reglas para la detección del sentimiento en la estructura de la frase, las relaciones entre los sintagmas y la categoría gramatical de cada palabra procesada.

El software que se diseñe sobre Textalytics actuará como “motor” leyendo y aplicando un conjunto de reglas escritas en un diccionario. Estas reglas se basarán en asignar polaridad a palabras o lemas. Además se podrán asignar operadores más complejos que asignen polaridades de forma variable, en función del contexto, de la categoría gramatical de la palabra o del resto de sentimientos encontrados en el texto, permitiendo así crear reglas complejas y avanzadas, con las que se pueda asignar sin error las polaridades adecuadas aún en casos ambiguos.

Tras el diseño e implementación del sistema, se pasará a evaluarlo y comprobar la efectividad del mismo. Para ello se ejecutará el software sobre una serie de tweets y se enfrentará la salida del programa con un etiquetado humano del sentimiento. A partir de aquí, se analizarán los resultados, buscando problemas en el análisis y proponiendo soluciones a los errores hallados.

1.3. Contenido de la memoria

El presente documento se ha estructurado de la siguiente forma:

En primer lugar, tras esta breve introducción, encontramos el capítulo referente al “**Estado del arte**”. En él se describe de forma general todos los conceptos relacionados con el proyecto, así como las problemáticas a las que debemos enfrentarnos durante el desarrollo del trabajo. También se describirá de forma breve el estado actual de los algoritmos y técnicas relacionadas con él.

Posteriormente, en el capítulo 3, “**Diseño del sistema**” abordaremos el diseño que se ha decidido implementar, haciendo una descripción detallada de su arquitectura, y del funcionamiento de cada una de sus partes.

Seguidamente, el punto 4, correspondiente a la “**Implementación del sistema**”, donde se especifican detalles más específicos sobre la forma en la que hemos desarrollado el sistema definido en el punto anterior.

Para concluir, comentaremos la evaluación realizada del sistema en el capítulo 5 de “**Evaluación**” que será seguido de un último capítulo de “**Conclusiones**” junto con una breve reflexión sobre las posibles mejoras y líneas de trabajo futuras del sistema.

2. Estado del arte

El proyecto que se nos presenta requiere de una cantidad elevada de conocimientos transversales, que tocan muchas ramas de la ciencia computacional, así como conocimientos de lingüística, y conocimientos generales del uso del lenguaje natural en distintos escenarios, como el lenguaje de las redes sociales, el cual posee sus propias particularidades que lo hacen especialmente difícil de procesar.

En este apartado se intentará dar una visión general de todos los aspectos teóricos que se relacionan con el proyecto, así como describir brevemente el estado actual de las tecnologías vinculadas a él y otros conocimientos relevantes, sus problemáticas y las técnicas relacionadas con el análisis de sentimientos.

2.1. Historia, interés y actualidad del análisis automático de sentimientos

Como ya hemos desarrollado brevemente en la introducción de este documento, cuando se habla de análisis de sentimientos nos referimos a la capacidad de detección, recolección y análisis a través de un sistema automático, de la información subjetiva contenida en un texto, es decir, opiniones y expresiones con carga emocional. Dicho de una forma más resumida, un sistema de estas características, sería capaz de decidir, con una cierta validez, si un texto expresa sentimientos positivos, neutrales o negativos. Por otro lado, el llamado análisis orientado a aspectos, relacionado estrechamente con el análisis de sentimiento, es la capacidad de seguimiento de opiniones referidas a entidades, así como los juicios sobre diferentes características o eventos relacionados con esas mismas entidades.

Los primeros trabajos desarrollados en el ámbito del análisis automático de sentimientos y opiniones, son relativamente recientes, datan del año 2002. En el paper *"Thumbs up? Sentiment classification using machine learning techniques"* desarrollado por *Bo Pang, Lillian Lee, y Shivakumar Vaithyanathan*, se describe un método para clasificar *reviews* de películas como positivas o negativas, por medio de técnicas computacionales de aprendizaje supervisado para realizar la clasificación [3].

El análisis de sentimientos ha generado en los últimos años un gran interés en las distintas comunidades de investigación, así como en el mundo profesional, convirtiéndose en un ámbito de mucha competencia e inversión. Este gran interés se debe principalmente al gran crecimiento de los medios sociales como los blogs, o las redes sociales, que han incrementado en una gran cantidad los contenidos creados por usuarios entre los que podemos encontrar *reviews*, comentarios, valoraciones y otras

formas de opinión [4]. El gran interés que ha despertado el tema se refleja en lo mucho que se ha trabajado en la última década en este campo.

Para dar solución a este reto tecnológico, generalmente se requiere el uso de un diccionario compuesto de palabras y expresiones junto con una polaridad a priori etiquetada para cada entrada del mismo. Diversos trabajos de investigación han realizado diferentes algoritmos que trabajan con este diccionario y utilizan técnicas de procesamiento del lenguaje natural (PLN) con el fin de realizar promedios con las diferentes etiquetas de polaridad de las palabras que aparecen en el texto y que han sido etiquetadas con ayuda del diccionario (*Turney, 2002*) [5].

Sin embargo, han sido los métodos de **aprendizaje automático**, concretamente de tipo supervisado, los que han terminado por convertirse en el enfoque más ampliamente utilizado para enfrentarse al problema, debido principalmente a la rapidez con la que se pueden generar con ellos un modelo de análisis de sentimientos a través de entrenamiento.

Las técnicas de aprendizaje supervisado consisten, de forma muy básica, en dar como entrada al sistema una serie de ejemplos junto con la salida deseada para cada uno, con los que éste se entrena [6]. Así el sistema generaría un modelo informático, que en fondo serían una serie de reglas deterministas, las cuales posteriormente le permitirían obtener la salida para nuevas entradas.

Los algoritmos que se pueden utilizar para generar el modelo a través de los ejemplos son muy variados (máxima entropía, *support vector machines* (SMV), expectation maximization, Naive Bayes, ...) pero en el fondo todos buscan lo mismo, realizar un modelo de regresión con los ejemplos proporcionados (entrenamiento), que permita estimar la salida del sistema para nuevas entradas. En el caso de su aplicación concreta para el análisis de sentimientos, la entrada del sistema sería un texto y la salida sería correspondiente su etiquetado.

Como ejemplo de sistema basado en aprendizaje computacional, el desarrollado por *Bo Pang, Lillian Lee, y Shivakumar Vaithyanathan* [3] propuso un modelo unigramas utilizando Support Vector Machines que no necesita ningún diccionario previo para clasificar críticas de películas.

Las últimas y más novedosas investigaciones han enfocado sus esfuerzos en realizar análisis de sentimientos sobre redes sociales de microblogging como Twitter. Por ejemplo, *Bollen, Mao y Zeng* (2010) estudiaron la posibilidad de realizar predicciones sobre los resultados del mercado bursátil a través del análisis de los sentimientos expresados en Twitter sobre él [7].

Las características concretas del lenguaje usado en redes sociales hace necesario realizar un tratamiento especial sobre el texto analizado. Por ejemplo en Twitter se

usan diferentes propias de esta red (RT, TT, @user, #tag...), además del uso de emoticonos y otras jergas habitualmente usadas entre internautas. A estos fenómenos se le suma las frases mal construidas o con errores ortográficos, resultando mensajes de difícil interpretación mediante herramientas habituales de PLN [8].

Muchos autores han optado por realizar diferentes técnicas de preprocesado con la intención de normalizar el texto antes de ser analizado, mejorando así los resultados obtenidos. Una vez se ha normalizado el texto, las técnicas habituales de PLN, como la extracción de lemas y el etiquetado POS pueden ser útiles para el análisis de los mensajes.

Diseñado un sistema, gracias a las técnicas de análisis de sentimiento automático, se podría analizar esta gran cantidad de información pública, pudiendo generar de forma rápida, por ejemplo, un informe sobre el impacto de un cierto producto, o conocer la percepción social que tiene una cierta marca, persona, o evento, convirtiéndose por tanto, en técnicas de gran interés para las empresas de marketing y de gestión de relaciones con los clientes.

La actualidad del tema ha sido también la causa de la creación de diferentes congresos, talleres y comunidades de investigación dedicados al análisis de sentimientos y otros temas relacionados en diferentes lugares del mundo.

Por ejemplo, en Estados Unidos existe el TREC (Text Retrieval Conference) [9] en el que podemos encontrar diferentes publicaciones de investigación sobre "*Opinion retrieval*", así como en Japón tenemos el congreso NTCIR (NII Testbeds and Community for Information access Research) [10] el cual celebra diferentes talleres para promover la investigación y el desarrollo en diferentes tecnologías de acceso a información, como los sistemas de recuperación de información o los sistemas de respuesta automáticos.

En Europa existe el congreso CLEF (Conference and Labs of the Evaluation Forum) [11] que de la misma forma, centra sus actividades en promover la investigación en este tipo de sistemas.

España cuenta con la SEPLN (Sociedad Española para el Procesado del Lenguaje Natural) [12] una organización que actualmente dirige cada año una workshop, entre otras, sobre análisis de sentimientos centrado en el lenguaje español como parte de su conferencia anual.

También como causa de ese interés, existe hoy en día una numerosa, y novedosa, serie de alternativas en el mercado del software para dar solución al problema en diferentes idiomas.

Por citar algunas de las diferentes soluciones comerciales que se pueden encontrar hoy en día en el mercado el análisis de sentimientos:

- **Textalytics** [13]
Textalytics, de la empresa Daedalus, es un motor de análisis de texto en lenguaje natural con múltiples funcionalidades, entre las que se encuentra el análisis de sentimientos en varios idiomas: español, inglés y francés. Además de analizar la polaridad asociada al texto, oración, concepto y entidad, es capaz de identificar otros aspectos relevantes, como la objetividad/subjetividad del mismo, así como el uso de la Ironía.

- **AlchemyAPI** [14]
La gran compañía AlchemyAPI, nacida en el año 2009, ofrece diferentes productos en forma de APIs para dar solución a distintos problemas relacionados con la inteligencia artificial. Entre otras, ofrece su propia API de análisis de sentimientos, basada en técnicas de procesamiento natural del lenguaje, la cual es capaz de realizar un análisis de sentimientos a diferentes niveles: a nivel de documento, de entidad y a nivel de palabras clave.

- **Semantria** [15]
Es una API de análisis de sentimiento de la compañía de procesamiento del lenguaje natural, Lexalytics. Su análisis se basa en técnicas de Machine Learning, con las que es capaz de analizar un texto y devolver tanto la polaridad general del texto como la polaridad asociada a la reputación de las entidades del texto. Sus servicios están disponibles en varios idiomas, entre ellos inglés, español y chino.

- **Repustate** [16]
Repustate es una compañía dedicada al análisis de sentimientos y el procesamiento de lenguaje natural. Posee de una API de análisis de sentimiento disponible en varios idiomas, la cual es capaz de analizar el sentimiento a diferentes niveles del texto.

Además de las herramientas citadas en la lista anterior, existe otra gran multitud de utilidades relacionadas con el procesamiento natural del lenguaje, así como con técnicas de Machine Learning, las cuales se utilizan en algunas ocasiones para realizar tareas relacionadas con el análisis de sentimientos. Por citar un par de ellas de diferente naturaleza:

- **NLTK [17]**
Natural Language Toolkit (NLTK) son unas librerías open source a través de las que se ofrece una plataforma para desarrollar aplicaciones en Python de procesamiento del lenguaje natural. Hoy en día existen algunas APIS de análisis de sentimiento basadas en este toolkit open source, como el sistema “Sentiment” publicado en la web text-procesing.com [18].
- **Google Prediction API [19]**
Google ofrece una serie de documentos online para desarrolladores sobre cómo realizar análisis de sentimiento a través de su API de Machine Learning.

2.2. Problemática del análisis de sentimiento

Antes de enfrentarnos al diseño del sistema, es necesario llevar un análisis de los problemas a los que nos enfrentamos para lograrlo, ya que el análisis de sentimientos es un problema realmente complejo de resolver.

Por un lado, el intento de realizar análisis sobre textos de opinión, nos va a forzar a manejar de forma computacional el lenguaje natural, lo cual implica lidiar con una serie de dificultades debidas a la propia naturaleza del lenguaje natural humano.

Por otro lado, independientemente de la dificultad del procesamiento de texto escrito en lenguaje natural, siguen existiendo una serie de problemas inherentes al análisis de sentimientos. La interpretación del sentimiento de un texto no es sencilla, ni siquiera para un humano, y cuanto más corto sea el texto (en Twitter por ejemplo, cada mensaje escrito por un usuario se limita a escasos 140 caracteres), más difícilmente es éste analizable, debido al poco conocimiento del contexto del mismo.

2.2.1. Problemática del procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN o NLP en sus siglas anglosajonas) es una rama de la Inteligencia Artificial encargada de estudiar métodos de comunicación entre máquina y hombre a través del lenguaje natural, es decir, el lenguaje empleado de forma habitual en una conversación escrita u oral entre personas.

El lenguaje natural presenta muchas características que lo hacen un verdadero reto para las ciencias de la computación. Como suele pasar en el mundo de la Inteligencia Artificial, una tarea que puede parecer muy sencilla para nosotros, como es utilizar el habla para comunicarnos, resulta un verdadero desafío para una máquina. Aspectos como la ambigüedad, la espontaneidad, la falta de fluidez, o las referencias y

abreviaturas, quedan muy lejos de ser completamente definidas por las disciplinadas reglas que rigen el comportamiento de un sistema informático. No obstante, la gran utilidad de un sistema que fuera capaz de usar el lenguaje natural llevó a muchos científicos e ingenieros desde finales de la década de 1940 a desarrollar sistemas que se acercaran lo máximo posible a este comportamiento [20].

Por lo tanto, las técnicas de la también llamada *Computación Lingüística*, actualmente son capaces de desarrollar sistemas que pueden trabajar con lenguaje humano de una forma más o menos efectiva, ya sea recibiendo instrucciones en lenguaje natural, produciendo una interfaz de respuesta conversacional o procesando y realizando tareas de análisis de lenguaje humano, como es el caso que nos concierne para nuestro sistema.

Problemática del lenguaje natural

El principal problema al que se enfrenta un sistema de este tipo es la tremenda ambigüedad del lenguaje natural, la cual se presenta como distintos fenómenos a distintos niveles del lenguaje.

Homonimia

La homonimia se da cuando palabras que tienen diferentes significados se escriben, o en el caso del lenguaje oral, suenan, de la misma forma. En el primer caso estamos hablando de palabras homógrafas, mientras que en el segundo, hablamos de palabras homófonas. Para nuestro caso, lenguaje natural escrito, esto supone que el sistema no podrá identificar con total certeza una palabra comparando únicamente su forma.

Tomando como ejemplo la palabra homónima bonito, sus acepciones, según la RAE (Real Academia Española) [21] son:

- Bonito (Sustantivo): Pez teleósteo comestible, parecido al atún, pero más pequeño.
- Bonito (Adjetivo): Lindo, agraciado de cierta proporción y belleza.

Ejemplo de uso:

- “El **bonito** es muy sano” [Sustantivo]
- “¡Qué **bonito** día!” [Adjetivo]

En ambas oraciones se utiliza una palabra que se escribe igual “bonito”, sin embargo su significado es completamente diferente, por lo que se debe dotar al sistema de algún mecanismo de desambiguación capaz de distinguir ambos casos.

Polisemia

La polisemia es un fenómeno similar al anterior. Se da cuando una palabra tiene varios significados. En este caso los diferentes significados vienen de un origen común y se han creado debido a la evolución del significado de la misma. Por ejemplo, la palabra “tecla”, es una palabra polisémica con una gran variedad de significados diferentes, que han sido adquiridos poco a poco debido al uso que hacen los hablantes de la misma. Puede ser la tecla de un instrumento musical, o de una máquina de escribir, por ejemplo.

Anáforas y elipsis

Un fenómeno habitualmente usado como mecanismo de economía del lenguaje es el uso de las elipsis y anáforas. Las elipsis son la supresión de elementos en la construcción sintáctica, ya que éstas se pueden deducir del contexto. Y de una forma parecida, las anáforas son un elemento deíctico cuyo significado depende también del contexto.

- Estoy de paso por Madrid. (*Madrid*) Es maravillosa. [Anáfora]
- El concierto fue genial, la comida (*fue*) regular. [Elipsis]

El uso de estos elementos que tan útiles resultan en el lenguaje habitual para que la comunicación sea más fluida, añade una gran dificultad a la hora de su procesado.

Sintaxis no normalizada

Otro problema que nos plantea el análisis de lenguaje natural es el hecho de que su estructura no está normalizada, es decir, a la hora de utilizar el lenguaje en español, no se utiliza una sintaxis concreta y bien definida, si no que a la hora de expresar una misma idea, ésta se puede estructurar de diversas maneras.

- Me encantan las mañanas de domingo.
- Las mañanas de domingo me encantan.

Ambas oraciones expresan la misma idea utilizando un orden diferente en las distintas palabras que las componen.

El contexto

Tanto el contexto lingüístico como el resto de elementos externos (factores culturales, y sociales entre otros) que dan forma a una situación concreta de comunicación, pueden modificar de forma importante el significado de los elementos de un texto. Son muchos los factores que pueden afectar, y es a través de ellos que los humanos somos capaces de realizar una correcta interpretación de la comunicación. Por poner un ejemplo sencillo, en la siguiente conversación:

- ¿Se te ha vuelto a estropear el teléfono?
- Si, es **magnífico**.

La segunda oración contiene un sentimiento negativo, en forma de ironía. Detalle que no conoceríamos si sólo hubiéramos leído la contestación de la pregunta y que aún con el texto completo, resulta un caso muy complejo de tratar computacionalmente hablando.

Otros problemas

Aparte de toda esta serie de fenómenos que se dan en la lengua formal, hay que tener que cuenta otros aspectos que se dan en ambientes más coloquiales.

Los textos que se pueden recoger de una red social están llenos de errores gramaticales y de expresión. Faltas de ortografía, ausencia o uso incorrecto de puntuación son algunos de los factores que pueden influir tremendamente a la hora de analizar texto. También el uso de ciertos lenguajes o “jergas” propias del medio o el grupo de usuarios concreto es un asunto a tener en cuenta. En la siguiente tabla se muestran algunos ejemplos de cómo un mismo mensaje puede tener formas diferentes según el usuario y el medio en el que lo escriba.

| Ejemplo | Explicación |
|--|---|
| Lo contrario de vivir es no arriesgarse. | Oración gramaticalmente correcta. |
| lo contrario de vibir es no ariegarse | Faltas de ortografía |
| Lo CoNtrArio de ViVir, es No ARiEsGaRsE. ;-))) | Uso de mayúsculas emoticonos y otros caracteres a gusto del usuario |
| Lo contrario de vivir es no arriesgarseeeeeee!!! | Repetición de vocales para dar expresividad. |
| l cntrario d vivr s no arrsrgrse | Lenguaje SMS |

Tabla 1: Ejemplos de uso del lenguaje en las redes sociales

2.2.2. Problemática del análisis de sentimientos

Dejando de lado toda la problemática que surge del análisis del lenguaje natural, el análisis de sentimiento por sí sólo también nos plantea una serie de problemas que debemos resolver.

Es importante tener en cuenta, por ejemplo, que si se utilizan observadores humanos para etiquetar el sentimiento de un corpus de textos, este etiquetado sólo suele coincidir en aproximadamente un 79% de los casos, lo cual quiere decir, que si nuestro sistema etiquetara con una coincidencia cercana al 70% con el análisis humano, estaríamos ante un sistema de muy altas prestaciones, aunque el porcentaje no parezca demasiado alto [22].

A modo de ejemplo sencillo, pensemos en una serie de tweets que estamos analizando sobre un nuevo videojuego que ha salido al mercado. ¿Qué sentimiento se debería etiquetar a un tweet como el siguiente? (Ejemplo real obtenido de Twitter)

- “Japoneses reportan sticks de 3DS destrozados por culpa de Super Smash Bros”

¿Es positivo? Indica que los usuarios están jugando mucho al juego. ¿Es negativo? Los usuarios se quejan de la rotura de la consola. Como vemos, no siempre es sencillo etiquetar, aun mediante la ayuda de observadores humanos.

Otro asunto que puede dificultar en gran medida el análisis es la gran flexibilidad que tiene el lenguaje para expresar sentimientos y emociones. Veamos en los siguientes ejemplos como se puede expresar la opinión desde una forma más directa hasta formas realmente sutiles y por tanto, muy difíciles de captar:

- “Me **encanta** el nuevo Smash Bros”
- “Este videojuego es **horrible**, no puedo parar de jugarlo.”
- “Desembalando el Smash Bros. Quedan cerradas las visitas hasta nuevo aviso.”

En los tres ejemplos que acabamos de presentar el sentimiento es positivo, tanto a nivel global como a nivel de reputación, sin embargo, mientras que en el primer texto se expresa la opinión de una forma directa, en los dos siguientes ejemplos existen dificultades añadidas. En el segundo texto, por ejemplo, se utiliza el adjetivo “horrible”, con un cierto toque irónico, que fácilmente podría llevar el análisis a un resultado de polaridad negativa, cuando realmente lo que se quiere decir es que el juego es realmente adictivo, lo cual es positivo, cuando se habla de un videojuego. Por último, el tercer texto es un ejemplo de como expresar un sentimiento positivo de forma muy sutil. En este ejemplo se podría decir que la opinión se encuentra “oculta” en el sentido de la frase, por lo que para detectarla se necesitaría un análisis de muy alto nivel, algo muy complejo de lograr.

Por último, comentar brevemente que, como suele ocurrir en los sistemas relacionados con la minería de datos, va existir un compromiso inevitable entre la cobertura del sistema y la precisión del mismo [23]. Ello se refleja en el hecho de que los adjetivos calificativos pueden cambiar de polaridad según el dominio en el que se esté realizando el análisis. Por seguir con el mismo ejemplo, el adjetivo “adictivo” sería

positivo para un videojuego, mientras que de forma general sería negativo o sin polaridad (“El tabaco es muy adictivo” frente a “Es un videojuego muy adictivo”).

Así, cuanto más general queramos que sea el sistema, menos precisión va a tener y viceversa, cuanto más específico sea su uso, más preciso podrá ser su diseño.

2.3. Procesamiento del lenguaje natural

A continuación se hará una descripción general de las técnicas de procesamiento del lenguaje natural con las que contaremos para realizar el sistema, de forma que nos familiarizamos con la terminología de las técnicas usadas.

2.3.1. Técnicas del procesamiento del lenguaje natural

Tokenización

La primera técnica que se realiza sobre un texto que va a ser procesado con técnicas de PLN es la tokenización. La tokenización consiste en la división en “bloques”, llamados tokens, del flujo de caracteres que componen el texto. Ésta se produce a varios niveles, básicamente, en frases y palabras. La separación del texto posibilita un análisis independiente para cada elemento, así como establecer relaciones entre los tokens resultantes [24].

