PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Implementación de un lematizador para una lengua de escasos recursos: caso shipibo-konibo

TESIS PARA OPTAR EL TÍTULO PROFESIONAL EN INGENIERÍA INFORMÁTICA

AUTOR
José Humberto Pereira Noriega

ASESOR:
Mag. Felix Arturo Oncevay Marcos

Lima, Diciembre, 2018

AGRADECIMIENTOS

En primer lugar, deseo agradecer a mis padres por todo su apoyo incondicional durante la elaboración de este proyecto de fin de carrera, así como durante toda la mi carrera universitaria.

En segundo lugar, agradezco a mi asesor, el profesor Arturo Oncevay por todo el apoyo brindado y por haberme guiado en esta tesis, y al profesor Andres Melgar, el que me presentó la oportunidad de participar en este proyecto.

En tercer lugar, agradezco a todos los integrantes del proyecto Chana, gracias a los cuales pude obtener diversos conocimientos y materiales necesarios para realizar este proyecto.

Finalmente, agradezco a mi universidad y al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC) que a través del proyecto "Una plataforma de software para la traducción automática de textos entre lenguas originarias de la Amazonía peruana y español" (225-2015 FONDECYT), dieron los recursos necesarios para que se logré este proyecto de fin de carrera.

PUBLICACIONES

El presente trabajo se realiza para optar por el título de Ingeniero Informático de la Pontificia Universidad Católica del Perú y como parte de este se han realizado las siguientes publicaciones relacionadas durante su elaboración, las cuales se incluyen como anexos:

- ***Ship-LemmaTagger: Building a NLP Toolkit for a Peruvian Native Language***, **José Pereira-Noriega**, Rodolfo Mercado-Gonzales, Andrés Melgar, Marco Sobrevilla-Cabezudo y Arturo Oncevay-Marcos; Text, Speech, and Dialogue: 20th International Conference, TSD 2017. Donde se presentó los resultados derivados de la elaboración de este proyecto. (Anexo 1)

- ***ChAnot: An Intelligent Annotation Tool for Indigenous and Highly Agglutinative Languages in Peru***, Rodolfo Mercado, **José Pereira**, Marco Antonio Sobrevilla Cabezudo and Arturo Oncevay; Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Donde se presentó la herramienta de anotación elaborada para la creación del corpus (ChAnot), así como su integración con el lematizador obtenido como resultado de uno de los objetivos de este proyecto. (Anexo 2)

RESUMEN

Desde que el Ministerio de Educación oficializó el alfabeto shipibo-konibo, existe la necesidad de generar una gran cantidad de documentos educativos y oficiales para los hablantes de esta lengua, los cuales solo se realizan actualmente mediante el apoyo de traductores o personas bilingües. Sin embargo, en el campo de la lingüística computacional existen herramientas que permiten facilitar estas labores, como es el caso de un lematizador, el cual se encarga de obtener el lema o forma base de una palabra a partir de su forma flexionada. Su realización se da comúnmente mediante dos métodos: el uso de reglas morfológicas y el uso de diccionarios. Debido a esto, este proyecto tiene como objetivo principal desarrollar una herramienta de lematización para el shipibo-konibo usando un corpus de palabras, la cual se base en los estándares de anotación utilizados en otras lenguas, y que sea fácil de utilizar mediante una librería de funciones y un servicio web. Esta herramienta final se realizó utilizando principalmente el método de clasificación de los k-vecinos más cercanos, el cual permite estimar la clase de un nuevo caso mediante la comparación de sus características con las de casos previamente clasificados y dando como resultado la clase más frecuente para valores similares. Finalmente, la herramienta de lematización desarrollada logró alcanzar una precisión de 0.736 y de esta manera superar a herramientas utilizadas en otros idiomas.

TABLA DE CONTENIDOS

Pág.