La tokenización se suele producir atendiendo al uso de los signos de puntuación del texto como los puntos, las comas, los espacios, y las interrogaciones. Algunas veces se atiende también a otros rasgos no tan propios de un texto formal, como el uso de emoticonos. Todo para conseguir dividir el texto en bloques independientes, con sentido significado independiente, que serán tratados posteriormente por el resto de procesos de análisis.

Lematización

El proceso de lematización consiste en asignar a cada palabra flexionada su lema, el cual se toma como forma canónica de ésta. El tener una forma canónica para representar a cada palabra del texto nos permitirá crear reglas más sencillas que afecten a un grupo de palabras sin tener que escribir todas las flexiones de ésta.

Por ejemplo, las formas flexionadas de un adjetivo son sus diferentes variaciones en género y número (bonito, bonitos, bonita, bonitas), así como a los verbos se les asigna su forma en infinitivo como forma canónica.

Part-of-speech tagging (POS)

El proceso de *Part-of-speech tagging*, abreviado POS, se corresponde con el análisis morfosintáctico del texto. Es el encargado de asignar a cada token del texto analizado su categoría morfosintáctica. De esta forma, tras pasar el texto por este procesado se asignará una etiqueta a cada token que indicará todos los aspectos relacionados con su gramática: categoría gramatical género, número, persona así como otros aspectos relevantes relacionados con el tipo de palabra concreto que se esté analizando.

Desambiguación

Los procesos de desambiguación son los encargados de resolver los problemas que se presentaron en el apartado anterior de “Problemática del procesamiento del lenguaje natural”. En ellos, como su propio nombre indica, se resuelven los problemas de ambigüedad que surgen por la naturaleza polisémica del lenguaje.

Las técnicas para realizar desambiguaciones son variadas. Van desde aplicar restricciones morfosintácticas, las cuales desambiguan el sentido concreto de una palabra en función de las categorías gramaticales que se usen en la construcción de la oración, hasta reglas más bien de tipo heurístico, o incluso desambiguación temáticas que identificando el tema del texto podrían por ejemplo resolver “Madrid” como equipo de fútbol frente a “Madrid” como ciudad en un contexto que se hable de fútbol.

Análisis sintáctico

El análisis sintáctico completo del texto se puede realizar en último lugar gracias a todos los procesos descritos anteriormente. Aunando toda la información que se dispone hasta este nivel, el análisis sintáctico es capaz de añadir una descripción funcional a cada parte de la oración, estableciendo además un árbol sintáctico con distintos niveles de segmentación en los que se establecen relaciones gramaticales del tipo sujeto, complemento del verbo, etc.

3. Diseño del sistema

3.1. Introducción

El proyecto que en este documento se describe consiste en el diseño e implementación de un nuevo módulo para el motor de procesamiento del lenguaje natural de Textalytics. Este nuevo módulo integrará las funcionalidades de análisis de sentimiento en el código del motor, realizando una mejora tanto en velocidad de ejecución como en potencia, utilizando todas las posibilidades que nos da como software para el procesamiento del lenguaje natural, dándole así un fuerte enfoque lingüístico.

Se pretende dotar al sistema de la capacidad de identificar las distintas expresiones de un texto, clasificándolas por polaridad (Sentimientos positivos o negativos) y realizando un análisis de sentimiento lo más fino posible, que nos permita conocer tanto el sentimiento global del texto, como la relación existente entre los distintos conceptos y entidades detectados con las polaridades encontradas en él, conociendo así la valoración del usuario sobre estos conceptos y entidades. Para este cometido, se ha optado por realizar un **sistema basado en reglas** con el que se pretende obtener una análisis muy fino y con un fuerte enfoque lingüístico.

Para el diseño del sistema, se ha tenido en cuenta las necesidades del mismo. Por un lado, al ser un sistema basado en reglas, podemos diferenciar dos partes claramente diferentes: el **diccionario** de reglas y el **motor** del sistema. Por otra parte, como el sistema se integrará como un nuevo módulo del software de procesamiento del lenguaje natural, el diseño se ha realizado sobre el código de éste, aprovechando todas las funcionalidades que ya tiene desarrolladas.

Podemos representar la arquitectura funcional del sistema como se ve en la siguiente figura (Ilustración 1).

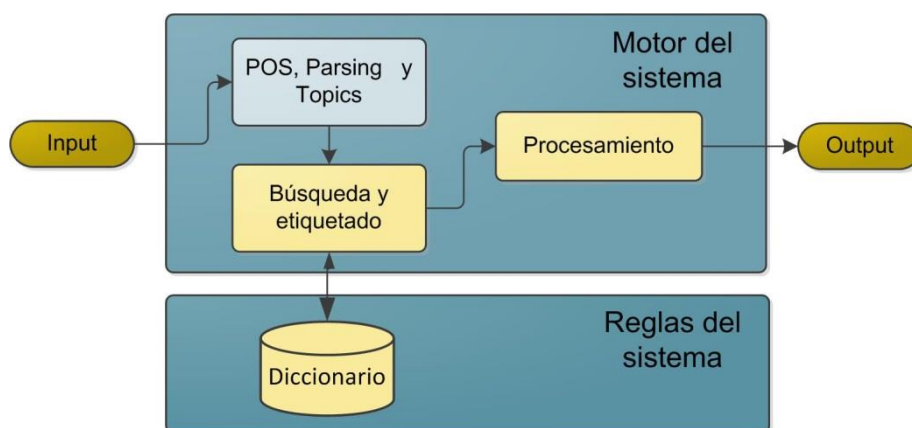


Ilustración 1: Arquitectura del sistema

La funcionalidad del sistema puede describirse brevemente viendo la Ilustración 1. Por un lado, la entrada de texto será preprocesada por el bloque de **POS, Parsing y Topic extraction**, el cual se encuentra ya diseñado, por lo que bastará con usarlo. Este bloque nos proveerá de información lingüística con la que trabajar, así como la detección de entidades y conceptos. Una vez la entrada se encuentra preprocesada, el bloque de **Búsqueda y etiquetado**, se encargará de leer las reglas del **diccionario** y aplicarlas sobre el texto, detectando los sentimientos que se encuentren en él, siguiendo las reglas definidas en éste. Por último, el texto etiquetado según el diccionario se procesa en el bloque de **Procesamiento**, encargado de extraer el sentimiento del texto etiquetado, escogiendo entre todas las etiquetas de polaridad y agregándolas siguiendo un cierto algoritmo hasta obtener el sentimiento global y el sentimiento asignado a cada entidad/concepto para plasmarlo en la salida.

A continuación se explicará más detalladamente el diseño para cada bloque.

3.2. Input: Entrada del sistema

La entrada del sistema es el texto que se pretende analizar. Éste puede ser de cualquier tipo, desde un largo texto de opinión, escrito con las más estrictas reglas de ortografía y estilo, hasta un simple tweet, muy corto y escrito con las particularidades de un texto no formal (Abreviaciones, faltas de puntuación, ortografía, uso de emoticonos, hashtags, URLs...).

El encargado de resolver esta gran variabilidad entre las posibles entradas será el bloque de POS y Parsing, que para nuestro sistema actúa como un preprocesado de la entrada, normalizándola, troceándola y etiquetándola con la información necesaria para realizar el resto del procesado. Por otra parte, el diseño de las reglas de diccionario también debe ser acorde al tipo de texto que se vaya analizar. Para realizar un buen análisis, las reglas de diccionario deben contemplar aspectos como idioma, jergas y temática del mismo.

3.3. Output: Salida del sistema

La salida de nuestro sistema será diseñada teniendo en cuenta que el sistema formará parte de las APIs de Textalytics.

Al tratarse de una herramienta que servirá de API, su salida debe ser estructurada y con un formato de intercambio de datos adecuado, ya que el destinatario de la salida de nuestro sistema será otro sistema software que realizará otras tareas con el resultado del nuestro.

```

{
  model: "es-general",
  score_tag: "NEU",
  agreement: "DISAGREEMENT",
  subjectivity: "SUBJECTIVE",
  irony: "NONIRONIC",
  sentence_list: [{
    text: "Me gusta Spotify pero es caro",
    inip: "0",
    endp: "28",
    score_tag: "NEU",
    agreement: "DISAGREEMENT",
    segment_list: [{
      text: "Me gusta Spotify",
      inip: "0",
      endp: "15",
      score_tag: "P",
      agreement: "AGREEMENT",
      polarity_term_list: [{
        text: "gustar",
        score_tag: "P",
        inip: "3",
        endp: "7",
        tag_stack: "=P"
      }],
      entity_list: [{
        form: "Spotify",
        variant: "Spotify",
        tag: "P",
        inip: "9",
        endp: "15",
        type: "Top>Product"
      }],
      concept_list: []
    }],
  },
  {
    text: "es caro",
    inip: "22",
    endp: "28",
    score_tag: "N",
    agreement: "AGREEMENT",
    polarity_term_list: [{
      text: "caro",
      score_tag: "N",
      inip: "25",
      endp: "28",
      tag_stack: "N"
    }],
    entity_list: [{
      form: "Spotify",
      variant: "Spotify",
      tag: "N",
      inip: "9",
      endp: "15",
      type: "Top>Product"
    }],
    concept_list: []
  }],
  entity_list: [{
    text: "Spotify",
    type: "Top>Product",
    tag: "NEU"
  }],
  concept_list: []
},
entity_list: [{
  text: "Spotify",
  type: "Top>Product",
  tag: "NEU"
}],
concept_list: []
}

```

Como formato de intercambio de datos se ha elegido **JSON** (JavaScript Object Notation), por ser un formato de intercambio de datos muy popular y de uso muy extendido en la actualidad para el intercambio de información entre aplicaciones web como es el caso de las APIs [25].

Además el formato JSON dispone de librerías para su fácil uso e integración bajo la mayoría de lenguajes populares entre los desarrolladores web, como PHP, JavaScript y Python, lo que ha posibilitado la generalización de su uso.

Como ejemplo de la estructura decidida para la salida, se presenta la salida concreta para la entrada "Me gusta Spotify pero es caro", sobre la que se explicarán sus partes más importantes (Ejemplo 1).

Como se puede ver de forma concreta en la salida presentada, los campos principales de ésta son los que se resumen en la siguiente tabla.

| Campo | Descripción |
|----------------------|--|
| model | Representa el modelo de diccionario elegido para el análisis. |
| score_tag | Polaridad global calculada para toda la entrada. |
| agreement | Marca si el texto contiene opiniones de polaridades diferentes (DISAGREEMENT) o si su opinión es uniforme (AGREEMENT). |
| subjectivity | Marca si el texto es subjetivo (SUBJECTIVE) u objetivo (OBJECTIVE). |
| irony | Marca si el texto es irónico (IRONIC) o no (NONIRONIC). |
| sentence_list | Un array que contiene las oraciones en las que se ha segmentado el texto. |
| segment_list | Un array de segmentos, porciones del texto de entrada total, en las que se ha encontrado expresiones con polaridad asociada. |
| concept_list | Un array con los conceptos encontrados en el texto. |
| entity_list | Un array con las entidades encontradas en el texto. |

Tabla 2: Campos de salida del sistema

Centrándonos en el ámbito de este proyecto, vamos a explicar detenidamente las variables de salida más importantes del software.

3.3.1. score_tag

El campo `score_tag` representa una polaridad asociada. Su uso en la salida del programa no sólo se restringe a el sentimiento global de la entrada, con él se expresa también la polaridad asociada tanto a palabras clave, entidades, conceptos, y segmentos de texto.

Los valores que puede tomar este campo son los mostrados en la tabla siguiente.

| Valor | Significado |
|-------------|--------------------------|
| P+ | Sentimiento muy positivo |
| P | Sentimiento positivo |
| NEU | Sentimiento neutral |
| N | Sentimiento negativo |
| N+ | Sentimiento muy negativo |
| NONE | Carece de sentimiento. |

Tabla 3: score_tag

Internamente, estas etiquetas poseen un valor numérico asociado. Así cuando se tengan varios sentimientos afectando a una misma entidad se podrán realizar operaciones matemáticas con ellas como la media aritmética, con el fin de dar un valor representativo del sentimiento agregado.

En cuanto a un significado más profundo de ellas, si éstas se refieren al análisis de aspectos, es decir, si están aplicadas sobre entidades o conceptos, el valor de la etiqueta indica la valoración por parte del autor, de la entidad o concepto. Sin embargo, para el análisis de sentimiento global, las etiquetas indican el estado de ánimo del autor del texto.

3.3.2. Agreement, Subjectivity, Irony

Estos tres campos avanzan un poco más allá en el propósito del proyecto, pero, ya que el motor también es el encargado de gestionarlos, explicaremos brevemente en qué consisten.

- **Agreement**

Este parámetro es un marcador que indica cuándo se han encontrado en el texto de entrada opiniones de diferente polaridad. Es decir, un texto que valore aspectos positivos y negativos de un producto dará como resultado `DISAGREEMENT`, mientras que un texto que exprese una opinión uniforme, dará como resultado `AGREEMENT`.

- **Subjectivity**

La variable de salida Subjectivity, da una idea de si el texto analizado es objetivo o subjetivo. Un texto con expresiones de opinión, del tipo “Yo creo que ...”, “Me gusta que ...” dará como resultado que el texto es subjetivo (`SUBJECTIVE`), mientras que una noticia formal se detectará como objetiva (`OBJECTIVE`).

El cálculo de esta salida se basa en reglas de diccionario también, tal y como las que explicaremos en el apartado Diccionario de reglas.

- **Irony**

Irony nos informa de si el texto de entrada ha sido detectado como irónico (**IRONIC**) o no (**NONIRONIC**). Éste es un matiz muy difícil de extraer del texto de entrada. Su evaluación, también se basa en un sistema de reglas de diccionario que se integran, al igual que las reglas de detección de subjetividad en el mismo diccionario que explicamos en el apartado Diccionario de reglas.

3.3.3. **entity_list** y **concept_list**

Los arrays **entity_list** y **concept_list** son los encargados de almacenar las entidades y conceptos detectados en el texto junto con la polaridad asignada a los mismos. Como se puede comprobar en el ejemplo de salida (Ejemplo 1), estos arrays aparecen a distintos niveles, identificado en cada nivel las entidades y conceptos hasta ese mismo nivel, es decir, estos mismos arrays pueden estar contenidos por ejemplo dentro de un segmento, significando en ese caso, que contienen las entidades y conceptos encontrados en él.

3.3.4. **sentence_list**

La variable de salida **sentence_list** contiene un array formado por cada una de las oraciones en las que el sistema ha segmentado el texto.

Cada oración queda definida por una serie de variables, las cuales ya han sido explicadas en su mayoría como campos a nivel global de la salida, en este caso, su uso se limita al nivel de la oración. Tenemos variables para almacenar el texto de la oración, su posición, la polaridad obtenida a nivel de oración y arrays de conceptos y entidades encontrados en ella. Además, la oración posee un array más, llamado **segment_list** el cual contiene un análisis de sentimiento a más bajo nivel, y que por tanto, aporta más granularidad al resultado. A continuación se explica en detalle.

| Atributo | Descripción |
|---------------------|---|
| text | Texto contenido en la oración. |
| inip | Posición del carácter inicial de la oración en el texto completo |
| endp | Posición del carácter final de la oración en el texto completo. |
| score_tag | Polaridad global de la oración. |
| agreement | Marca si la oración contiene opiniones de polaridades diferentes (DISAGREEMENT) o si su opinión es uniforme (AGREEMENT) |
| segment_list | Array con los segmentos en los que se ha dividido el análisis. |
| entity_list | Un array con las entidades encontradas en la oración. |
| concept_list | Un array con los conceptos encontrados en la oración. |

Tabla 4: Atributos de una oración

3.3.5. `segment_list`

La variable de salida `segment_list` contiene un array formado por cada uno de los segmentos encontrados en la oración. Cuando hablamos de “segmentos” nos referimos a una porción del texto de entrada, en la que se realiza un análisis local de sentimiento. Los segmentos se van a corresponder básicamente con las oraciones coordinadas que se encuentren dentro de cada frase del texto.

Cada segmento se describe mediante una serie de variables (atributos) bastante similares a los ya explicados pero que ahora describen un análisis a un nivel aún más bajo (nivel de segmento). Incluimos una descripción completa de ello en la siguiente tabla.

| Atributo | Descripción |
|---------------------------|--|
| inip | Posición del carácter inicial del segmento en el texto completo |
| endp | Posición del carácter final del segmento en el texto completo. |
| text | Texto contenido en el segmento. |
| score_tag | Polaridad global del segmento. |
| agreement | Marca si el segmento contiene opiniones de polaridades diferentes (<code>DISAGREEMENT</code>) o si su opinión es uniforme (<code>AGREEMENT</code>) |
| polarity_term_list | Array de términos con polaridad, encontrados en el segmento. |
| concept_list | Un array con los conceptos encontrados en el segmento. |
| entity_list | Un array con las entidades encontradas en el segmento. |

Tabla 5: Atributos de un segmento

3.4. POS, Parsing y Topic extraction

Como hemos comentado anteriormente en la introducción (Apartado 3.1) las herramientas de las que nos servimos para realizar el preprocesado del texto de entrada se encuentran ya desarrolladas en el software de PLN, por lo que no es necesario realizar ningún diseño de este bloque. No está de más sin embargo, conocer las facultades de éste, para poder diseñar el resto de partes en función de las capacidades que nos aporta.

En primer lugar, entran en funcionamiento las herramientas de POS y Parsing, encargadas de realizar un análisis morfosintáctico completo del texto.

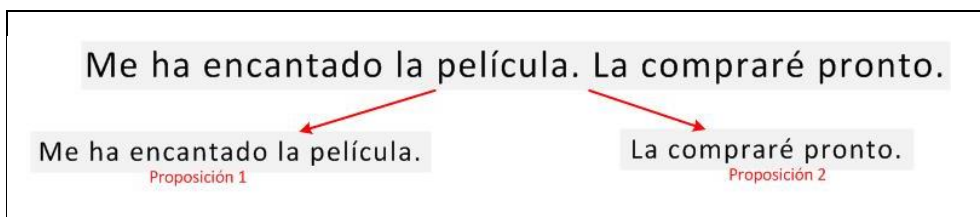


Ilustración 2: Separación por frases.

Como se contempla en la Ilustración 2, primero el sistema debe realizar una separación del texto en frases, lo cual realiza fijándose principalmente en la puntuación del texto. Las distintas frases halladas en el texto de entrada se tomarán como una unidad a procesar por el sistema, por lo que los posteriores bloques **trabajarán en serie con cada frase** encontrada.

El bloque de POS y Parsing continúa con su trabajo, separando cada frase en distintos tokens y a distintos niveles, en forma de **árbol** y además, los clasifica con etiquetas que marcan la función morfosintáctica (*POS tagging*) de cada token, como se ve a continuación.

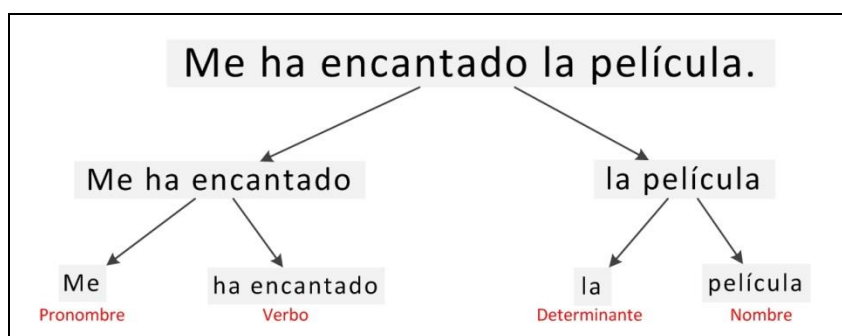


Ilustración 3: Segmentación en árbol

Junto con este análisis, también se realiza una lematización en cada token, como en la Ilustración 4.

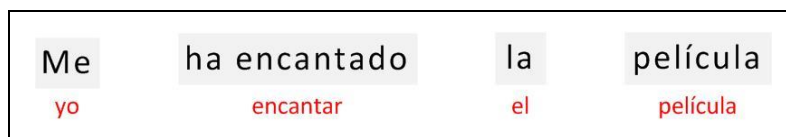


Ilustración 4: Lematización

Para realizar una lematización correcta, el sistema se vale de técnicas de desambiguación que le permiten determinar el lema en casos ambiguos. Sin embargo, en algunos casos especialmente difíciles, éste podría proponernos varios lemas por token, que también deberán ser tratados por el sistema diseñado posteriormente.

Aparte de la **segmentación en tokens**, el motor realiza una identificación de conceptos y entidades. Con los conceptos y entidades estamos identificando los términos más

significativos del texto, que luego serán utilizados para descubrir la relación existente entre ellos y los sentimientos del texto.

Con **entidades** nos estamos refiriendo a nombres de personas, empresas, marcas comerciales o lugares. Sería algo así como una identificación de nombres propios algo más sofisticada. Con **conceptos** nos referíamos a palabras significativas del texto, pero que en este caso, nombres comunes. Los conceptos a identificar pueden variar según el contexto del análisis que queramos realizar. Por ejemplo, en un análisis sobre la valoración de un cierto terminal móvil, los conceptos a identificar serían las características de un terminal, como la batería, la pantalla, el sonido etc... Una de las tareas a realizar antes de llevar a cabo uso plenamente efectivo del sistema, sería configurar adecuadamente la identificación de entidades y conceptos, tema que no trataremos en este escrito por ser una parte ajena a nuestro sistema, que la supone correctamente realizada.

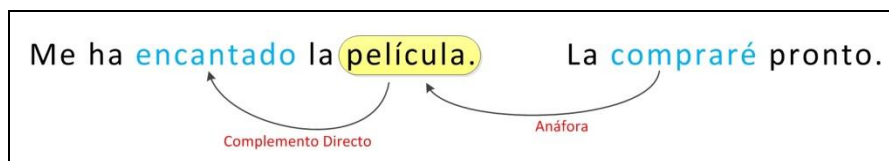


Paul Gilbert ofreció un genial concierto en la Sala Custom.

Ilustración 5: Identificación de entidades y conceptos

En el ejemplo presentado en la Ilustración 5 se ha detectado como entidad al músico “Paul Gilbert” y la sala de conciertos “Sala Custom”, mientras que como concepto, se ha identificado la palabra “concierto”.

Por último, el software establece en cada token las **relaciones sintácticas** encontradas. Esto es, se identifican relaciones entre dos tokens A y B, de forma que el token A queda relacionado con el token B, asignándole además el tipo de relación existente. Las relaciones que se detectan van desde los complementos verbales (Complemento directo, indirecto, de modo...) hasta, incluso, detección de anáforas entre distintas frases, permitiéndonos utilizar estas relaciones posteriormente para descubrir la relación entre los conceptos/entidades y las polaridades del texto.



Me ha encantado la película. La compraré pronto.

Complemento Directo

Anáfora

Ilustración 6: Detección de relaciones

Por concluir este apartado, en la Ilustración 7 se representa junta toda la información que obtenemos tras procesar el texto “Me ha encantado la película. La compraré pronto”: la segmentación en forma de árbol morfosintáctico, la lematización y las

relaciones semánticas encontradas. Esta imagen se ha tomado de la propia API de POS y Parsing de Textalytics [26].

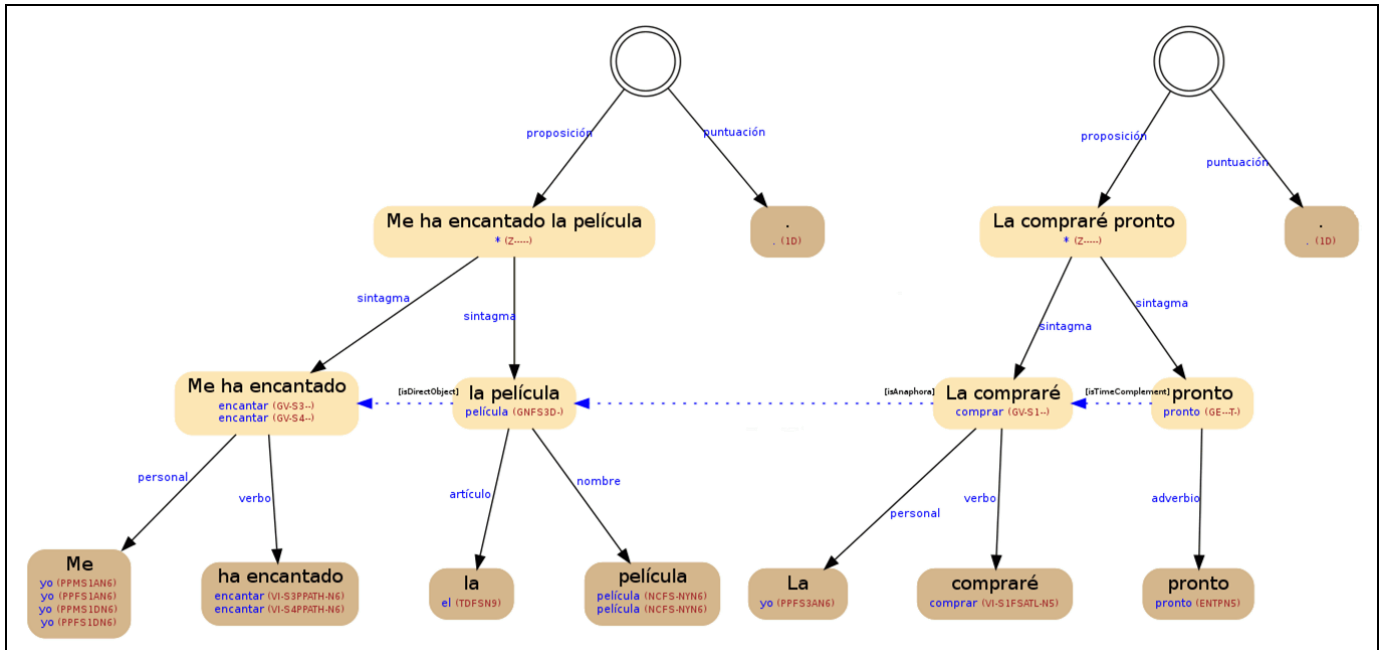


Ilustración 7: Salida del preprocesado

3.5. Diccionario de reglas

El diccionario es el bloque que contiene el “conocimiento” del sistema. En él se volcarán en forma de reglas las expresiones que el motor debe identificar como poseedores de información de opinión y sentimiento, así como identificar la polaridad de éstas. Gracias a éste, el motor puede realizar el análisis de sentimiento de forma adecuada.

La creación de un modelo de conocimiento general que englobe el análisis del sentimiento en toda la amplitud del lenguaje natural y que funcione a la perfección, resulta por supuesto, una tarea imposible. Sin embargo, teniendo en cuenta los problemas principales que se presentan cuando se trabaja con el lenguaje natural, podremos diseñar el diccionario de forma que se puedan crear modelos adecuados para trabajar en contextos un poco más reducidos de la forma lo más rápida y efectiva posible. Por ejemplo, podríamos crear diccionarios restringidos en idioma, y orientados al tema concreto que se quiera analizar.

Los diccionarios se han diseñado de forma que se pueda lidiar con los principales problemas que surgen al trabajar con lenguaje natural, por lo que se ha dotado al diccionario de mecanismos de desambiguación, identificación de términos por lema y creación de alias en forma de macros.

El diccionario se organizará por líneas. Para cada línea se definen varios campos, separados por tabuladores¹.

Existirán dos tipos de líneas posibles, las de **definición de macros** y las de **definición de reglas**, las cuales se explican a continuación.

3.5.1. Definición de macros

La creación de macros es posible en el diccionario. Ésta se utiliza con el único fin de simplificar la definición de reglas. Para declarar una macro, se utiliza el siguiente formato:

```
##NOMBRE_MACRO## \t termino_1|termino_2|termino_3|termino_4 ...
```

Ejemplo 2: Definición de una macro

Una macro crea una equivalencia entre el nombre de la macro y todos los términos que se escriban en su definición, separados por el carácter “|”. Así, si definimos la siguiente macro:

```
##FELIZ## contento|alegre|feliz|animado|enérgico
```

Ejemplo 3: Creación de una macro

Podremos utilizar simplemente el nombre de la macro, delimitándolo con el carácter “#”, para hacer referencia a todos los términos definidos. De esta forma, en el

Ejemplo 3, podremos escribir el nombre de la macro así #FELIZ#, y será equivalente a escribir todos los términos (contento|alegre|feliz|animado|enérgico)

Como detalle, las macros deberán ser declaradas siempre antes de ser usadas.

El uso de macros da una gran potencia a la hora de escribir reglas en el diccionario, ya que permiten lidiar con la grandísima riqueza del lenguaje natural, creando listas de términos equivalentes para su uso de forma rápida en la definición de reglas. Se podrá ver un ejemplo de uso de macros en un ejemplo posterior (Ejemplo 11).

¹ En este documento, y por motivos de visibilidad, los tabuladores se expresarán como “ \t “ o como una sucesión de espacios en blanco.

3.5.2. Definición de reglas

El aspecto simplificado de los campos existentes en una línea de definición de regla es el siguiente:

```
[REGLA] \t [POLARIDAD]
```

Ejemplo 4: Sintaxis general de una regla del diccionario

Como se puede ver en el Ejemplo 4, cada línea dispondrá de dos campos, el campo “regla” y el campo “polaridad”. En el primer campo, se define la regla, es decir, en este campo se especifican las palabras a las que se les aplicará la etiqueta de polaridad especificada en el segundo campo de la línea, el campo “polaridad”.

Así, en cada línea estamos definiendo una cierta regla de etiquetado de polaridad.

La definición de la regla se basa en la especificación directa de los términos (palabras) que entran en la regla, y además, nos valdremos de dos mecanismos para efectuar desambiguaciones, la categoría gramatical y el contexto. La estructura general del campo “regla” se especifica a continuación:

```
[TERMINO (@CATEGORIA) | TERMINO (@CATEGORIA) | · · · ] [ (CONTEXTO) ]
```

Ejemplo 5: Sintaxis del campo regla

Viendo el campo detenidamente, éste se compone de dos partes. Por un lado, una serie de términos que entran en la regla, separados por el carácter “|”, a los que opcionalmente se les puede especificar la categoría gramatical. Y seguidamente, en la segunda parte, también opcional², se escribe el contexto de la regla.

Dejando de lado la alta complejidad que presenta la escritura de una regla, primero nos centraremos en el uso de reglas sencillas, sin el uso de la categoría o el contexto, para así centrarnos en explicar cómo son los distintos operadores de polaridad que podemos aplicar sobre ésta, y más tarde, en el apartado “*Desambiguación: uso de la Categoría gramatical y el contexto*” se verá cómo hacer uso de la categoría gramatical y el contexto para desambiguar correctamente el texto de entrada.

3.5.2.1. Términos: forma y lema

Una regla sencilla tiene un aspecto como el que se puede apreciar en el Ejemplo 6. Ésta se compone de los términos que queremos etiquetar, listados uno detrás de otro y separados por el carácter “|”.

² Los paréntesis del Ejemplo 6 representan partes opcionales

bueno|bonito|barato

Ejemplo 6: Agrupación de términos

Cuando en la estructura de la regla (Ejemplo 4) hablamos de los términos (`TERMINO`), nos referimos a la palabras que entran en la regla, escritas directamente. Podemos escribir tanto la **forma** de una palabra como su **lema**, con el que englobaríamos todas las palabras que lo comparten. Así, podemos ir definiendo las palabras a las que les afecta la regla en cuestión.

Como ejemplo ilustrativo, para hacer referencia a todas las conjugaciones posibles del verbo “romper”, los escribiremos así, en infinitivo, ya que el lema de las distintas formas verbales, “rompí, romperé, he roto...” es el mismo, “romper”.

El uso del lema nos permite realizar reglas amplias sin tener que escribir todas las posibles flexiones de una palabra, agrupándolas bajo el lema. Esta característica es imprescindible para nuestro sistema pues sin ella sería muy difícil lidiar con la flexión del lenguaje natural en español.

3.5.2.2. *Expresiones multipalabra*

Para escribir expresiones que se compongan de varias palabras, hay que tener en cuenta que en el diccionario no se usan espacios. Para separarlas usaremos guion bajo “_”. Se puede ver un caso de uso en el Ejemplo 7. Las expresiones multipalabra son muy útiles para etiquetar expresiones largas con sentimiento único, como las frases hechas, o ciertas formaciones coloquiales habituales.

3.5.2.3. *Operadores de polaridad*

A continuación, y antes de empezar a explicar más de lleno los mecanismos más avanzados de los que el sistema dispondrá para realizar reglas complejas, pasamos a explicar el segundo campo de una línea de definición de regla, el campo de polaridad.

En el campo `[POLARIDAD]` (Ejemplo 4) se determina la polaridad que va a ser asignada a los términos que entren en la regla. Por un lado podemos escribir directamente la polaridad que queremos asignar, o se pueden usar operadores más complejos que actúan modificando al resto, o toman valores variables según el resto de etiquetas del texto. A continuación se detalla el uso de los distintos operadores.

Asignación directa de polaridad

La funcionalidad más básica del diccionario es asignar polaridad a un lema o palabra. Para ello nos bastará con escribir en el campo [POLARIDAD] la polaridad deseada, tal y como se describen en la tabla de etiquetas (Tabla 3: score_tag).

| | |
|------------------------|----|
| emocionante divertido | P |
| aburrido ser_una_chapa | N+ |

Ejemplo 7: Etiquetado básico de sentimiento

En este ejemplo estamos asignando a las palabras “emocionante” y “divertido” una polaridad positiva “P”, mientras que a la palabra “aburrido” y a expresiones como “es una chapa” se le asigna una polaridad muy negativa “N+”.

Operadores modificadores

Con los operadores modificadores, se modifican el resto de etiquetas de polaridad encontradas en el texto. Los operadores disponibles son los representados en la siguiente tabla.

| Operador | Nombre | Función |
|----------|-----------------------|---|
| + | Amplificador positivo | Aumenta el nivel de intensidad de una polaridad |
| - | Amplificador negativo | Disminuye el nivel de intensidad de una polaridad |
| NEG | Negador | Invierte la polaridad |
| !1 | Negador unario | Invierte la polaridad de un único sentimiento. |
| @ | Operador Nulo | Asigna polaridad nula a una palabra. |

Tabla 6: Operadores modificadores

La utilidad de los operadores amplificadores es reforzar, o disminuir la intensidad de un sentimiento, por ejemplo, si asignamos a la palabra “muy” el operador “+” el texto “Es muy divertido” dará como resultado “P+” en lugar de “P”, siguiendo el etiquetado del Ejemplo 7.

Por otra parte, el operador negador, sirve para modelar las propias partículas negadoras de un lenguaje. Así conseguiremos que la frase “No es divertido”, sea etiquetada como negativa, invirtiendo la polaridad “P” que se le etiqueta en una primera pasada. El caso del operador “!1”, su función es igual que la del operador “NEG” pero de una forma más limitada, sólo invierte la polaridad del primer sentimiento encontrado, útil para negadores más limitados como “sin”.

El operador nulo asigna una polaridad “vacía”, es decir, marca casos en los que no existe sentimiento. Su uso básicamente se limita eliminar falsos positivos mediante reglas de desambiguación, más tarde se verá el uso de este operador en ejemplos más avanzados (Ejemplo 9 y Ejemplo 11).

Operadores variables

En la siguiente tabla se describen resumidamente los operadores variables existentes.

| Operador | Nombre | Ejemplos de uso | Función |
|----------|-------------------------------------|-----------------|---|
| = | Igualador | =P =N =N+ | Adquiere el valor de otro sentimiento. |
| ! | Inversor | !P !N | Invierte el valor de otra polaridad. |
| += | Amplificador o igualador | +=P +=N | Adquiere el valor de otro sentimiento aumentándolo. |
| -= | Amplificador (negativo) o igualador | -=P -=N | Adquiere el valor de otro sentimiento disminuyéndolo. |
| . | Operador de preferencia | N. P+. P. | Marca una polaridad como polaridad final. |

Tabla 7: Operadores Variables

Estos operadores cobran sentido cuando se intenta determinar la polaridad de un segmento en el que se encuentren varios sentimientos etiquetados. Así, al agregarse con otras polaridades encontradas en el segmento, determinarán la polaridad final del mismo. Todos ellos se escriben junto con una polaridad a su derecha, que indica la polaridad que adquieren por defecto.

El **operador igualdad** "=", por ejemplo, adquiere la polaridad de su sentimiento con el que se agrega. En caso de no tener ninguna etiqueta con la que combinarse, como ocurre con todos los operadores variables, éste adquiere su polaridad por defecto, que se escribe a la derecha del mismo. Su uso quedará más claro después del siguiente ejemplo.

Imaginemos que se quiere asignar polaridad al lema "querer". En principio se puede pensar en etiquetar una "P", de esta forma un texto como "Quiero comprarme un iPhone" sería positivo, y en general, sería una buena aproximación para analizar el verbo "querer". Sin embargo, que ocurría si analizamos un texto escrito por un usuario que, descontento con su teléfono, escribe algo como "Quiero romper mi iPhone". Nuestro etiquetado asignaría un sentimiento positivo hacia iPhone, o en el mejor de los casos, neutro, si hemos asignado una polaridad negativa al lema "romper". Para estos casos, se utiliza el operador igualdad, que, como hemos descrito, "pierde" su polaridad al agregarse con otro sentimiento. Así, para resolver este caso, definiríamos las siguientes polaridades en el diccionario:

| | |
|--------|----|
| querer | =P |
| romper | N |

Ejemplo 8: Uso del operador igualdad

De esta forma:

- “Quiero [=P] un iPhone” => iPhone: P
- “Quiero [=P] romper [N] mi iPhone” => iPhone : N

Un uso parecido tienen el resto de operadores descritos en esta sección.

El **operador de inversión**, básicamente realiza la misma operación que el igualador, pero en lugar de quedarse con el sentimiento con el que es agregado, la invierte. Es útil para verbos que invierten el sentido de las polaridades de otros como “dejar” o “fracasar”.

Dicho con palabras coloquiales, “Dejar algo malo es bueno”, en un ejemplo, “Dejar el tabaco” sería positivo, el verbo dejar invertiría la polaridad de la palabra “tabaco”, que en principio, consideramos negativa.

Al final de este apartado se incluye una serie de ejemplo prácticos de uso de cada uno de los operadores que pueden aclarar las ideas sobre su uso.

Los **operadores de amplificación con igualdad**, actúan como amplificadores (o amplificadores negativos) si se agregan con otro sentimiento etiquetado. Si no encuentran con quien agregarse, el resultado final es su polaridad por defecto.

Por último, el **operador de preferencia**, se expresa colocando el carácter punto “.” detrás de una polaridad. En una agregación entre dos polaridades, la que tiende a dominar es la marcada con un este operador. Por ejemplo, el verbo “morir”, se podría marcar con este operador, para que prevaleciera su polaridad negativa en una agregación. “Ha muerto una buena persona”, sería así marcado como negativo.

3.5.2.4. *Desambiguación: uso de la Categoría gramatical y el contexto*

Una vez explicados los operadores que se pueden aplicar en el campo de polaridad, pasamos a explicar los dos mecanismos principales de desambiguación, que se usan a la hora de escribir la regla en el diccionario.

Categoría gramatical

El campo “categoría”, se escribe inmediatamente después de un lema o forma, utilizando como separador el operador de categoría, que se expresa con una arroba “@”.

Gracias a este campo, podemos restringir la categoría gramatical de la forma o lema que entra en la regla definida. Si este campo no se utiliza no se comprobarán las etiquetas gramaticales asociadas al lema, lo que significa que cualquier categoría posible activará la regla. Este campo resulta tremendamente útil para realizar desambiguaciones de palabras homónimas.

| | |
|----------|---|
| bonito@A | P |
| bonito@N | @ |

Ejemplo 9: Uso del operador de categoría

Siguiendo los significados de la palabra “bonito”, los cuales ya comentamos brevemente en el apartado “Problemática del procesamiento del lenguaje natural” (página 9), nos interesa, que en el caso de actuar como adjetivo tenga polaridad, mientras que en el caso de actuar como sustantivo no. Esto se consigue usando las reglas definidas en el Ejemplo 9. Ahora la forma “bonito” se etiquetaría como positiva, sólo, cuando está fuera un adjetivo, tal y como se expresa con el campo de categoría (@A). La segunda regla, en este caso resulta redundante, aunque no está demás definirla. Si luego en el diccionario se emplea más veces la forma “bonito”, nos aseguramos de que no se le asigna polaridad alguna cuando actúa como sustantivo (@N).

Las etiquetas morfosintácticas que podemos aplicar al campo de categoría son, entre otras, las definidas en la siguiente tabla:

| Etiqueta | Significado |
|----------|---------------|
| A | Adjetivo |
| E | Adverbio |
| T | Artículo |
| C | Conjunción |
| D | Demostrativo |
| L | Interjección |
| I | Interrogativo |
| N | Sustantivo |
| M | Numeral |
| H | Onomatopeya |
| P | Personal |
| S | Posesivo |
| Y | Preposición |
| Q | Cuantificador |
| V | Verbo |
| Z | Oración |

Tabla 8: Etiquetas morfosintácticas

Estas etiquetas son sólo un resumen. En la tabla se reflejan sólo las más útiles de todas las que se pueden aplicar. Incluso, además de especificar la categoría gramatical general de la palabra, se puede ser más específico, por ejemplo, si la regla que estuviéramos definiendo así lo requiriese, podríamos especificar que la palabra sea un nombre común y femenino, utilizando la etiqueta @NCF.

El uso completo de estas etiquetas se describe en la documentación oficial de Textalytics [27], donde se encuentra una extensa descripción de las etiquetas que se utilizan en la API del lematización y POS en la que hemos basado para el desarrollo.

Es necesario tener en cuenta, que los algoritmos de etiquetado morfosintáctico (POS) no son perfectos, por lo que el uso de este campo debe realizarse con precaución.

Contexto

El campo “contexto” sirve para realizar desambiguaciones basándose en el resto de palabras contenidas en una misma oración. Este campo no es otra cosa que una búsqueda de las palabras, o expresiones incluidas en él, de manera que sólo si la regla cumple su contexto, es decir, si alguna de las expresiones indicadas en el campo se encuentran contenidas en la misma oración, se activará la regla.

El uso del contexto a la hora de realizar una regla resulta esencial para poder desambiguar correctamente el uso de un lema.

Para especificar el contexto de un término, se podrán utilizar cualquiera de los siguientes operadores.

| Operador | Nombre | Función |
|----------|-----------------------|---|
| : | Contexto general | Busca el contexto en toda la oración. |
| > | Contexto a derechas | Busca el contexto a la derecha del término. |
| < | Contexto a izquierdas | Busca el contexto a la izquierda del término. |

Tabla 9: Operadores de contexto

Como ejemplo, supongamos el caso del verbo “subir”.

El verbo “subir” no parece tener una polaridad clara asociada, depende su uso en cada caso.

- “Argentina: El viernes la *gasolina* de 7 pesos **sube** a 10 pesos!!” => N
- “Hacienda da permiso a Interior para **subir** el *suelo* a los guardias civiles” => P
- “Sony opta por **subir** el *precio* de los periféricos de su videoconsola de sobremesa.” => N

Estos resultados se pueden lograr gracias al uso de contexto, definiendo las reglas del Ejemplo 10.

| | |
|-----------------------|---|
| subir:gasolina precio | N |
| subir>sueldo | P |

Ejemplo 10: Uso de contextos

Como se aprecia en el ejemplo propuesto, el contexto admite también una lista de términos separados con el operador “|”. La regla será activada pues, cuando se dé positivo en la búsqueda de cualquiera de los contextos especificados en la lista.

3.5.3. Otros detalles de la implementación del diccionario

Escapado de caracteres

Como ocurre muy a menudo en las ciencias de la computación, el uso de caracteres con significados especiales, como ocurre con los operadores de contexto, hace necesario dotar al lenguaje de algún recurso con el que poder referirse a este carácter de forma literal, sin recurrir a su uso como operador. Por ejemplo, para poder etiquetar un emoticono como “:-)” se debe poder distinguir cuando el uso de los dos puntos “:” es para formar el texto, como en este caso, o cuando se usan como operador, como en el caso de uso de contextos.

Para ello la técnica habitual es utilizar el escapado de caracteres, que consisten en anteponer una barra “\” delante de los caracteres que se quieran usar de forma literal.

Los caracteres que por tener otros usos en el diccionario deben ser escapados para ser utilizarlos de forma explícita, son los expresados en la siguiente lista (Tabla 10)

| Caracteres que deben ser espaciados | | | | | | | |
|-------------------------------------|---|---|--|---|---|---|---|
| : | < | > | | @ | _ | \ | # |

Tabla 10: Caracteres escapados

3.5.4. Anexo, ejemplo de reglas de diccionario

A continuación se incluye un ejemplo más completo con varias reglas de diccionario para la mejor comprensión del uso del mismo.

```

##POBRE##      abandonado|desamparado|desahuciado|desvalido|miserable|parado
##AYUDAR##     ayudar|auxiliar|socorrer|aportar|contribuir|colaborar
#AYUDAR#       =P
#POBRE#        N
#AYUDAR#>#POBRE#  P
los_miserables:película|trailer|musical  @
patatas_a_lo_pobre @
no|ningún|nadie|nunca|ni  NEG
estar_en_contra_de|ir_en_contra_de      !1
muy|bastante|tan|mucho|más|muchísimo    +

```

Ejemplo 11: Uso del diccionario

En el Ejemplo 11 se puede ver un uso más general del diccionario para definir distintas reglas. Primero, se definen un par de macros, en las que se crean alias de la palabra “pobre” y el verbo “ayudar”. Una vez creadas estas macros, se les asigna un valor por defecto (N y =P respectivamente) para después pasar a describir algunos casos más específicos mediante el uso de contexto. Por ejemplo, tal y como está definido “ayudar al pobre” (#AYUDAR#>#POBRE#) será positivo, y no negativo como se decidiría por la regla general. Además se desambiguan algunos ejemplos sin polaridad como el uso de la forma “los miserables” cuando ésta se refiere al musical con este título, o el uso de la palabra “pobre” para referirse a la receta “patatas_a_lo_pobre”, en cuyo caso, tampoco tendría polaridad asignada. Por último se definen de forma básica algunos negadores del lenguaje (no|ningún|nadie|nunca|ni) y algunos aumentativos (muy|bastante|tan|mucho| más|muchísimo).

Evidentemente, este ejemplo es muy poco exhaustivo con la creación de alias, la desambiguación o la definición de negadores y aumentativos. La complejidad del lenguaje natural requiere de un análisis muy profundo para crear reglas más completas. El ejemplo está simplificado para facilitar la comprensión del mismo.

3.6. Búsqueda y etiquetado

Este es el primer bloque que entra en funcionamiento tras el preprocesado del sistema. Básicamente, en él se manejan las reglas de diccionario, aplicándolas a los tokens de la entrada.

Las tareas de las que se encarga son:

- Carga completa del diccionario
- Ejecución de las reglas de etiquetado sobre el texto
- Ante la coexistencia de etiquetas, selección de las mejores reglas

3.6.1. Carga del diccionario

El diccionario está pensado para ser un archivo de texto en el que se escriban miles de líneas, y en las que además, con el uso de las macros, se definirá más de una regla por línea. El gran tamaño del diccionario puede ser un problema a la hora de leerlo y realizar búsquedas en él, aumentando considerablemente el tiempo de procesamiento del sistema. Por ello, se debe poner especial cuidado a la hora de implementar su carga en memoria y uso.

Para su uso de forma rápida y eficiente, se ha decidido que el diccionario debe ser almacenado en memoria en forma de Trie. Los Tries son una estructura de datos tipo árbol que permiten una rápida recuperación de la información almacenada en ella [28].

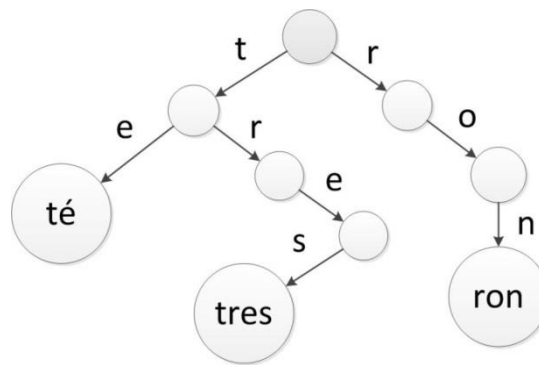


Ilustración 8: Trie

Los Tries almacenan la información en nodos indizados por una clave que corresponde precisamente con la información buscamos, por eso resultan muy rápidos a la hora de acceder a ésta. Por ejemplo, siguiendo el esquema del trie representado en la Ilustración 8, para buscar la información asociada a la palabra “ron”, nos guiaríamos usando sus propias letras (r, o, n) hasta llegar al nodo que contiene la información asociada a éste.

Para rellenar el trie con la información del diccionario, el sistema debe recorrer el diccionario línea a línea, tal y como se refleja en el diagrama de flujo mostrado en la Ilustración 9.

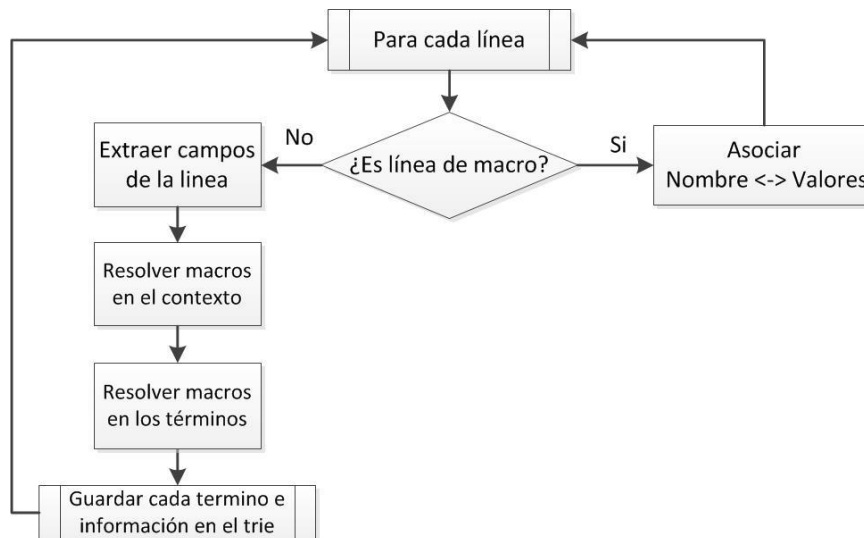


Ilustración 9: Carga del diccionario

El algoritmo de carga del diccionario determina el tipo de línea, basándose en si comienza o no por los caracteres “##” y analiza la línea extrayendo cada campo de información, separado por tabuladores o por los caracteres especiales explicados en la sección 3.5 “Diccionario de reglas”.

Si la línea es de definición de macro, ésta se almacena temporalmente como un par nombre-valor que nos permitirá resolver las macros que encontremos en el resto de líneas. Para las líneas de definición de reglas, se deberán resolver las macros encontradas y almacenar cada término creando un nodo en el trie e introduciendo en él la información asociada a la regla: la palabra, el contexto, la categoría gramatical y la etiqueta de polaridad asociada.

3.6.2. Búsqueda en el diccionario

La búsqueda de reglas que encajen con la entrada del sistema se realiza token a token, sobre cada frase detectada. Puesto que previamente se han cargado todas ellas sobre el trie, nos basta con realizar búsquedas en éste, usando como clave de búsqueda tanto la forma del token como sus posibles lemas. Debido al uso de contextos, en el diccionario se podrá encontrar más de una regla de etiquetado para cada lema o forma. Si realizamos la búsqueda para un lema concreto, es probable que encontremos más de una regla para aplicar sobre el token, por lo tanto, la siguiente tarea de este bloque será decidir, entre todas las reglas encontradas, cuales son válidas (comprobación de contextos y categoría gramatical), y de entre ellas, cual es la que debemos aplicar finalmente (regla prioritaria).

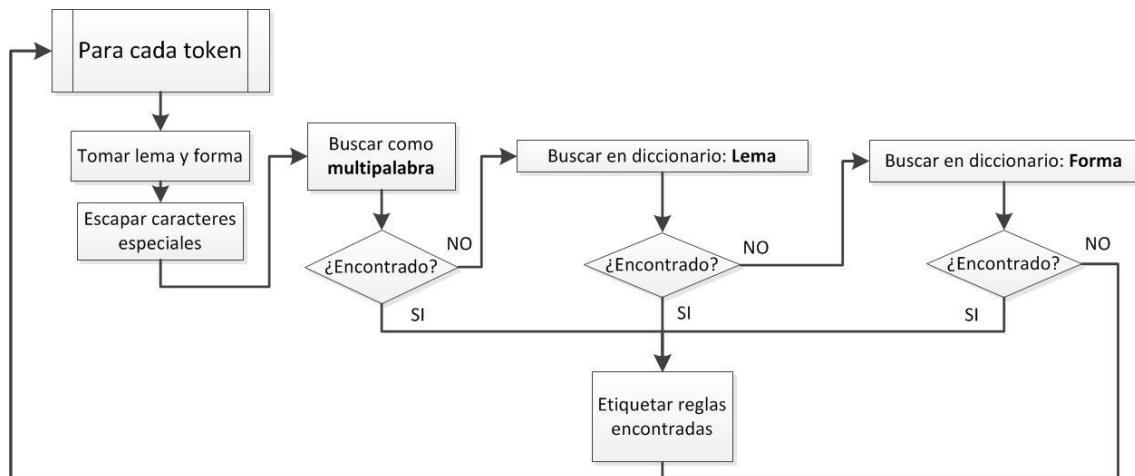


Ilustración 10: Búsqueda de reglas

Como se ve en la ilustración, la búsqueda se realiza siguiendo una prioridad: primero se buscan posibles multipalabras, es decir, expresiones de varias palabras, como puede ser “feliz_como_una_lombriz” frente a “feliz”, después se busca en el trie reglas asociadas al lema, dándoles prioridad frente a la forma que se busca en último lugar.

El bloque “Buscar como multipalabra” realiza una búsqueda recursiva en busca de una expresión multipalabra que comience en el token analizado. En este bloque también se busca en cada token que coincida en lema o en forma con prioridad en la coincidencia del lema.

3.6.3. Selección de reglas

La elección de qué regla aplicar entre todas las reglas encontradas para un token se realiza mediante una **selección por prioridad**. En esta selección se priorizan en primer lugar reglas que usen contextos, por ser más específicas, en segundo lugar reglas que no utilicen contextos pero utilicen desambiguación por categoría gramatical y por último reglas simples, sin uso de desambiguación. Frente a igualdad de prioridad con contexto, se toma el contexto más cercano, y ante igualdad en el resto de casos se elige la primera regla disponible.

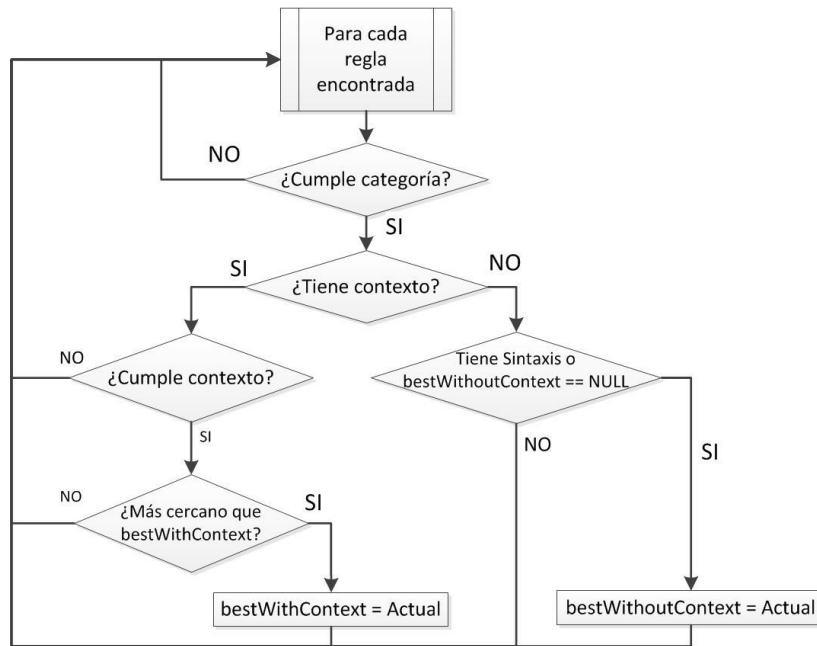


Ilustración 11: Selección por prioridad

En el diagrama se ilustra la selección por prioridad, que elige la mejor regla con contexto y sin él. Tras el funcionamiento de este pequeño algoritmo se elegiría la regla con contexto si existiera, o la regla sin contexto en caso contrario.

Además, durante esta selección, se está comprobando que cada regla cumple su contexto y la categoría gramatical, en caso contrario, sería descartada.

La comprobación de contexto se realiza dependiendo del operador de contexto de la regla (Tabla 9: Operadores de contexto) como una búsqueda del mismo hacia derecha, izquierda o en toda la frase. Durante esta búsqueda también se tendrá en cuenta tanto forma como lema a la hora de comparar el contexto buscado con los tokens de la oración.



Ilustración 12: Prioridad en elección de reglas

3.7. Procesamiento

3.7.1. Introducción

Una vez la entrada ha sido preprocesada y debidamente etiquetada, el motor debe realizar el resto del procesamiento hasta obtener la salida. Por lo tanto, la tarea de este bloque es la más compleja de diseñar, así como la más interesante. Su cometido será tratar las etiquetas de sentimiento del texto, junto con toda la información que nos ha aportado el núcleo del sistema (árbol morfosintáctico, relaciones semánticas, topics, etc.) hasta extraer la información deseada.

Supongamos como entrada un texto que opine sobre una película, como el siguiente: “Me encanta la película”.

En este punto del diagrama de flujo (Ilustración 1) el software cuenta con la siguiente información: por un lado, el etiquetado debida al bloque de POS Paring y extracción de Topics, que se puede ilustrar con un árbol morfosintáctico³ como el que se representa en la Ilustración 13.

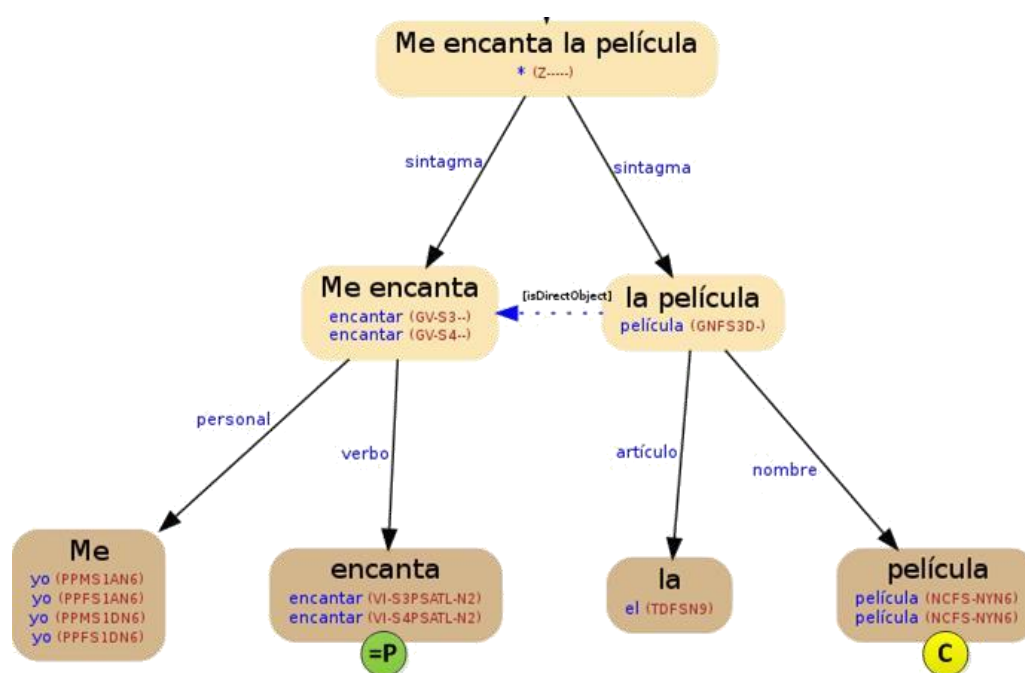


Ilustración 13: Información del preprocesado

Y las tareas de las que debe encargarse este bloque son:

- Detección de segmentos y cálculo de su sentimiento.

³ En la ilustración se ha representado sobre cada token las etiquetas de sentimiento que éste posee, si posee alguna, o se ha marcado con una “C” o “E” si el token ha sido identificado como concepto o entidad, respectivamente.

- Cálculo del sentimiento global a través de los segmentos detectados.
- Asociación entre sentimientos detectados y entidades/conceptos.

Viendo la información de la que disponemos en este caso (Ilustración 13) puede parecer una tarea sencilla, pero no lo es en absoluto. Este ejemplo es particularmente sencillo para simplificar la explicación. Vamos algunos ejemplos algo más elaborados:

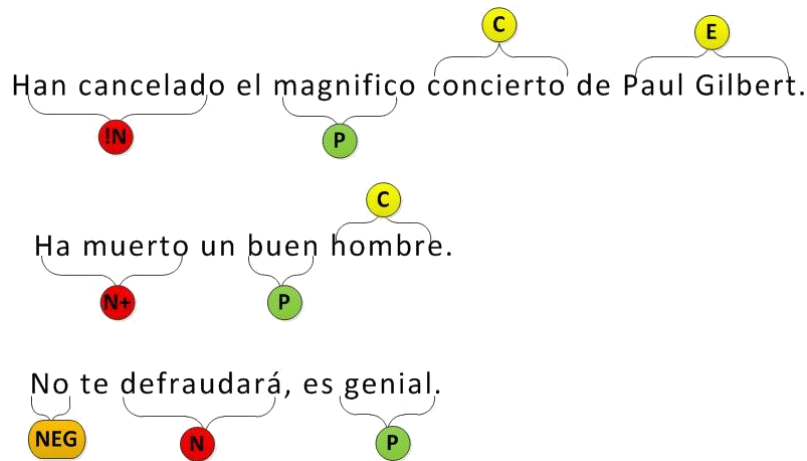


Ilustración 14: Ejemplos de etiquetado

Como vemos en la Ilustración 14, cuando nos encontramos varias etiquetas, la resolución del problema no es tan sencilla. Debemos enfrentarnos a varios problemas:

- ¿Qué etiquetas seleccionar como representantes y como relacionarlas entre ellas?
- ¿Cómo aplicar los negadores y hasta dónde?
- ¿Cómo asociar las entidades/conceptos con los sentimientos del texto?

En el primer ejemplo de la ilustración vemos que el sentimiento global es negativo, "Han cancelado el magnífico concierto", y que no debemos guiarnos de la etiqueta positiva "magnífico", más que para resolver el valor del verbo "cancelar", cuya polaridad es variable en función del resto.

En el segundo caso, "Ha muerto un buen hombre", ocurre algo parecido, tenemos dos sentimientos enfrentados, pero el sentimiento general de la frase es negativo.

En el tercer ejemplo, se enfatiza en la importancia de la aplicación correcta de los negadores. "No te defraudará, es genial", aquí el negador sólo debería afectar al sentimiento de "defraudar", y el diseño debería impedir que éste llegara hasta el adjetivo "genial", de lo contrario el sentimiento sería neutral ("No defraudar" positivo y "No genial" negativo). Puesto que el uso de negadores en el lenguaje natural es bastante variado, será complejo definir cómo debe actuar el motor con ellos.

3.7.2. Aplicación de negadores

La aplicación de los negadores es la primera tarea de la que se encarga este bloque. Nos resulta esencial conocer las etiquetas cuya polaridad se encuentra invertida, antes de continuar con el procesamiento.

El uso de negadores en español es muy variado y diverso. Por ello, no resulta sencillo realizar un modelo que funcione sin equivocación alguna. Para el diseño se ha tenido en cuenta que:

- La gran mayoría de negadores en español se aplican hacia la derecha.
- La distancia a la que éstos afectan es bastante limitada.
- Si se usan varios negadores en la oración, éstos no se afectan entre sí.

En estas tres reglas seguidas para el diseño de esta parte se resumen la mayoría de los casos de uso de los negadores en español.

Si vemos algunos ejemplos se puede ver que, efectivamente, actúan a derechas. “Me gusta la película pero **no puedo verla**”, “**Jamás le odié**” “Hay que disfrutar **en lugar de estar todo el día llorando**”. Existen algunos ejemplos en los que actúan hacia la izquierda, pero son casos muy aislados, coloquiales y complejos, “Os diré cuando voy a ver la película: **Nunca**” que semánticamente es equivalente a “**Nunca voy a ver la película**”.

Por otro lado, la tercera pauta de diseño que hemos comentado se debe al frecuente uso de la doble negación en español, caso en el que la concurrencia de esas dos o más “negaciones” no anula el sentido negativo del enunciado, sino que lo refuerza, al contrario de como ocurre en otras lenguas [29]. “**No le gusta ni que le abrace**”, “**No he visto ninguna buena película**”.

Para el funcionamiento de esta parte del bloque nos hemos basado en el árbol morfosintáctico. Localizada una partícula negadora, hemos definido una distancia máxima configurable en la que ésta puede subir en el árbol y afectar al resto de partículas encontradas, siempre hacia la derecha. Este comportamiento se explica gráficamente sobre el árbol morfosintáctico en la Ilustración 15.

Una vez encontrado un token en el que se puedan invertir partículas, el negador actúa sobre él, y termina su búsqueda sobre el árbol. Además, si el operador es de tipo “!1”, definido en la Tabla 6: Operadores modificadores, los cambios que realiza el operador son limitados a uno.

El funcionamiento completo se describe en el diagrama de la Ilustración 16.

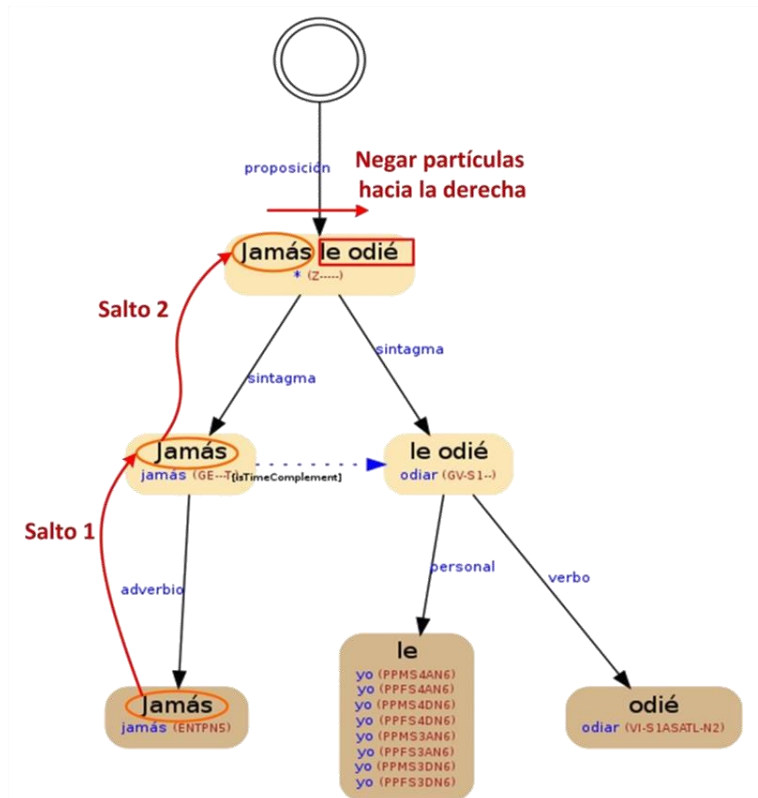


Ilustración 15: Negación sobre el árbol morfosintáctico

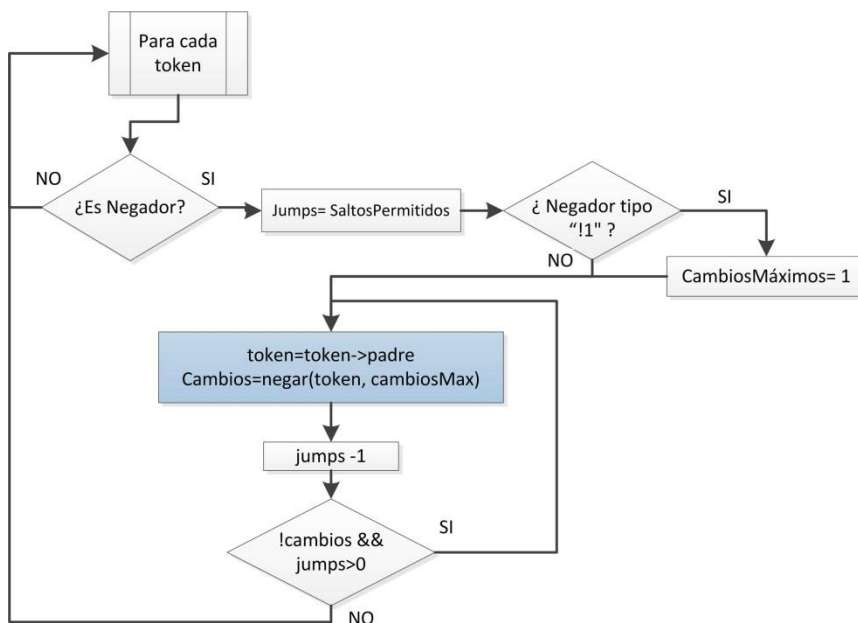


Ilustración 16: Negación de polaridades

3.7.3. Creación de segmentos

Para justificar el diseño de este apartado volvamos sobre los ejemplos que aparecen en la Ilustración 14.

Mirando sobre el segundo caso, “Ha muerto un buen hombre”, podemos plantearnos, ¿Este texto es positivo, neutro, o negativo? Por un lado podríamos decir que positivo, porque contiene un adjetivo que lo califica como positivo, “bueno”. La oración también contiene un verbo con un sentido negativo, “ha muerto”, por lo que se la podría calificar de negativa. Incluso, se podría considerar la oración como neutra, agregando ambas polaridades realizando una suma a modo de media aritmética.

No obstante, desde un punto de vista humano, la oración parece estar dotada de forma clara de un sentido negativo, e intuimos que, con alta probabilidad, su autor se está expresando con un estado de ánimo bajo. Ocurre pues, que debemos de alguna forma priorizar el sentimiento encontrado sobre el token “ha muerto”, frente al token “bueno”.

Reflexionando sobre este dilema, hemos decidido que la información de **sentimiento más relevante** se encuentra sobre los **verbos** de las oraciones. Es decir, lo que realmente nos interesa es que “El hombre **ha muerto**”, y es lo que debería dar el sentimiento negativo a la oración. Ocurre así con otros ejemplos: “Han **cancelado** el magnífico concierto”, “La gran empresa Telefónica **baja** en *bolsa*” “Tras una terrible adicción, **ha salido** de las *drogas*”. Por lo tanto, el diseño de este bloque se ha basado en buscar la polaridad de las oraciones sobre ellos.

Esto nos deja otras cuestiones a resolver:

- ¿Qué hacer con oraciones con varios verbos?
- ¿Cómo resolver los operadores variables ($!N$, $=P$, $+N$, ...) de los verbos?
- ¿Qué hacer en caso de que no exista verbo o no tenga polaridad asociada?

Explicaremos cada caso con tranquilidad, pero primero, se presenta el diagrama de funcionamiento general del este bloque, para dar una idea del funcionamiento general e inmediatamente a continuación, pasamos a detallar el funcionamiento de cada sub-bloque de éste, de forma que quede claro todo el comportamiento deseado para la creación de segmentos en el texto.

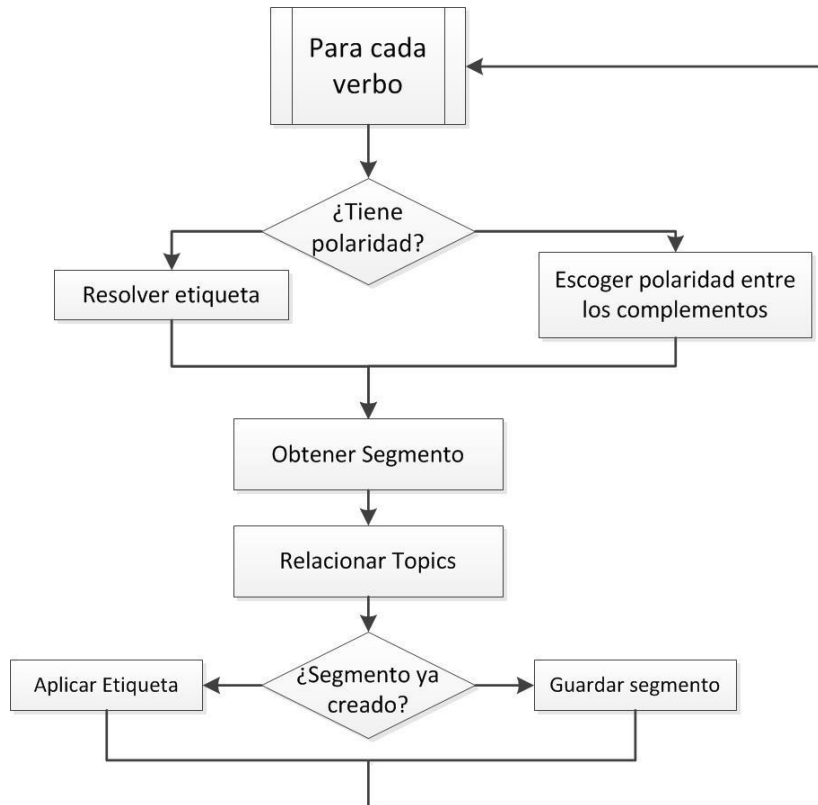


Ilustración 17: Algoritmo creador de segmentos

En la ilustración se muestra el funcionamiento del sistema a la hora de detectar segmentos. Existen básicamente dos puntos de decisión que varían la forma en que funciona el software. Primeramente el software comprueba si existe una **etiqueta** de sentimiento asociada al **verbo**, en caso afirmativo se toma como sentimiento predominante, en caso contrario se **examinan** las **relaciones encontradas** del mismo (complementos directo e indirectos, complementos de modo etc...) y se elige el sentimiento predominante entre ellos. En ambos casos, si la etiqueta necesita ser “resuelta”, es decir, es de tipo variable, se resuelve a su valor final. Posteriormente se elige la porción de texto que representa el segmento (**Obtener segmento**) y se establecen las relaciones entre las entidades y conceptos encontrados y el sentimiento elegido como predominante. Por último, si el segmento encontrado no se había descubierto anteriormente se almacena. En caso contrario, su sentimiento se apila junto con el ya creado, de forma que posteriormente se tome la media como valor del segmento. Este último sería el caso de varios verbos en el mismo segmento.

3.7.3.1. Polaridad entre los complementos

Si el verbo no posee sentimiento, se examinarán sus relaciones, esto es, sujeto, complemento directo, complemento de modo... Entre todas ellas se ha establecido

una prioridad configurable, que nos permite elegir, entre todas ellas, la relación con sentimiento más valorada como representante.

| Complemento | Prioridad | Ejemplo |
|------------------------------|-----------|---|
| Complemento Directo | 6 | En Granada, vi <u>un paisaje espectacular.</u> |
| Complemento Indirecto | 5 | Otra vez he hablado <u>a ese idiota.</u> |
| Atributo | 4 | Este libro es <u>muy malo.</u> |
| Predicativo | 3 | Los niños llegaron <u>cansados.</u> |
| Complemento de Modo | 2 | He esperado <u>cómodamente</u> en la sala de espera. |
| Complemento | 1 | [<i>Otros complementos genéricos encontrados</i>] |
| Sujeto | -1 | <u>La piña rica</u> se ha acabado |

Tabla 11: Prioridad de los complementos

En la tabla que acabamos de representar (Tabla 11: Prioridad de los complementos), se describe la configuración que, tras estudiar cada caso por medio de distintos ejemplos, hemos considerado la más adecuada. El valor de prioridad “-1” marca complementos que son eliminados de la selección, es decir, con “-1” como valor de prioridad, jamás se escogerán como representantes del sentimiento.

La búsqueda a través de las relaciones del verbo se realiza mediante un algoritmo de búsqueda con prioridad, similar explicado en la sección Búsqueda y para la selección de reglas.

En la tabla, se puede observar que la relación de sujeto, posee una prioridad de “-1”. Esta decisión se debe principalmente, a que los operadores variables (!N, =P, ...) están pensados para que resuelvan su valor mediante polaridades encontradas entre los complementos verbales que describen la acción del verbo, más que con “quien la realiza”, que se expresa en el sujeto de la oración, y como veremos en breve, esta misma prioridad se usará para resolver las etiquetas de verbo. Además, el usar el sujeto para encontrar el sentimiento del segmento no encaja con la forma de evaluar basada en el verbo que hemos propuesto. Por ejemplo, vemos los siguientes ejemplos para el verbo “ser”, que no tiene por defecto un sentimiento definido:

- A) “La **buena** de Lucía **es** bastante **tonta**”
- B) “La **buena** de Lucía **es** muy alta”

Vemos que, la palabra con polaridad “buena” no queremos que influya en el sentimiento global detectado, ya que es un adjetivo calificativo que describe al sujeto, pero lo que realmente nos importa es la acción del verbo, es decir, en el caso A, sería un segmento negativo ya que el atributo lo es “es bastante tonta”, mientras que en el caso B no tendríamos sentimiento, por que el sujeto no nos influye.

3.7.3.2. Resolución de etiquetas

Una vez tenemos seleccionado el sentimiento dominante del verbo, éste puede ser una etiqueta variable (de tipo !N, =P, =N, ...) , que necesite ser resuelta en función del resto del segmento. Para ello debemos aplicar las reglas de funcionamiento de cada operador, descrita en la sección del Diccionario (apartado 3.5).

Como acabamos de adelantar en el apartado anterior, su polaridad se resuelve usando la polaridad de la relación prioritaria (Tabla 11). De esta forma los operadores funcionarían con el fin que se diseñaron, para modelar sentimientos que dependen de los complementos.

Por ejemplo, si queremos que “encarcelar” sea negativo, “encarcelar a alguien culpable” sea positivo y “encarcelar a alguien inocente” sea negativo, “encarcelar” se etiquetaría con un operador !N , y la resolución de la etiqueta debe de realizarse con el “inocente” o el “culpable” que no es otra cosa que el complemento sobre el que actúa el verbo, tal como hemos descrito.

3.7.3.3. Agregación de etiquetas

La asignación de polaridad a un complemento completo se realiza enfrentando las etiquetas modificadoras contenidas en él una a una hacia la derecha, realizando medias aritméticas entre ellas y por último resolviendo las etiquetas especiales, si las hubiera, tal y como se ilustra en la siguiente ilustración:

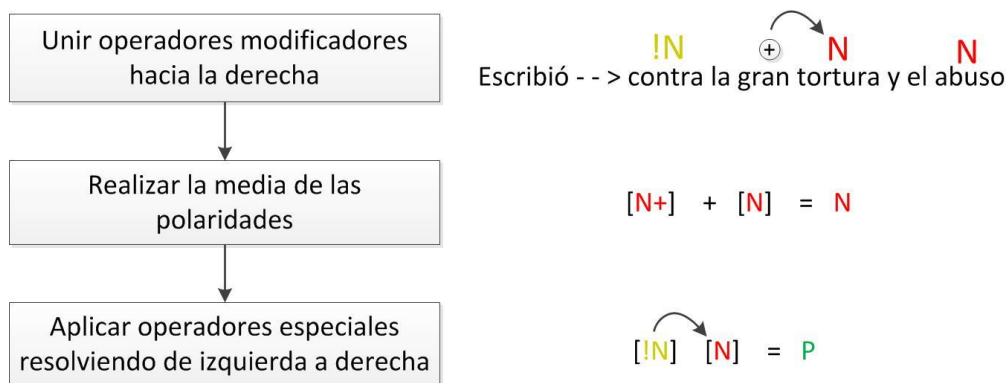


Ilustración 18: Agregación de etiquetas

El algoritmo desarrollado en la ilustración, describe de forma genérica como se agregan las distintas etiquetas que podemos encontrar a un mismo nivel en el árbol morfosintáctico. Se representa un ejemplo completo, aunque generalmente las

uniones a un mismo nivel del árbol no contarán con tantas etiquetas, ni suelen contener etiquetas especiales, ya que éstas están pensadas para ser utilizadas con los verbos principalmente.

3.7.3.4. Obtención del segmento

Siguiendo con la explicación detallada del funcionamiento explicado en la Ilustración 17, el bloque “Obtener segmento” es el encargado de determinar qué porción del texto representa al segmento. Para este fin, hemos decidido realizar una búsqueda vertical en el árbol desde el verbo que crea el sentimiento del segmento, en busca de la primera porción de texto que tiene sentido completo. Esta unidad es la identificada como “oración”, con la etiqueta “Z” del parser (Tabla 8). Es decir, vamos a dejar esta tarea al simple hecho de buscar qué tokens del árbol morfosintáctico nos han sido etiquetados por el código del motor de procesamiento del lenguaje como oraciones con sentido completo, lo cual básicamente va a terminar descomponiendo la frase en sus distintas oraciones coordinadas.

3.7.3.5. Relación con los Topics

Este sub-bloque se encarga de decidir qué entidades y conceptos tienen relación con el sentimiento calculado para el segmento.

Ejemplo:

- “Me **gusta** mucho mi Samsung. “
 - El segmento creado por “gustar” debería afectar positivamente a la entidad “Samsung”
- “Es **horrible, prefiero** el iPhone.”
 - A la entidad “iPhone”, no debería llegar el sentimiento de “Es horrible”, pero sí el de “preferir”.

La detección de estas relaciones las vamos a basar también en las relaciones existentes en la oración. De manera que, cuando se cree un segmento en un verbo, con un determinado sentimiento, éste afecte a todas las entidades y conceptos que pueda alcanzar a través de las relaciones del verbo.

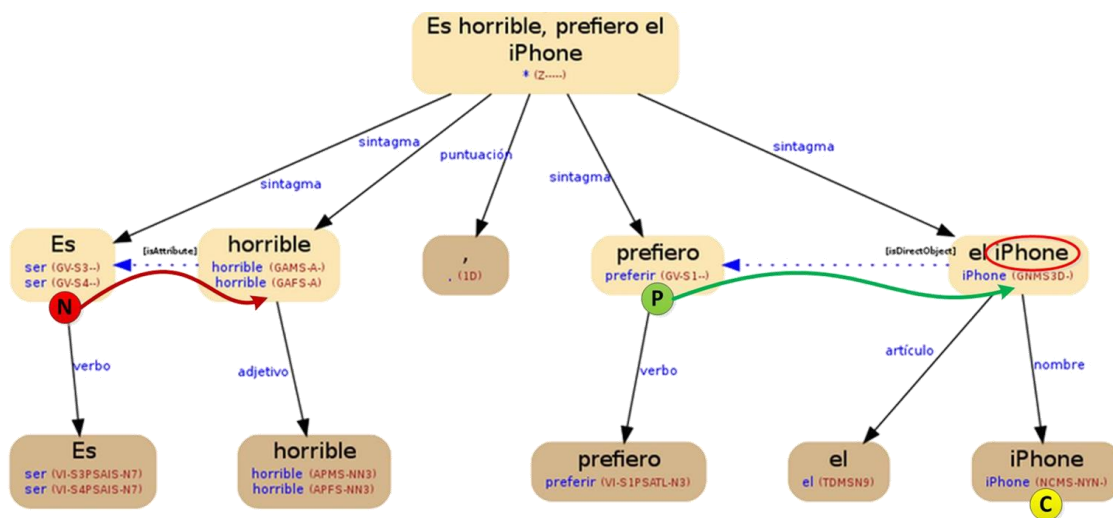


Ilustración 19: Relación entre Topics y Segmentos

3.7.3.6. Oraciones con varios verbos

La última parte del diagrama de flujo expresado en la Ilustración 17 es la encargada de gestionar la creación de segmento cuando éste tiene **varios verbos**.

Cuando la oración posee distintos verbos con sentimiento, puede ocurrir que éstos pertenezcan al mismo segmento. Cuando esto ocurra, los distintos sentimientos encontrados en el segmento, se combinarán a través de una media aritmética de sus valores numéricos dando como resultado la polaridad global del segmento.

Por ejemplo, si tomamos la oración “Camino de **ver** una película que **odio**”, nos encontramos con dos verbos con sentimiento, por un lado el verbo “ver” que es positivo (P) en el contexto de película, y por otro “odiar”, que tiene una polaridad negativa (N). Como resultado del análisis de la oración, se creará un solo segmento, que agrupa los valores de ambos verbos, de forma que, “ver una película que odio” quedaría neutral, o negativo en caso de que “odiar” fuera etiquetado como fuertemente negativo y “ver” sólo como positivo (N+ y P).

3.7.3.7. Oraciones sin verbo

Para terminar este apartado, nos queda sólo fijar el criterio a seguir cuando no exista verbo. En la lengua española a las oraciones sin verbos se las denomina oraciones nominales. Las oraciones nominales son comúnmente usadas en textos tanto coloquiales como formales. En ellos, el verbo se encuentra omitido debido a que existe suficiente contexto para su correcta interpretación [30]. Este tipo de oraciones son muy usadas tanto en las redes sociales para la expresión de opiniones, como en los

titulares de noticias, por lo que resulta importante tenerlas en cuenta para el diseño del sistema. Algunos ejemplos: “Nueva **derrota** del Real Madrid”, “¡Qué **pesadez** de clase!”, “Mallorca: **buenas** playas, bares **caros**”.

En este tipo de oraciones nos resulta imposible guiarnos por un criterio de prioridad como el que hemos definido para el resto del diseño, y es muy difícil definir un criterio lingüístico que seguir, ya que en ellas no serán tan claras las relaciones (o incluso inexistentes).

Por tanto, el criterio escogido en esta sección será utilizar el árbol morfosintáctico como guía para relacionar los distintos sentimientos encontrados en el texto, así como crear segmentos y relacionarlos con las entidades y conceptos del mismo.

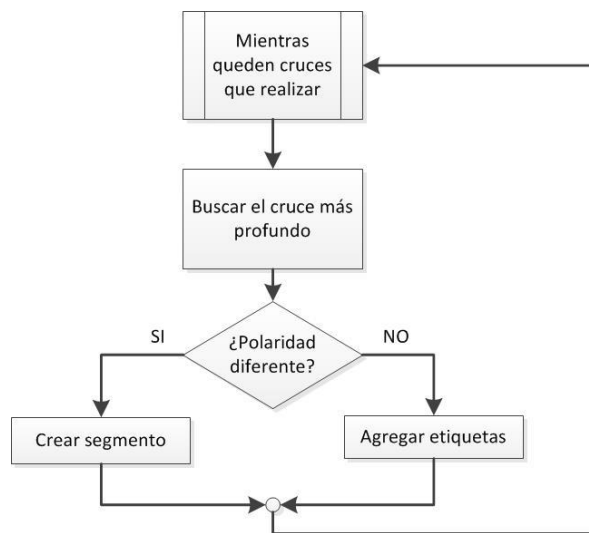


Ilustración 20: Algoritmo de cruce

El algoritmo usado en este caso se representa en la Ilustración 21. Como se puede leer en él, el objetivo es ir realizando la agregación de las etiquetas del texto, con un orden dado por su profundidad en el árbol. De forma que, de todos los “cruces” que se puedan realizar entre las etiquetas del texto se realice siempre el más profundo. Entendemos por cruce el primer nodo del árbol en el que se encontrarían las etiquetas si éstas se “propagaran” hacia arriba por las ramas de éste. En la siguiente ilustración se presenta un ejemplo de cruce en el que se crea un segmento.

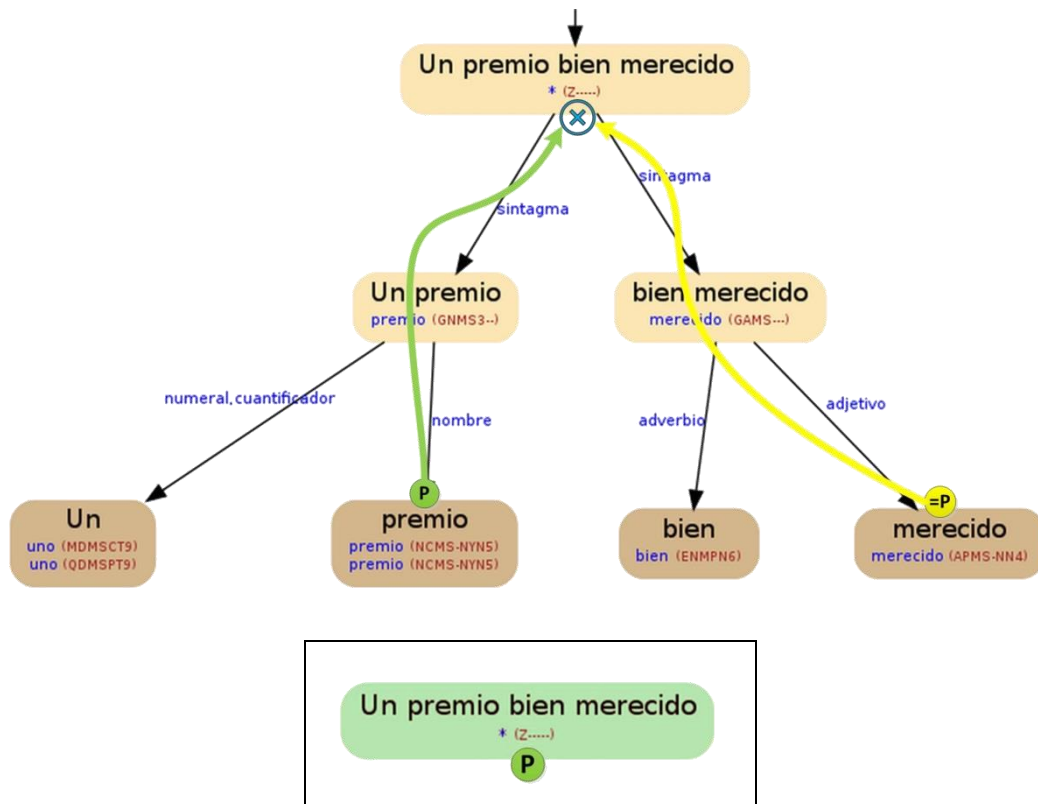


Ilustración 21: Ejemplo de segmento creado por cruce

En el caso que se acaba de exponer, tras la agregación en el primer nodo del árbol, se crea un segmento, debido a que no se encuentran más etiquetas con las que realizar cruces.

El criterio completo para crear un segmento se representa en el siguiente ejemplo. Con el objetivo de mantener la granularidad del sentimiento analizado, cuando se encuentren etiquetas de **polaridades opuestas**, éstas generarán dos segmentos diferentes, de forma que mantengamos las polaridades opuestas, y podamos mantener tanto los aspectos positivos como los negativos encontrados en el texto, en lugar de agregarlos en un solo segmento de polaridad neutral, en el que perderíamos información.

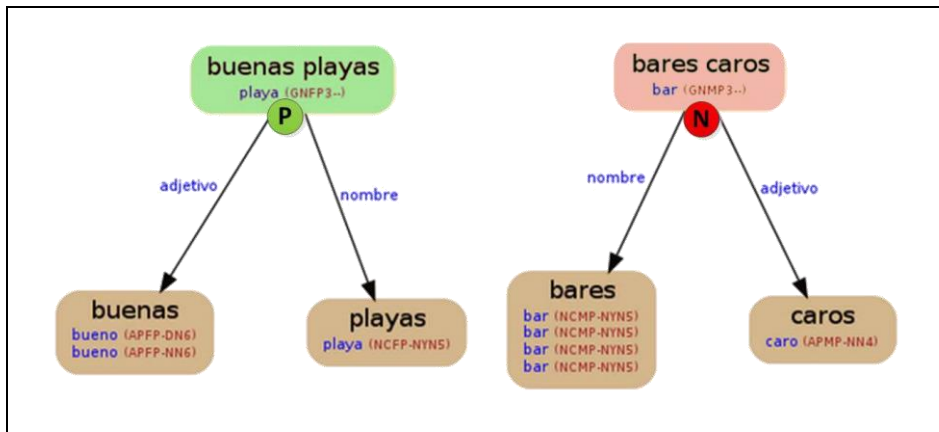
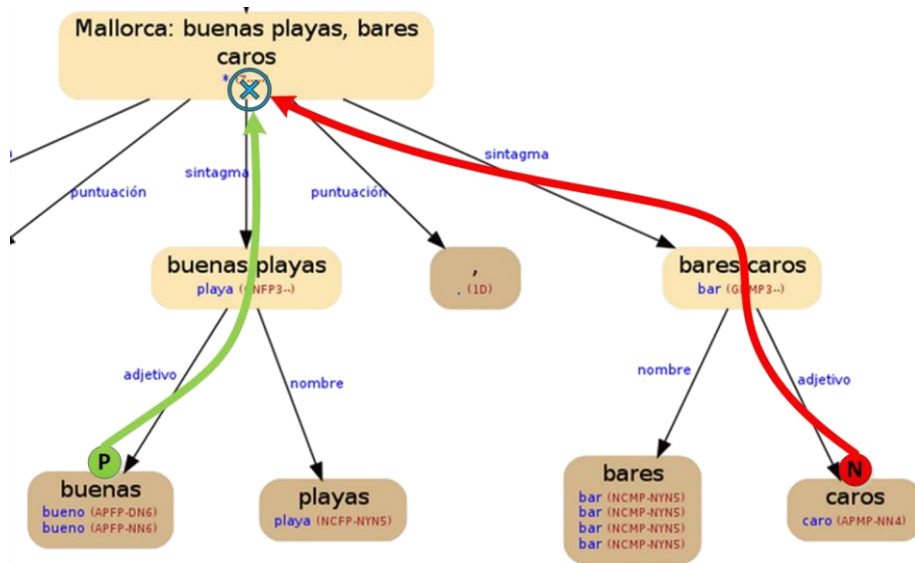


Ilustración 22: Ejemplo de segmento creado por cruce con desacuerdo

En este caso los conceptos y entidades se relacionarían con los segmentos encontrados de una forma más sencilla. Simplemente consideraremos que una entidad o concepto posee un determinado sentimiento si éste se encuentra en dicho segmento. La simplicidad de este criterio se debe a que en este caso no disponemos de tanta información como en las oraciones verbales, y a que éstas suelen ser más sencillas en cuanto a estructura se refiere, por lo que, en principio éste parece un buen criterio.

4. Implementación del sistema

En este apartado aclararemos algunos detalles más específicos, relacionados con la implementación concreta que hemos realizado del sistema descrito en el apartado anterior.

4.1. Implementación del diccionario

El diccionario del sistema es un fichero de texto sin formato que contiene un gran número de reglas que modelan el análisis de sentimiento, por lo que para su desarrollo no se necesita de ninguna herramienta externa ni lenguaje de programación.

En concreto, para este proyecto se ha desarrollado un modelado de sentimiento general, que intenta analizar estado emocional del autor del texto. Este modelo se ha realizado mediante un análisis manual de ejemplos, recogidos principalmente de la red social Twitter, e intentando a través de ellos, escribir las reglas necesarias para detectar correctamente el mayor número de sentimientos de forma correcta.

Al ser un modelo general, su cobertura intenta ser tan ambiciosamente grande, que es probable que su precisión sea demasiado baja, debido al compromiso existente entre cobertura y precisión. Aun así, es tarea posterior realizar distintas evaluaciones del modelo y realizar, en caso de que fuera necesario, un ajuste más fino del mismo, eliminando o modificando las reglas del diccionario que provoquen demasiados análisis incorrectos.

4.2. Implementación del motor

Para la implementación del motor, puesto que se quería que éste fuera creado como un nuevo módulo del motor de PLN (núcleo del sistema), hemos adoptado por continuidad C++ como lenguaje de programación.

C++ es un lenguaje de propósito general, muy potente y flexible, que goza de gran popularidad en la actualidad. Es un lenguaje compilado, lo cual lo hace una alternativa mucho más rápida en ejecución frente a lenguajes interpretados, como podría ser, por ejemplo, PHP. Implementa un paradigma orientado a objetos que da al mismo la capacidad de desarrollar grandes y complejos programas de forma estructurada mediante elementos más simples, que facilitan el mantenimiento y la modificación posterior de los programas, sin perder la gran potencia del mismo [31]. Las características del sistema, un programa muy complejo y de gran tamaño, junto con sus necesidades de eficiencia, hacen a C++ un lenguaje idóneo para el mismo, razones por las que el sistema completo se encuentra implementado en C++.

4.2.1. Estructura del programa

El código del programa que implementa el módulo de análisis de sentimiento se ha estructurado en varios ficheros fuente. Como es habitual en lenguajes de programación tipo C, por un lado se encuentran los ficheros de cabecera (*.h) que contienen las declaraciones de las clases, mientras que los archivos de código (*.cpp) se encargan de la implementación.

Los archivos en los que se ha dividido el programa se describen en la siguiente tabla.

| Fichero | Descripción |
|----------------------------|---|
| SentimentEs.cpp | Implementa la rutina principal del programa (main) |
| Sentiment.cpp | Implementa el módulo de análisis de sentimiento. Es el encargado de heredar las funciones del motor de PLN y utilizarlas, así como gestionar el funcionamiento del módulo (parámetros de entrada, llamada al motor de PLN, ejecución de análisis de sentimiento y salida del programa). |
| Sentiment.h | Archivo cabecera correspondiente al fichero <i>Sentiment.cpp</i> . |
| Sentimentalyzer.cpp | Implementa las funcionalidades correspondientes al motor de sentimiento (Lectura de diccionario, búsqueda de reglas en el mismo, procesamiento de las etiquetas de sentimiento). |
| Sentimentalyzer.h | Fichero de cabecera correspondiente a las declaraciones del fichero <i>Sentimentalyzer.cpp</i> . |

Tabla 12: Estructura de ficheros

4.2.2. Clases implementadas

La gran ventaja de utilizar un lenguaje orientado a objetos, como en nuestro caso es C++, es poder desarrollar de forma independiente, por medio de clases, objetos que modelen los distintos elementos del programa, agrupando en ellos tanto las variables que los representan como la metodología para trabajar con ellos. A continuación se realizará una breve descripción de las clases que se han implementado para el desarrollo del sistema.

Emotion

La clase `Emotion` modela los sentimientos encontrados en el texto. Agrupa los atributos que definen la regla de diccionario que lo crearon, como su etiqueta de polaridad, la forma, el contexto y la categoría gramatical de la misma. Por otro lado, también almacena las características que determinan su posición en el texto: índices de posición y el token del árbol en el que se encuentra. Además, en esta clase agrupamos diversos métodos para trabajar con el `Emotion`: métodos para obtener la etiqueta, invertir su polaridad, realizar diferentes comprobaciones sobre su tipo, así como métodos para obtener su representación en JSON.

SentimentInfo

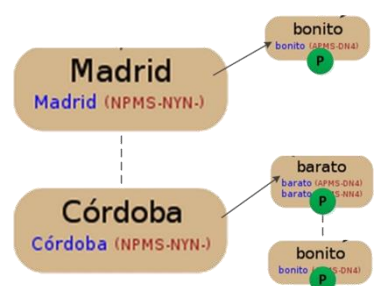
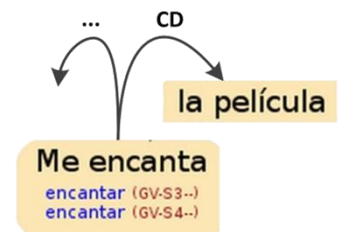
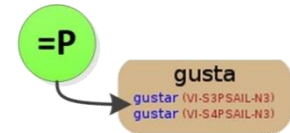
Esta clase forma una estructura de datos en la que almacenaremos diferentes aspectos del texto que estamos analizando, con el fin de simplificar las tareas de análisis. Puesto que nuestro análisis de sentimiento trabaja principalmente con los verbos y sus relaciones semánticas, la clase contendrá exactamente eso: un verbo junto con las relaciones de éste. Así podremos agrupar en ella los elementos que más trabajamos a nivel de código, y agrupar los diferentes métodos que necesitemos implementar.

Segment

`Segment` es la clase que modela el elemento al que varias veces nos hemos referido ya en este texto como un “segmento”. Se trata un fragmento del texto analizado sobre el que se ha detectado un sentimiento. Por tanto, esta clase será modelada con atributos como: el fragmento de texto, sus índices de posición, la polaridad elegida para el fragmento, los diferentes sentimientos “keywords” detectados y usados para generarlo, así como los conceptos o entidades detectadas junto con su polaridad asociada.

EmotionMap

`EmotionMap` es un objeto que agrupa por un lado los conceptos/entidades encontrados, junto con una lista de `Emotions` que, tras el análisis, han sido asignados como `Emotions` que afectan al concepto/entidad en cuestión. El



nombre de “Map” indica que su estructura de datos interna se asemeja a la de un objeto tipo mapa. Un objeto tipo mapa, usados comúnmente en los lenguajes orientados objetos, es un objetos de tipo contenedor asociativo, que agrupa una serie de claves (keys) y para cada una de ellas un valor (mapped value) [32]. Es decir, en nuestra clase, estamos almacenando entidades/conceptos como clave, y para cada una, un vector de `Emotions` que les afectan, a modo de valor mapeado.

Sentimentalyzer

La clase `Sentimentalyzer` representa al analizador de sentimiento en sí. Es decir, es la clase que se vale de todas las que acabamos de desarrollar para implementar el funcionamiento completo descrito en el apartado de diseño del motor. En ella tendremos pues, los métodos que realizan todas las operaciones principales del sistema (carga del diccionario, búsqueda en él, selección de reglas, aplicación de negadores y creación de segmentos), así como una serie de métodos auxiliares que implementan funciones concretas que son utilizadas de forma repetitiva a lo largo del código, como funciones para el escapado de caracteres, para la unión de `Emotions` por medio de sus medias o funciones auxiliares de post procesado.

Sentiment

Por último, la clase `Sentiment` es la clase de más alto nivel del sistema. Esta clase se encarga de implementar las funciones más básicas: leer los parámetros de entrada del programa y llamar al núcleo con la configuración adecuada para que realice los análisis pertinentes, así como llamar al analizador de sentimiento (`Sentimentalyzer`) e imprimir la salida del sistema. A continuación, en la Ilustración 23 se representa un diagrama resumen del uso que hacen las clases las unas de las otras para llevar a cabo el análisis de sentimiento.

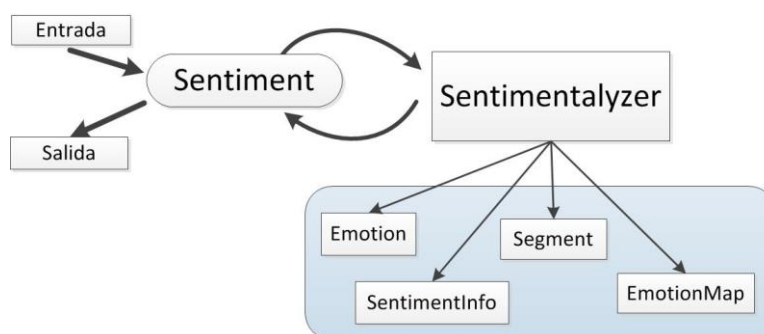


Ilustración 23: Esquema de clases

4.2.3. Implementación

A nivel de código y siguiendo el flujo de funcionamiento habitual del programa, la primera función en ejecutarse es el `main`, la función principal del programa. Ésta se encuentra implementada en el fichero `SentimetES.cpp`. Examinando el fichero, tras las primeras líneas, dedicadas a la inclusión de los ficheros de cabecera de librerías estándar (`stdlib.h`) y de otros ficheros del programa, encontramos la función `main`.

```
#include <stdlib.h>
#include "Sentiment.h"

int main(int argc, char *argv[]) {
    Sentiment s;
    SentimentConfig * config = s.getDefaultConfig();
    Sentimentalyzer * sentimentalyzer;

    /* Lectura de argumentos*/
    if(!s.readArgs(config, argc, argv))
        exit(1);
    /*Inicialización del analizador de sentimientos*/
    sentimentalyzer = new Sentimentalyzer();
    /*Iniciar el análisis de sentimiento*/
    s.setSentimentalyzer(sentimentalyzer);
    s.run(config);
}
```

Lo primero que hace el `main` es declarar un objeto de tipo `Sentiment`. Este es el objeto principal del programa, como hemos explicado en la sección previa. Usando los métodos de este objeto, se validan los argumentos de entrada del programa, a través del método `readArgs()`, para después, si todo es correcto, inicializar un nuevo objeto de tipo “analizador de sentimiento” (`Sentimentalyzer`) y asignarlo como el analizador del objeto `Sentiment`, tal y como explicamos en la Ilustración 23. Esta asignación se realiza a través de la sentencia `setSentimentalyzer()`. Finalmente, el analizador de sentimientos se pone en marcha a través de su método `run()`.

Si profundizamos en el bloque de código que se ejecuta cuando llamamos al método `run()`, encontraremos que éste, entre otras sentencias, se encarga principalmente de ejecutar una llamada al método `action()`, cuyo código hemos incluido abajo.

Este código es muy ilustrativo, ya que en él se realizan una a una las llamadas al código del núcleo para realizar las diferentes etapas de análisis que son necesarias como paso previo al análisis de sentimiento. La primera sentencia, que llama al método `open()`, es la encargada de cargar en memoria el contenido del diccionario, de forma que todas sus reglas estén disponibles para las oraciones a analizar. En las siguientes sentencias, podemos ver cómo se ejecutan las etapas de desambiguación, análisis sintáctico, el

detector de relaciones y el detector de topics (entidades y conceptos) para dar paso en último lugar al análisis de sentimiento, mediante llamada al método `getSentiment()` que explicaremos a continuación.

```
void Sentiment::action(Config* config, ostream &out) {  
  
    /* Carga del diccionario de sentimiento*/  
    SentimentDocument *doc = sentimentalyzer->open(config);  
    // Desambiguación  
    sentimentalyzer->analyzer->disambiguationRules(doc);  
    // Desambiguación y detección heurística  
    sentimentalyzer->analyzer->nerAnalyses(doc);  
    // Análisis sintáctico  
    sentimentalyzer->analyzer->syntacticAnalysis(doc);  
    // Búsqueda de posibles relaciones  
    sentimentalyzer->analyzer->relationDiscovery(doc);  
    // Detección de entidades/conceptos  
    sentimentalyzer->analyzer->detectTopics(doc);  
    // Post-procesado  
    sentimentalyzer->analyzer->lastAdjustments(doc);  
    // Asignar relaciones  
    sentimentalyzer->analyzer->detectRelations(doc);  
    // Obtener sentimiento  
    sentimentalyzer->getSentiment(doc);  
    // Imprimir el resultado final  
    printReportToJSON(doc, out);  
  
} // action
```

Cuando el análisis acaba, se utiliza el método `printReportToJSON()`, encargado de realizar la impresión del resultado del análisis en formato JSON, terminando así la ejecución del método.

Si analizamos un poco más en detalle el método `getSentiment()`, veremos que éste se podría considerar como el método principal de nuestro sistema, ya que es el encargado de utilizar las distintas partes implementadas y realizar el análisis oración a oración, tal y como hemos descrito en el apartado de diseño.

```

void Sentimentalyzer::getSentiment(SentimentDocument *doc){

    EmotionMap  listOfTopics;           /* EmotionMap del texto*/
    vector<Emotion *> auxList;          /* Lista de Emotions */
    vector<SentimentInfo *> finalSentiments; /* Verbo y relaciones */
    vector<Topic *> topicsList;        /* Lista de entidades/conceptos */

    /*Análisis frase a frase*/
    for(size_t i=0, iEnd=doc->currentFlatSentences.size(); i<iEnd; i++){

        /* Etiquetar sentimiento */
        searchSentiment(doc->currentFlatSentences[i],i,doc);

        /* Aplicar negadores con cada token */
        long tam=doc->currentFlatSentences[i]->children.size();
        for (long j=0; j<tam; j++){
            applyNeg(doc->currentFlatSentences[i]->children[j]);
        }

        /*Rellenar lista de Emotions, SentimentInfo y Topics*/
        for (long j=0; j<tam; j++){
            Token * tok = doc->currentFlatSentences[i]->children[j];

            if(tok->hasEmotion())
                auxList.push_back(tok->Emotion);           /*Almacenar Emotion*/
            if(tok->isVerb()){
                SentimentInfo *s = new SentimentInfo(tok, tok->tree_relations);
                finalSentiments.push_back(s);              /* Almacenar verbos */
            }else if(tok->isConcept()){
                topicsList.push_back(tok->Concept); /*Añadir las entidades*/
            }else if(tok->isEntity()){
                topicsList.push_back(tok->Entity); /*Añadir los conceptos*/
            }
        }

        /*Crear Segmentos, análisis por verbos*/
        bool bc = createSegments(finalSentiments, topicsList,
                                auxList, doc, &listOfTopics);
        /*Último caso, Suma de etiquetas por cruce*/
        if(!bc)
            crossJoin(auxList, topicsList, &listOfTopics, doc);
    }
} //getSentiment

```

Tras las primeras declaraciones de variables, encontramos el bucle encargado de iterar sobre cada oración. Con cada oración, el método `getSetiment()` realiza básicamente cuatro operaciones, que se corresponden con los cuatro bloques de código que se pueden intuir al leer el código de la función.

Primeramente se realiza el etiquetado de sentimiento, operación de la que se encarga la función `searchSentiment()`.

A continuación, token a token, se aplican los negadores, en caso de que los hubiera, por medio del método diseñado para tal fin, `applyNeg()`.

En tercer lugar, se ejecuta el código pertinente que rellena las variables declaradas en el método, esto es, la lista de `Emotions` de la oración, las entidades, los conceptos y los verbos con sentimiento, almacenados en forma de objeto `SentimentInfo`. La razón de recopilar tantas variables a las que ya se puede acceder directamente a través del árbol morfosintáctico, no es otra que mejorar la eficiencia del sistema. El resto de funciones trabajarán con ellas y de esta forma podrán acceder directamente en lugar de realizar costosas operaciones de búsqueda y lectura del árbol, que en muchos casos significarían la implementación de métodos recursivos, con el coste computacional que ello conlleva.

En último lugar se ejecuta la función encargada de analizar y crear los segmentos en el texto, la función `createSegments()`. En caso de que ésta no cree ningún segmento, caso habitual cuando no existen verbos, se llamará a la función `crossJoin()`, encargada de la creación de segmentos mediante el algoritmo basado en el árbol morfosintáctico que ya comentamos en la sección “Oraciones sin verbo” del apartado de diseño del sistema.

5. Evaluación

5.1. Introducción

Tras el diseño e implementación del sistema, nuestro objetivo es realizar diferentes pruebas de forma que podamos comprobar su correcto funcionamiento, así como evaluar su efectividad.

Además de intentar realizar una medida numérica de la calidad del mismo, realizaremos un análisis de las respuestas y los errores cometidos, de forma que podamos ilustrar el funcionamiento del sistema de manera práctica y al mismo tiempo analizar los fallos cometidos, buscando su origen y dando si es posible, una solución a los mismos.

5.2. Evaluación del sentimiento a nivel global

Primeramente, hemos realizado una evaluación del sentimiento a nivel global. El objetivo es conocer hasta qué punto es capaz nuestro sistema de evaluar correctamente la polaridad global asociada a un determinado texto. Recordemos que con polaridad global nos referimos a una medida del estado emocional en el que se encontraba el autor del texto cuando lo escribió.

Para llevar a cabo esta evaluación es necesario utilizar un corpus de textos etiquetados manualmente por humanos, los cuales serán evaluados por el sistema, comparando posteriormente las etiquetas manuales con el cálculo realizado por la máquina.

También será necesario preparar un diccionario de reglas para usar con nuestro sistema.

5.2.1. Corpus de pruebas

El corpus de pruebas que hemos decidido emplear para evaluar la polaridad global del sistema se corresponde con el corpus utilizado como entrenamiento en el workshop de análisis de Sentimiento conocido como el TASS (Taller de Análisis de Sentimientos en la SEPLN), el cual se celebra actualmente en la conferencia anual del SEPLN [33]. Concretamente, hemos empleado el corpus correspondiente al workshop celebrado en el año 2013.

La elección del mismo se debe por un lado a su disponibilidad y a que los criterios de etiquetado se corresponden bastante bien con los propuestos por nuestro diseño, pudiendo realizar una evaluación más adecuada. Además, otra gran ventaja es

disponer de los resultados que obtuvieron los diferentes grupos de investigación que participaron en el workshop. De forma que podamos comparar los resultados de nuestro sistema con los obtenidos por diferentes implementaciones, buscando obtener conclusiones que nos lleven a la mejora del sistema.

5.2.2. Contenido del corpus

El corpus de entrenamiento se compone de 7220 mensajes de la red social Twitter escritos en español por diferentes personalidades importantes de diferentes áreas como política, economía, cultura, o comunicación. Aunque todos ellos estén escritos en español, es importante resaltar que debido a las diferentes nacionalidades de los autores, en ellos se utilizan distintas variantes del español.

Para cada mensaje se ha etiquetado la polaridad del mismo siguiendo la misma nomenclatura que hemos tomado en nuestro sistema, es decir, mediante cinco niveles de polaridad: Muy positivo (P+), positivo (P), neutral (NEU), negativo (N) o muy negativo (N+), más una etiqueta que indica que el texto no posee sentimiento (NONE).

No está de más también comentar que el etiquetado del corpus se ha realizado de forma semiautomática, es decir, ha sido etiquetado mediante la aplicación de un modelo de aprendizaje automático y posteriormente se ha realizado una revisión humana de las etiquetas asignadas a cada texto [34].

```
<tweet>
  <tweetid>000000000</tweetid>
  <user>usuario0</user>
  <content><![CDATA['Conozco a alguien q es adicto al drama!
                Ja ja ja te suena d algo!]]></content>
  <date>2011-12-02T02:59:03</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>P+</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>entretenimiento</topic>
  </topics>
</tweet>
```

Ejemplo 12: Contenido del corpus

En el Ejemplo 12 se puede observar toda la información que contiene el corpus para cada tweet, así como su estructura de datos.

5.2.3. Preparación del diccionario

El diccionario para la realización de esta prueba se ha preparado atendiendo al escenario de la misma. Por un lado se tendrá en cuenta que el dominio de la misma es general, es decir, no tiene una temática concreta, y por otro lado al ser un análisis de mensajes de Twitter, se ha tenido en cuenta el lenguaje habitualmente usado en éste (uso de hashtags, emoticonos, abreviaturas, ...).

Para el desarrollo del mismo se han reutilizado recursos lingüísticos que se tenían disponibles de trabajos anteriores y, posteriormente, a través del análisis de ejemplos concretos, se han ido añadiendo o modificando reglas del diccionario.

El resultado es un diccionario de dominio general con unas **3000 entradas**, en el que mediante la creación de unas 200 macros, se han construido aproximadamente **11200 reglas** diferentes de etiquetado de polaridad.

5.2.4. Batería de pruebas

Para realizar la evaluación de todo el corpus se ha escrito un pequeño script usando PHP, cuyo desarrollo queda fuera de los objetivos del proyecto.

El script realiza una batería de pruebas, evalúa uno a uno los textos del corpus usando el sistema desarrollado y posteriormente clasifica cada resultado comparándolo con el etiquetado humano. Esta clasificación se ha realizado de la siguiente forma:

- **Aciertos.**
Resultados idénticos al etiquetado humano.
- **Errores.**
Resultados diferentes al etiquetado humano, los cuales a su vez se han clasificado en:
 - **Errores leves**
Son errores debidos a fallos en el cálculo de la intensidad. Por ejemplo, un etiquetado de P+ sobre un texto P o un etiquetado N sobre uno N+. También se clasifican en esta categoría cuando el sistema etiqueta NONE frente a NEU y viceversa.

- **Falsos negativos**
Se corresponden con los casos no detectados, es decir, cuando el texto posee sentimiento, ya sea positivo o negativo, y el sistema lo ha etiquetado como NONE o NEU.
- **Falsos positivos**
Este caso se corresponde con el contrario del anterior. Los falsos positivos son aquellos casos en los que los textos a los que hemos etiquetado polaridad no tenían (NONE o NEU) en el etiquetado manual.
- **Cambios de polaridad**
En último lugar, los cambios de polaridad se corresponden con el tipo de error más grave. Éste se da cuando el sistema asigna una polaridad opuesta a la debida, por ejemplo N en lugar de P+.

Los resultados de la prueba se representarán como un porcentaje para cada uno de los casos anteriores respecto del número total de pruebas realizadas. El porcentaje de aciertos se suele conocer como “precisión del sistema”, y es la medida más comúnmente usada para la evaluación de la calidad de un sistema de este tipo.

$$\text{Precisión}(\%) = \frac{N(\text{etiquetados correctos})}{N(\text{pruebas realizadas})} \cdot 100$$

Ilustración 24: Precisión del sistema

Con el fin de analizar más profundamente los resultados de las pruebas, éstos se representarán también mediante la matriz de confusión obtenida en la evaluación. Una matriz de confusión es una herramienta usada ampliamente para la evaluación de los sistemas de aprendizaje supervisado [35] en la que se clasifican los resultados de cada prueba mediante una tabla bidimensional a través del par “valor obtenido” frente a “valor deseado”, que en nuestro caso se traduce por el par “Etiquetado del sistema” frente a “Etiquetado humano”. Con ella se puede analizar de una forma sencilla el funcionamiento del sistema, encontrando defectos en su comportamiento, así como formas de mejorarlo.

5.2.5. Resultados de la prueba

Terminada la ejecución de las pruebas, pasamos a analizar los resultados obtenidos.

A continuación se presenta el resultado general de la prueba, en el que se ha obtenido un porcentaje de aciertos total del 42.73%.

| Resultados: | |
|------------------|---------------|
| Número de textos | 7220 |
| Aciertos | 3085 [42.73%] |
| Errores | 4135 [57.27%] |

Tabla 13: Resultados generales de la evaluación

En principio, este resultado no parece demasiado bueno, sin embargo, antes de juzgar el sistema, analicemos también la clasificación de los errores cometidos, la cual se representa en la siguiente tabla.

| Errores: | |
|----------------------|-------------------------|
| Fallos leves | 1547 [21.43% del total] |
| Falsos negativos | 1320 [18.28% del total] |
| Falsos positivos | 740 [10.25% del total] |
| Cambios de polaridad | 526 [7.29% del total] |

Tabla 14: Clasificación de errores

Como se aprecia, las pruebas revelan que un alto porcentaje de los errores cometidos son leves (errores en el cálculo de la intensidad), siendo éstos el resultado dado por el sistema en un 21.43% del total de las pruebas. Estos errores, como su propio nombre indica, no son demasiado importantes, pues realmente si se medita, es muy complejo determinar incluso para un humano la intensidad de una opinión, aún más en un tweet, cuya longitud es tan limitada. Etiquetar un texto como “positivo” frente a “muy positivo” es más bien, algo subjetivo que depende fuertemente de la persona que lo etiqueta.

Por otro lado, tenemos una proporción de falsos negativos (casos no detectados) del 18.28% y una proporción del 10.25% de falsos positivos, lo cual no parece a priori un valor demasiado alto. Los falsos negativos normalmente se pueden solucionar escribiendo nuevas reglas en el diccionario, mientras que los falsos positivos se corrigen haciendo las reglas existentes en el modelo más específicas mediante métodos de desambiguación (contextos, categoría gramatical, etc.).

Por último, el porcentaje de errores por cambio de polaridad no llega al 8% de los casos, lo cual parece un buen resultado. Este tipo de errores son los más graves que puede cometer el sistema y es importante mantenerlos controlados siempre como un pequeño porcentaje de los casos.

Aprovechando que disponemos de los resultados de estas pruebas para los diferentes grupos que participaron en el TASS 2013, podemos realizar un pequeño análisis comparativo. Los resultados obtenidos en precisión, para los diferentes grupos participantes en el evento se mueven en un rango entre el 61.6% y el 12.6%, con una

media de precisión del 43.3% [36]. Con este último dato se puede decir que nuestro sistema se encuentra aproximadamente en la media.

Durante el workshop, con el fin de desarrollar una evaluación más detallada, se realizó también una evaluación de los sistemas basada únicamente en tres niveles de polaridad (positivo, neutral o negativo). En ese caso la precisión obtenida por los diferentes equipos se mantuvo en un rango entre el 68.6% y el 23.0%, con una media del 53.0% de aciertos [36].

Para nuestro sistema, una evaluación basada únicamente en tres niveles de polaridad, sería equivalente a contabilizar los errores leves como si fueran aciertos, por lo que para este caso, nuestro resultado sería el mostrado en la siguiente tabla, alcanzando ahora un 64.15 % de aciertos, lo cual es un valor bastante alto.

| Resultados: | |
|------------------|---------------|
| Número de textos | 7220 |
| Aciertos | 4632 [64.15%] |
| Errores | 2588 [35.85%] |

Tabla 15: Resultados para el sistema de tres niveles

Con este resultado se podría decir que la precisión de nuestro sistema se por encima de la media de los participantes del TASS 2013, incluso, nuestro sistema se acerca bastante al máximo conseguido por los mismos (68.6%).

En la siguiente ilustración incluimos la precisión (en tanto por uno) obtenida para todos los modelos enviados por los diferentes participantes del TASS 2013, tanto para el sistema de 5 niveles de polaridades como al sistema de 3 niveles de polaridad.

| Evaluación de 5 niveles | |
|---------------------------------------|-----------|
| Run Id | Precisión |
| DLSI-UA-pol-dlsiua3-3-5l.txt | 0.616 |
| Elhuyar-TASS2013_Elhuyar_run1 | 0.601 |
| Elhuyar-TASS2013_Elhuyar_run2 | 0.599 |
| DLSI-UA-pol-dlsiua3-2-5l.txt | 0.596 |
| UPV_ELiRF_task1_run2.txt | 0.576 |
| UPV_ELiRF_task1_run1.txt | 0.574 |
| UPV_ELiRF_task1_run3.txt | 0.573 |
| CITIUS-task1_CITIUS_1.txt | 0.558 |
| lys_global_sentiment_task_6c | 0.553 |
| DLSI-UA-pol-dlsiua3-1-5l.txt | 0.552 |
| CITIUS-task1_CITIUS_2.txt | 0.541 |
| lys_global_sentiment_task_6c_wui | 0.533 |
| JRC-tassTrain-base-DICT-5way.tsv | 0.519 |
| JRC-tassTrain-lemmaStop-SVM-5way.tsv | 0.515 |
| JRC-tassTrain-lemmaStop-DICT-5way.tsv | 0.507 |
| JRC-tassTrain-base-SVM-5way.tsv | 0.505 |
| JRC-tassTrain-lemma-SVM-5way.tsv | 0.504 |
| JRC-tassTrain-lemma-DICT-5way.tsv | 0.497 |
| JRC-tassTrain-lemmaStop-4CLS-5way.tsv | 0.481 |
| JRC-tassTrain-base-4CLS-5way.tsv | 0.477 |
| JRC-tassTrain-lemma-4CLS-5way.tsv | 0.477 |
| ITA_ResultadosAnálisisOpiniónAlg | 0.439 |
| LSI_UNED_2_TASK1_RUN_09 | 0.402 |
| LSI_UNED_2_TASK1_RUN_14 | 0.402 |
| LSI_UNED_2_TASK1_RUN_04 | 0.398 |
| LSI_UNED_2_TASK1_RUN_11 | 0.398 |
| LSI_UNED_2_TASK1_RUN_15 | 0.396 |
| LSI_UNED_2_TASK1_RUN_05 | 0.395 |
| LSI_UNED_2_TASK1_RUN_07 | 0.395 |
| LSI_UNED_2_TASK1_RUN_10 | 0.395 |
| LSI_UNED_2_TASK1_RUN_02 | 0.393 |
| UNED-JRM-task1-run2.txt | 0.393 |
| LSI_UNED_2_TASK1_RUN_03 | 0.391 |
| LSI_UNED_2_TASK1_RUN_08 | 0.391 |
| LSI_UNED_2_TASK1_RUN_12 | 0.386 |
| LSI_UNED_2_TASK1_RUN_13 | 0.386 |
| LSI_UNED_2_TASK1_RUN_06 | 0.359 |
| LSI_UNED_2_TASK1_RUN_01 | 0.354 |
| TECNALIA-UNED.txt | 0.348 |
| ETH-task1-Warriner.txt | 0.328 |
| sinai_emml_task1_6classes.txt | 0.314 |
| ETH-task1-OptT1.txt | 0.249 |
| ETH-task1-OptT2.txt | 0.244 |
| sinai_cesa-task1_raw.tsv | 0.135 |
| sinai_cesa-task1_normalized.tsv | 0.131 |
| UNED-JRM-task1.txt | 0.126 |

| Evaluación de 3 niveles | |
|---|-----------|
| Run Id | Precisión |
| Elhuyar-TASS2013_Elhuyar_run1 | 0.686 |
| Elhuyar-TASS2013_Elhuyar_run2 | 0.684 |
| UPV_ELiRF_task1_run2.txt | 0.674 |
| UPV_ELiRF_task1_run3.txt | 0.674 |
| UPV_ELiRF_task1_run1.txt | 0.672 |
| CITIUS-task1_CITIUS_1.txt | 0.668 |
| DLSI-UA-pol-dlsiua3-3-5l.txt | 0.663 |
| lys_global_sentiment_task_6c | 0.657 |
| lys_global_sentiment_task_6c_wui | 0.647 |
| DLSI-UA-pol-dlsiua3-2-5l.txt | 0.640 |
| CITIUS-task1_CITIUS_2.txt | 0.622 |
| DLSI-UA-pol-dlsiua3-1-5l.txt | 0.620 |
| JRC-tassTrain-base-DICT-3way.tsv | 0.612 |
| JRC-tassTrain-lemmaStop-SVM-3way.tsv | 0.608 |
| JRC-tassTrain-lemmaStop-DICT-3way.tsv | 0.607 |
| JRC-tassTrain-lemma-DICT-3way.tsv | 0.599 |
| JRC-tassTrain-lemma-SVM-3way.tsv | 0.599 |
| JRC-tassTrain-base-SVM-3way.tsv | 0.597 |
| JRC-semevaltassTrain-base-DICT-3way.tsv | 0.590 |
| JRC-semevaltassTrain-base-SVM-3way.tsv | 0.585 |
| JRC-tassTrain-lemmaStop-4CLS-3way.tsv | 0.582 |
| ITA_ResultadosAnálisisOpiniónAlg | 0.543 |
| TECNALIA-UNED.txt | 0.496 |
| UNED-JRM-task1-run2.txt | 0.496 |
| LSI_UNED_2_TASK1_RUN_06 | 0.479 |
| LSI_UNED_2_TASK1_RUN_07 | 0.476 |
| LSI_UNED_2_TASK1_RUN_02 | 0.474 |
| LSI_UNED_2_TASK1_RUN_01 | 0.471 |
| LSI_UNED_2_TASK1_RUN_08 | 0.470 |
| LSI_UNED_2_TASK1_RUN_03 | 0.467 |
| ETH-task1-Warriner.txt | 0.466 |
| LSI_UNED_2_TASK1_RUN_09 | 0.464 |
| ETH-task1-OptT2.txt | 0.461 |
| LSI_UNED_2_TASK1_RUN_04 | 0.461 |
| LSI_UNED_2_TASK1_RUN_11 | 0.459 |
| LSI_UNED_2_TASK1_RUN_10 | 0.457 |
| LSI_UNED_2_TASK1_RUN_05 | 0.454 |
| ETH-task1-OptT1.txt | 0.441 |
| sinai_emml_task1_3classes.txt | 0.409 |
| LSI_UNED_2_TASK1_RUN_14 | 0.408 |
| LSI_UNED_2_TASK1_RUN_15 | 0.408 |
| LSI_UNED_2_TASK1_RUN_13 | 0.407 |
| LSI_UNED_2_TASK1_RUN_12 | 0.405 |
| sinai_cesa-task1_raw.tsv | 0.389 |
| sinai_cesa-task1_normalized.tsv | 0.388 |
| UNED-JRM-task1.txt | 0.230 |

Ilustración 25: Resultados del TASS 2013

Por último, representamos la distribución de los resultados mediante la matriz de confusión.

| | | Etiquetado manual | | | | | |
|------------------------|------|-------------------|-----|-----|-----|-----|------|
| | | P+ | P | NEU | N | N+ | NONE |
| Etiquetado del sistema | P+ | 493 | 134 | 31 | 30 | 28 | 16 |
| | P | 832 | 604 | 188 | 210 | 108 | 246 |
| | NEU | 100 | 109 | 154 | 177 | 108 | 14 |
| | N | 36 | 82 | 148 | 566 | 362 | 77 |
| | N+ | 12 | 21 | 27 | 84 | 144 | 14 |
| | NONE | 178 | 282 | 121 | 268 | 98 | 1124 |

Ilustración 26: Matriz de confusión

En ella, como ayuda visual, se ha representado en cada celda una barra de tamaño proporcional al número contenido en ella respecto al máximo de la columna. De esta forma, leyendo la matriz por columnas, podemos ver comparativamente para un mismo tipo de etiquetado humano (por ejemplo en la primera columna, los textos etiquetados como P+), cómo se reparten las respuestas dadas por el sistema, identificando así si existe una concentración importante sobre la respuesta correcta o si existen errores cometidos sistemáticamente que la desvían del resultado deseado.

Para nuestro caso concreto, se observa como en su mayoría el etiquetado se produce sobre la respuesta correcta en cada caso (diagonal de la matriz). Sin embargo, la matriz revela algunos errores sobre los que se podría trabajar para mejorar el sistema.

Por un lado vemos en la primera columna como los textos muy positivos (P+) se etiquetan como positivos (P) en su mayoría. Para mejorar el sistema se podrían revisar estos casos en busca de reglas de diccionario que deberían etiquetar P+ en lugar de P. Algo parecido ocurre con los textos muy negativos (N+). Aun así estos errores forman parte de los llamados “errores leves”, por lo que su corrección no es tan importante.

Más atención merecen los textos etiquetados como neutrales, pues en ellos (tercera columna) las respuestas del sistema se están distribuyendo en gran medida sobre las etiquetas P y N, lo cual indica que se están cometiendo algunos errores a la hora de agregar los sentimientos encontrados o que no se están detectando algunos de ellos, de forma que la polaridad final tienda a ser positiva o negativa en lugar de neutra. Este es el mayor fallo que nos revela la matriz sobre nuestro sistema y que deberá ser revisado en la siguiente versión del mismo.

5.3. Evaluación del sentimiento a nivel de entidad

Con evaluación del sentimiento a nivel de entidad nos referimos a una medida de la capacidad de nuestro sistema para realizar un análisis orientado a aspectos.

Por exigencias de tiempo, no hemos podido realizar una evaluación totalmente formal sobre esta característica del sistema, aun así, con el fin de probar de alguna forma esta característica del sistema, hemos realizado una evaluación más informal en la que hemos sometido el sistema a una serie de pruebas propuestas y etiquetadas por nosotros mismos, en las que el sistema se ha ejecutado usando el mismo diccionario de carácter general desarrollado para la prueba de análisis de sentimiento a nivel global.

Durante el análisis de las respuestas dadas por el sistema, hemos encontrado diferentes aspectos que dificultan el análisis de sentimientos orientado a aspectos.

Para una gran parte de los ejemplos analizados, hemos encontrado que la expansión a través de las relaciones definida en el diseño del sistema (apartado 3.7.3.5) aporta un análisis muy fino y efectivo de la polaridad. En verbos del tipo “gustar”, “ser”, o “querer” el sistema es capaz de detectar adecuadamente las relaciones del verbo, asignando correctamente las polaridades adecuadas a cada entidad/concepto (“Me **encanta** la película”, “Ese chico **es** detestable”, “**Quiero** ese móvil”).

Sin embargo, existen otros verbos en la que el funcionamiento no está tan claro. El principal problema detectado en nuestro algoritmo son los falsos positivos que provoca.

Por una parte, la detección de relaciones realizada por el motor de PLN no es perfecta, provocando en ocasiones una propagación excesiva de las polaridades hasta elementos que no deberían propagarse.

Por otra parte, algunas de las relaciones detectadas, aunque correctamente detectadas por el motor, no deberían usarse para asignar polaridad. Por ejemplo, ante una oración con complementos de lugar o tiempo, generalmente no se deberían expandir las etiquetas encontradas. Por ejemplo, “Me **gusta** el gato de la tienda” o “Me **encantó** el partido de ayer”. En estos dos casos a los conceptos “tienda” o “ayer” no debería de llegar las polaridades detectadas por los verbos “gustar” y “encantar”.

Algo parecido ocurre con las relaciones de sujeto, pero aún más complejo, ya que el sujeto en ocasiones se ve afectado por la polaridad del verbo y en otras no (“El policía **arrestó** al preso” frente a “Telefónica **baja** en bolsa”).

En principio un algoritmo de filtrado que actuara sobre las relaciones parece que podría solucionar en gran medida el problema, sin embargo, es una mejora que está por estudiar en busca de definir totalmente su funcionamiento.

Además, existen ejemplos más complejos, en los que los verbos que actúan de distinta manera para la polaridad a nivel global y a nivel de entidad. El verbo “extrañar” es un buen ejemplo de ello. Poniendo como ejemplo la oración “**Extraño** mucho mi antiguo

teléfono”, podemos comprobar cómo la polaridad a nivel global debería ser negativa (ya que el autor expresa tristeza) y, sin embargo, la entidad “teléfono” debería ser identificada idealmente con una polaridad positiva, pues es autor expresa indirectamente su sentimientos de apego hacia el mismo.

Finalizada la valoración de las respuestas dadas por el sistema, hemos estimado que la precisión del mismo para la evaluación del sentimiento a nivel de entidad es de un 30% aproximadamente, en el caso del sistema con 5 niveles de polaridad y de aproximadamente un 45% en el sistema de 3 niveles que se describió en el apartado anterior (3.2.5 Resultados de la prueba).

El resultado es más pobre que en el caso del sentimiento global. Ya que, como hemos comprobado con algunos ejemplos, el sentimiento a nivel de entidad requiere de técnicas algo más avanzadas que aún están por diseñar.

5.4. Análisis de respuestas y errores del sistema.

En lo siguiente se presentan de forma explicativa las respuestas que dio el sistema para algunos de los textos contenidos en el corpus usado para la evaluación del sentimiento a nivel global. Con el fin de mostrar su funcionamiento de la forma más clara posible, así como ejemplificar algunos errores cometidos con su posible corrección, se han agrupado las respuestas en cuatro bloques: correctas, erróneas leves, falsos positivos/negativos y cambios de polaridad. Además, sobre cada texto se han subrayado⁴ las palabras que el sistema ha detectado y seleccionado para calcular la polaridad global del mismo y se han marcado con cursiva otras expresiones con polaridad que han afectado a la polaridad como operadores.

Primero, mostramos algunos ejemplos en los que el sistema ha etiquetado de forma correcta la salida.

| Texto | Etiquetado | Salida |
|--|------------|--------|
| Hoy asisitiré en Madrid a un seminario sobre la Estrategia Española de Seguridad organizado por FAES. | NONE | NONE |
| Buen día todos! Lo primero mandar un abrazo grande a Miguel y a su familia @libertadmontes Hoy podría ser un día para la grandeza humana. | P+ | P+ |
| El juez imputa a dos directivos de la Ciudad de las Artes y Ciencias de Valencia por el caso Urdangarín. Parece que todos lo están menos él | N | N |

⁴ Para representar la polaridad de cada palabra se ha coloreado el subrayado, utilizando el color verde para las polaridades positivas y el color rojo negativas.

| | | |
|---|-----|-----|
| Bueno, pausa para desayunar y cosas que hacer. Poco tiempo pero siempre agradable compartirlo con vosotros. | P | P |
| Iberia cancelará 107 vuelos este viernes por la huelga de pilotos - http://t.co/GGPDnBZ1 | N | N |
| La Universidad confía en De la Calle para encarar sus retos más ambiciosos http://t.co/7dzPPija | P+ | P+ |
| RT: La mayoría de los españoles apoya los recortes pero rechaza la subida de impuestos http://t.co/21zx9EA3 #palabrasqueduelen | NEU | NEU |
| Andalucía está comprometida con la estabilidad, pero exigimos lealtad institucional. No discriminar a CCAA por color político #debateAND | P+ | P+ |
| Ha muerto Antonio Mingote, gran dibujante gran humoristaY mejor persona. | N+ | N+ |
| Solucionaremos las necesidades que hay en Las Golondrinas y el colegio Pedro Garfia. Será un compromiso electoral cumplido en #sevillahoy | P | P |

Tabla 16: Respuestas correctas

En los primeros 5 ejemplos de la Tabla 16 se muestra un etiquetado básico, en el que la detección de palabras con polaridad ha permitido llegar hasta el resultado deseado. Posteriormente, mostramos ejemplos algo más complejos, como por ejemplo en “La Universidad confía en De la Calle para encarar sus retos más ambiciosos” en el que la palabra “reto” se ha desambiguado como positiva gracias a la aparición siguiente de “ambiciosos”.

En el siguiente ejemplo, “La mayoría de los españoles apoya los recortes pero rechaza la subida de impuestos”, los verbos “apoyar” “=P” y “rechazar” “!N” modifican sus polaridades según sus complementos, “recortes” “N” y “impuestos” “N” respectivamente. Igualmente ocurre en el siguiente, en el que el verbo “exigir” que por defecto posee una polaridad negativa en nuestro modelo “=N”, ha cambiado su polaridad por positiva gracias al complemento directo “lealtad” “P”.

Finalmente, es destacable en los dos últimos textos, como el hecho de priorizar el verbo sobre otras posibles palabras con polaridad ha contribuido a que éstos obtengan un etiquetado correcto en la salida.

Para seguir mostramos algunos ejemplos en los que se ha cometido errores en el cálculo de la intensidad, los que hemos llamado durante la evaluación “Errores leves”.

| Texto | Etiquetado | Salida | Error |
|---|------------|--------|---------------------|
| Agradezco a trabajadores y sindicatos la desconvocatoria de la huelga en el aeropuerto. Me he comprometido a mediar #sevillahoy | P | P+ | Etiquetado |
| Odio la presunta Justicia politizada. Profesionales, ya! | N | N+ | Etiquetado |
| Je,je!!! Eso es bueno @mkreyes80: @Buenafuente me has hecho llorar...De risa!" | P+ | P | Análisis sintáctico |

Tabla 17: Errores leves

Tanto en el primer ejemplo como en el último, el error se ha cometido por una discrepancia a la hora etiquetar las palabras clave. En el primero, el verbo “agradecer” ha sido etiquetado en el diccionario como “P+”, sin embargo, el texto ha sido considerado como “P” en el etiquetado humano, por lo que difícilmente vamos a llegar a ese resultado con una etiqueta definida como “P+”. Esto es un buen ejemplo de aquello que ya comentamos sobre los errores leves en el apartado de evaluación, la decisión de etiquetar un texto como “P” o como “P+” es un tanto subjetiva. Lo mismo ocurre en el segundo texto, en el que nuestras reglas consideran “Odiar” como “N+” y sin embargo el texto está etiquetado como “N”.

En el tercer caso el error es un poco diferente. Aunque nuestro etiquetado coincida con el humano, es un “error” en la etapa previa de análisis sintáctico el que produce la discrepancia en la salida. El motor de PLN separa el texto “De risa” como una nueva frase, por el uso previo de los puntos suspensivos y la mayúscula, lo que hace que quede una oración como “@Buenafuente me has hecho llorar...”, la cual se etiqueta como con una “N” al no detectarse el resto del contexto.

Siguiendo con los errores que comete el sistema, damos en la siguiente tabla algunos errores que han dado lugar tanto a falsos positivos como a falsos negativos.

| Texto | Etiquetado | Salida | Error |
|---|------------|--------|---------------------|
| Toca @crackoviadeTV3 . Grabación dl especial Navideño...Mari crismas! | P+ | NONE | Falta regla |
| Otro passing escalofriante sobre la linea pone a Nadal 5 a 1 en el tercero.La ensaladera, mas cerca. #VamosRafa #CopaDavis ole @RafaelNadal!! | P+ | NEU | Falta regla |
| Eso no es cierto, nos llevamos de maravilla. RT @Deyra_love_BTR: | P | NONE | Análisis sintáctico |

| | | | |
|---|------|------|---------------------|
| @AlejandroSanz -- POR QE!!! (cont) http://t.co/7DyF7Sus | | | |
| "@marinitaMF: @Edurnity Ojo, que yo me apellido "Molas", tal cual! y no es broma."en serio?? Como Mola!! Jajaja | P+ | NONE | Análisis sintáctico |
| Les comparto ,http://t.co/FFGlrD4 | NONE | P | Falso positivo |
| En el blog: Defender la <u>solidaridad</u> http://t.co/8sDDYC8j | NONE | P | Falso positivo |
| @MarianoRajoy : "Salir <u>adelante</u> no es tarea de un Gobierno solo" http://t.co/mJSq4fz1 | NONE | P | Falso positivo |

Tabla 18: Falsos positivos y falsos negativos

Por un lado algunos de estos errores se están produciendo por falta de reglas de detección en el modelo. En el primer texto, sería necesaria una regla que identificará las diferentes formas coloquiales con las que un usuario puede felicitar la navidad y etiquetarlas como positivas. Por ejemplo definiendo una regla como la siguiente "mari_crismas|merry_christmas|mery_christmas|merry_xmas P+" ya estaríamos contemplando el ejemplo del primer texto de la Tabla 18 y algunos más.

En el segundo caso, es más bien el dominio general del modelo el que produce el error. En nuestro modelo, el adjetivo "escalofriante" se etiqueta como negativo, lo que en principio parece ser correcto, sin embargo, su uso en este texto concreto es con carácter positivo. Este problema se solucionaría si hubiéramos desarrollado un modelo de análisis de sentimiento para el dominio específico del deporte, en el que en principio, no se le asignaría polaridad a dicho adjetivo.

En los dos siguientes ejemplos, nos encontramos con errores en la fase de análisis sintáctico, los cuales terminan produciendo una salida incorrecta para el sistema. El primer caso es sencillo de solucionar, el error se debe a que no se ha tenido en cuenta a la hora de definir una regla para el adjetivo "maravilla" que el de motor de PLN segmenta la expresión "de maravilla" como un único token multiword ya que posee un significado único. Bastaría con añadir la regla correspondiente al diccionario para solucionar este error: "maravilla|de_maravilla P". El segundo ejemplo de error por análisis sintáctico es algo más complejo. El motor de PLN es un software muy complejo que aplica una gran variedad de técnicas al texto de entrada con el fin de extraer toda la información posible de él. Entre ellas, algunas contemplan el uso de mayúsculas con el fin de extraer del mismo entidades que no sean detectadas por los recursos lingüísticos base del sistema, lo que en este caso provoca la unión de la

expresión “Como Mola” como un único token de tipo entidad, que pasa a ser indetectable para el modelo de análisis de sentimiento.

Por último, queríamos ilustrar con los tres últimos textos un error común y difícil de superar para un modelo de análisis de sentimientos general. En el texto “Les comparto, <http://t.co/FFGIreD4>” es inevitable que el sistema obtenga como salida un caso de falso positivo, al detectar el verbo “compartir” que posee una polaridad positiva de forma general. El texto realmente es un simple enlace que un usuario comparte con los demás, lo cual en sí mismo está ausente de subjetividad u opinión. Igualmente pasa con otros verbos usados en tweets a modo de publicidad, del tipo “*Comparte* este enlace”, “*Gana* está estupenda guitarra” o “Por favor *dona* para colaborar” los cuales pueden dar lugar a muchos falsos positivos durante el análisis. Siempre que el texto realmente sea una referencia en forma de citas o enlaces web, no existe información subjetiva que deseemos detectar, pero existe una gran dificultad para desambiguar el uso de los verbos en referencias, frente a su uso general.

Finalmente, agrupamos en la siguiente tabla (Tabla 19) algunos ejemplos de los errores más dañinos para la calidad del sistema diseñado, los errores de cambio de polaridad.

| Texto | Etiquetado | Salida | Error |
|--|------------|--------|-------------|
| Soy consciente de que estamos pidiendo un gran esfuerzo a las familias castellanomanchegas. | N | P+ | Falta regla |
| Me reservo dónde y cómo deposito mis ahorros con mi familia. Sólo estoy obligado a la transparencia pública aunque veo que hay mucho experto. | N | P | Falta regla |
| Bien que lo siento si es así.La UEDila la bienvenida ayer.Esperemos que aun salga adelante . | N | P | Falta regla |
| Desgraciadamente el paro en España sigue creciendo . Madrid va mejorando con 8.000 cotizantes más y 4.000 desempleados menos. | N | P+ | Falta regla |

Tabla 19: Cambios de polaridad

Estos errores básicamente se producen por falta de reglas de desambiguación, que finalmente acaban llevando al sistema a dar como salida la polaridad opuesta a la etiquetada. Algunos son relativamente sencillos de corregir mediante los mecanismos de desambiguación por contexto o por morfosintaxis que nos proporciona el

diccionario. Otros sin embargo, contienen la información de opinión a extraer de una forma muy sutil y compleja, haciendo que en muchos casos sea imposible crear una regla de diccionario que los detecte.

En el primer caso por ejemplo, faltaría añadir una regla a la palabra “esfuerzo” para que fuese detectada, por ejemplo así: “`pedir>esfuerzo` N”.

En los dos siguientes ejemplos no es sencillo dar con una regla que nos permita tratar de forma general ese tipo de casos, por llevar la información subjetiva de una forma muy sutil, aunque se podría introducir reglas como “`haber_mucho_experto` N” y “`debería_de` NEG”, es probable que estas condujeran a otros cuantos errores que necesitaran ser desambiguados de una forma más exhaustiva.

En el último ejemplo de todos, el error se debe a la ausencia de una regla del tipo “`subir|crecer|aumentar>paro` N+” y “`bajar|decrecer|disminuir>paro` P+” para detectar bien la polaridad del texto, la cual además, en este caso parece mal etiquetada por los observadores humanos, ya que la regla parece poseer una polaridad neutral, al expresar una idea positiva y otra negativa, en lugar de negativa, como indica su etiquetado.

A modo de **resumen** final, tras todos los ejemplos vistos en este apartado, se puede decir que a través de ellos hemos podido comprobar el funcionamiento de las herramientas diseñadas para el sistema, como los operadores modificadores, o la priorización del verbo y que, de la misma forma, hemos detectado las causas que habitualmente hacen al sistema fallar, las cuales son principalmente son: la falta de reglas en el diccionario, la falta de desambiguación o los errores cometidos en la fase de análisis morfosintáctico.

6. Conclusiones y Trabajos futuros

6.1. Conclusiones

Después del diseño del sistema, la implementación del mismo y toda la fase de pruebas y evaluación, podemos decir que éste cumple con las expectativas dictadas en los objetivos del proyecto.

Se ha implementado un sistema de análisis de sentimientos basado en reglas con el que se pueden construir potentes modelos de análisis de sentimiento de la forma más rápida y efectiva posible, gracias a que se apoya en las reglas del motor de análisis del lenguaje natural.

Con respecto a los resultados obtenidos en las evaluaciones, aunque la precisión obtenida para el sistema puede parecer un poco pobre (un 42% para el sistema de 5 niveles de polaridad y un 64% para el sistema de 3 aproximadamente), en este caso no hay que juzgar al sistema estrictamente por este resultado numérico. Es necesario tener en cuenta el contexto de la evaluación realizada.

La evaluación a la que se ha sometido el sistema ha sido realizada sobre todo por comprobar el correcto funcionamiento del mismo, así como obtener una primera estimación de su calidad y analizar su comportamiento en busca de posibles mejoras, sin embargo, en la prueba se analizaron textos de Twitter muy variados, que no tenían ninguna temática fija, y en los que sus autores no pertenecían a ningún grupo social limitado, haciendo que la forma de expresión variara mucho en cada uno. Bajo este caso de uso, el desarrollo de un sistema mediante reglas que intente abarcar todos los casos posibles es muy complejo, y por ello hemos obtenido una precisión no tan alta.

En este tipo de escenarios es normal que el uso de técnicas de aprendizaje automático supere a sistemas basados en reglas, como el nuestro, pues desarrollar mediante reglas un sistema para un escenario tan amplio requiere de un gran esfuerzo y tiempo, aun así, en el caso de la evaluación para la evaluación de la polaridad en tres niveles se alcanzó una precisión muy cercana a las mejores obtenidas por los participantes del TASS 2013, los cuales usaban técnicas de aprendizaje automático [36].

En lo que realmente nuestro sistema va a destacar sobre los sistemas de aprendizaje automático es en dominios más restringidos, donde la temática de los textos sea más reducida y la forma de expresión en ellos esté más limitada, pues en estos escenarios, nuestro sistema nos aporta un análisis muy fino y completamente configurado por el usuario del mismo. Al contrario que en los sistemas de aprendizaje automático, somos nosotros los que debemos aportar la “inteligencia” necesaria para realizar el análisis, definiendo manualmente las reglas que conforman el sistema, algo que no es tan

rápido de desarrollar, pero que nos da la oportunidad de tener un control total sobre el etiquetado del texto.

Además, su implementación bajo C++ ha logrado dar al mismo tiempo una velocidad de respuesta adecuada, con tiempos que ronda menos de 1 segundo por análisis (carga completa de diccionario incluida) para textos de un tamaño mediano (unos 500 caracteres), en un hardware de referencia con 8GB de RAM, Intel Xeon DualCore a 2 GHz.

6.2 Trabajo futuro

El análisis de sentimientos es una tarea difícil en la que nunca se puede llegar a una precisión del 100%, ya que hasta dos personas diferentes tendrían problemas en llegar a un acuerdo sobre ciertos textos, pero partiendo del sistema implementado, se puede trabajar en desarrollar distintas ampliaciones con el fin de conseguir poco a poco que éste tenga una precisión lo más alta posible.

Por un lado se pueden realizar mejoras **particularizando su diseño** para un escenario concreto, y por otra parte se puede **mejorar el diseño general** del mismo, ampliando sus capacidades o analizando y mejorando los algoritmos actualmente propuestos.

De la misma forma, otra línea de trabajo futuro sería llevar el sistema a diferentes idiomas, para lo que se debe tener en cuenta las particularidades del idioma al que se desee llevar la implementación, a parte del desarrollo del diccionario de reglas.

A continuación se lista una serie de posibles mejoras en las que se podría invertir esfuerzo para mejorar el sistema.

- Adaptación a dominios concretos
Como ya hemos comentado en algunas ocasiones a lo largo de este documento, según el escenario de aplicación se puede afinar el diseño del sistema, principalmente **modificando el diccionario** de reglas general y ampliando las reglas existentes en el diccionario de forma que éstas sean coherentes con la temática del escenario de uso, así como realizando desambiguaciones basadas en el conocimiento del dominio más efectivas.

También se puede **perfilar el diseño interno** del sistema en función del dominio de la aplicación, por ejemplo, el preprocesado del texto será muy diferente para un texto coloquial, como puede ser un tweet, frente al preprocesado de un texto formal, que sigue unas reglas de escritura mucho más rígidas.

Por otra parte, el diseño interno del sistema también puede sufrir diferentes modificaciones en función del idioma que se esté analizando, ya que no es el

vocabulario lo único que cambiaría de un idioma a otro, existen otros aspectos a tratar que pueden sufrir cambios de enfoque más complejos, como puede ser la gestión del uso de negadores en otro idioma particular que no sea el español.

- Mejoras de diseño

Durante la implementación y evaluación del sistema, se han conocido mejor las necesidades del mismo, lo que ha definido de una serie de mejoras con vista a siguientes versiones, las cuales comentamos brevemente en la siguiente lista:

- Tiempos verbales y tipo de oración

El tiempo verbal o el tipo de oración (imperativa, interrogativa, afirmativa, etc.) afecta en gran medida al significado e interpretación de ésta, afectando con ello al sentimiento. Por ejemplo si analizamos la oración “El servicio es bueno” frente a “El servicio era bueno”, vemos que ambas frases no poseen la polaridad de sentimiento, a pesar de que a nivel de lema sean la misma, de hecho, se podría decir que tienen una polaridad inversa.

En el sistema actual, el análisis de sentimientos se realiza de manera insensible a tiempos verbales o tipos de oraciones, lo cual se podría mejorar aprovechando de nuevo las capacidades del motor de PLN.

- Análisis extendido de opiniones

En nuestro diseño, con el fin de obtener un análisis de polaridad a nivel global más preciso, hemos decidido priorizar el sentimiento dado por los verbos de las oraciones, sin embargo, pueden existir otras opiniones relevantes, como adjetivos calificativos acompañando al nombre, que actualmente no se analizarían. Por ejemplo, en la oración “Han cancelado el magnífico concierto”, aparte del sentimiento negativo asociado al verbo “cancelar”, habría que detectar de alguna forma el adjetivo “magnífico” que afecta a la entidad “concierto” positivamente. Para mejorar el análisis a nivel de entidad éste es uno de los primeros aspectos que tendríamos que implementar.

- Análisis de comparaciones

Existe toda una serie de formaciones habitualmente usadas para expresar opiniones de forma comparativa que no son tratadas por nuestro sistema de una forma específica. Para tratar con oraciones como “La pantalla **es mejor** en el modelo nuevo **que** en el viejo” o “La comida es buena **pero** es demasiado cara” sería conveniente realizar un análisis específicamente diseñado para evaluar la estructura

comparativa de la frase. La implementación de esta mejora sería especialmente útil para el análisis de *reviews*, en las que se suelen usar mucho este tipo de comparaciones.

- Mejoras en la potencia de las reglas de diccionario
Por último, siempre se pueden mejorar las técnicas de desarrollo de modelos introduciendo nuevos mecanismos útiles para la definición de reglas, por ejemplo, permitiendo reglas con contextos limitados en un cierto número de palabras o permitiendo configurar la distancia máxima de un negador (apartado 3.7.2) de forma explícita en la regla.

Además, por falta de tiempo, queda como trabajo futuro realizar una **evaluación** más formal y detallada del **sentimiento a nivel de entidad**. Con ella, además de obtener un resultado más riguroso de la precisión del sistema en dicho cometido, se pretende analizar en profundidad los textos evaluados por el sistema, detectando problemas en el actual algoritmo en busca de una mejora del mismo.

7. Bibliografía

Las referencias abajo incluidas aparecen por orden de citación en el texto.

- [1] Stream Social Q1 2013, Estudio de sobre el uso de redes sociales en Internet <http://www.rvillanuevarios.com/las-redes-sociales-de-mayor-crecimiento/>
[Visitado en Julio 2014]
- [2] Daedalus <http://www.daedalus.es/> [Visitado en Julio 2014]
- [3] Bo Pang, Lillian Lee, Shivakumar Vaithyanathan. Proceedings of EMNLP - Thumbs up? Sentiment classification using machine learning techniques, pag. 79-86. Año 2002 <http://www.cs.cornell.edu/home/llee/papers/sentiment.home.html> [Visitado en Julio 2014]
- [4] Julio Villena Román, Janine García Morera. TASS 2013 - Workshop on Sentiment at SEPLN 2013. <http://www.daedalus.es/TASS2013/papers/tass2013-overview.pdf>
[Visitado en Julio 2014]
- [5] Turney, P.D. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, 2002. <http://arxiv.org/abs/cs.LG/0212032>
[Visitado en Julio 2014]
- [6] Aprendizaje supervisado, Artículo en Wikipedia http://es.wikipedia.org/wiki/Aprendizaje_supervisado [Visitado en Julio 2014]
- [7] Johan Bollen, Huina Mao, Xiao-Jun Zeng. Twitter mood predicts the stock market <http://arxiv.org/abs/1010.3003> [Visitado en Julio 2014]
- [8] Xabier Saralegi Urizar, Iñaki San Vicente Roncal. Elhuyar at TASS 2013. http://www.elhuyar.org/hizkuntza-zerbitzuak/informazioa/lritzien-Erauzketa/Elhuyar_TASS2013report_v2.pdf [Visitado en Julio 2014]
- [9] Text REtrieval Conference (TREC) <http://trec.nist.gov/> [Visitado en Julio 2014]
- [10] NII Testbeds and Community for Information access Research (NTCIR) <http://research.nii.ac.jp/ntcir/index-en.html> [Visitado en Julio 2014]
- [11] The CLEF Initiative <http://www.clef-initiative.eu/> [Visitado en Julio 2014]

- [12] SEPLN (Sociedad Española para el Procesamiento del Lenguaje Natural) <http://www.sepln.org/> [Visitado en Julio 2014]
- [13] Textalytics <https://textalytics.com> [Visitado en Julio 2014]
- [14] AlchemyAPI <http://www.alchemyapi.com/> [Visitado en Julio 2014]
- [15] Semantria <https://semantria.com/> [Visitado en Julio 2014]
- [16] Repustate <https://www.repustate.com/> [Visitado en Julio 2014]
- [17] NLTK (Natural Language Toolkit) <http://www.nltk.org/> [Visitado en Julio 2014]
- [18] Text-processing, sentiment analysis demo <http://text-processing.com/demo/sentiment/> [Visitado en Julio 2014]
- [19] Google Prediction API, Creating a Sentiment Analysis Model. https://developers.google.com/prediction/docs/sentiment_analysis [Visitado en Julio 2014]
- [20] Jaime Carbonell. Carnegie Mellon University. Congreso de Sevilla, 1992. Centro Virtual Cervantes http://cvc.cervantes.es/obref/congresos/sevilla/tecnologias/ponenc_carbonell.htm [Visitado en Julio 2014]
- [21] RAE, Diccionario de la Real Academia Española: <http://www.rae.es/> [Visitado en Julio 2014]
- [22] Eric Rochester. Mastering Clojure Data Analysis, Understanding sentiment analysis. Año 2014.
- [23] Precision and recall, Artículo en Wikipedia http://en.wikipedia.org/wiki/Precision_and_recall [Visitado en Julio 2014]
- [24] Tokenization, Artículo en Wikipedia <http://en.wikipedia.org/wiki/Tokenization> [Visitado en Julio 2014]
- [25] JSON, Artículo en Wikipedia <http://en.wikipedia.org/wiki/JSON> [Visitado en Agosto 2014]

- [26] API de POS y Parsing, Textalytics
<https://textalytics.com/core/parser-info> [Visitado en Agosto 2014]
- [27] Documentación online, Textalytics
<https://textalytics.com/core/tagset?lang=es> [Visitado en Agosto 2014]
- [28] Trie, Artículo en Wikipedia
<http://en.wikipedia.org/wiki/Trie> [Visitado en Agosto 2014]
- [29] Rae, La doble negación en español:
<http://www.rae.es/consultas/doble-negacion-no-vino-nadie-no-hice-nada-no-tengo-ninguna> [Visitado en Agosto 2014]
- [30] Web sobre Gramática. Ejemplos de oraciones nominales.
<http://www.gramaticas.net/2012/05/ejemplos-de-oraciones-nominales.html>
[Visitado en Agosto 2014]
- [31] Steve Oualline . O'Reilly Media, Inc. Practical C++ Programming, What Is C++?. Año 2002.
- [32] Documentación sobre C++, Map
<http://www.cplusplus.com/reference/map/map/> [Visitado en Agosto 2014]
- [33] Congresos del SEPLN <http://www.sepln.org/category/congresos/>
[Visitado en Septiembre 2014]
- [34] Descripción del corpus, TASS 2013
<http://www.daedalus.es/TASS2013/corpus.php> [Visitado en Septiembre 2014]
- [35] Matriz de Confusión, Artículo en Wikipedia
http://es.wikipedia.org/wiki/Matriz_de_confusi%C3%B3n [Visitado en Septiembre 2014]
- [36] TASS 2013 - Workshop on Sentiment Analysis at SEPLN 2013
<http://www.daedalus.es/TASS2013/papers/tass2013-overview.pdf> [Visitado en Septiembre 2014]