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

CAPÍTULO 1: Problemática

1.1 Problemática

El 13 de junio del 2015 el Ministerio de Educación oficializó 24 alfabetos de lenguas originarias a través de la Resolución Ministerial N° 303-2015-MINEDU, con el fin de uniformizar su escritura para su uso en los ámbitos de la educación intercultural bilingüe, la publicación de documentos oficiales y la transmisión de información escrita entre entidades públicas y los grupos étnicos que utilizan estas lenguas [MINEDU, 2015]. Debido a esto, ha surgido la necesidad de generar una gran cantidad de documentos para los hablantes de estas lenguas.

Además, gracias a la ley Nº 29735 que regula el uso, preservación, desarrollo, recuperación, fomento y difusión de las lenguas originarias del Perú, se reconoce que es necesario una regulación activa por parte del Estado para hacer cumplir dos derechos constitucionales de suma importancia: el de que los peruanos puedan comunicarse con el Estado y con otras personas en su lengua materna (artículo 2° inciso 19), y el de oficializar el uso de las lenguas originarias en las zonas en donde estas sean predominantes (artículo 48) [CONGRESO DE LA REPÚBLICA, 2011], por lo que es necesario fomentar el uso de estas lenguas no solo de forma oral sino también de forma escrita.

De esta forma, se presentan diversos problemas que se afrontan actualmente sólo en parte con el apoyo de traductores o personas bilingües: la preparación y desarrollo de gran cantidad de material educativo para el programa de Educación Intercultural Bilingüe [Ministerio de Educación, 2016], la generación de documentos oficiales de entidades estatales en cada una de las lenguas originarias para la comunicación con comunidades nativas y la verificación ortográfica de estos documentos.

Sin embargo, en el campo de la lingüística computacional y el Procesamiento del Lenguaje Natural (PLN) existen ramas que se enfocan en la solución de estos problemas mediante el uso de diversas técnicas como la normalización, la tokenización, la lematización o el etiquetado gramatical [Tufiş; Ceauşu, 2009]. Estas permiten procesar las reglas morfológicas de las palabras de forma automática y son ya utilizadas con éxito en lenguas como el español o el inglés, pero que aún no son aplicadas en el caso las lenguas que poseen escasos recursos o que se encuentran en un ámbito menos privilegiado [Singh, 2008], como son las lenguas originarias mencionadas anteriormente.

Entre todas estas lenguas originarias surge el caso del shipibo-konibo, lengua representativa de la familia lingüística Pano. El shipibo-konibo es la sexta lengua con mayor cantidad de hablantes nativos en el Perú, con 22,517 hablantes según el censo realizado en el 2007 [Sullón et al., 2013], y que es hablada en diversas partes del país y es usada en 299 colegios públicos (Huánuco, Loreto, Madre de Dios, Ucayali y Lima) que se encuentran dentro del programa de Educación Intercultural Bilingüe (EIB). Lamentablemente, el shipibo-konibo no posee aún ningún recurso lingüístico computacional propio, como sí sucede en otras lenguas nativas peruanas, entre ellas el quechua (ANTIMORFO [Gasser, 2011]) o el aymara (Aymara LFG Grammar [Homola, 2011]), o ha sido adaptado a alguno existente.

Por este motivo, es necesario la creación de herramientas que permitan el acceso del lenguaje a un marco computacional de PLN y que estas se elaboren bajo el estándar actual (Universal Dependencies) que se vienen usando con otros lenguajes [Pyysalo et al., 2015], de modo que estas apoyen a la solución de los problemas actuales presentes en esta lengua y que puedan ser utilizadas junto a otras herramientas ya existentes de manera sencilla.

Una de estas herramientas es el lematizador, el cual se encarga de generar una palabra general a partir de un *token* de entrada mediante el proceso de lematización. Este proceso consiste en la obtención del lema o forma base de una palabra a partir de su forma flexionada

usando diccionarios y análisis morfológico [Ting et al., 2014], de modo que sea posible la agrupación de varias formas flexionadas con una sola para facilitar su posterior uso [Frakes; Fox, 2003]. Por ejemplo, para las palabras flexionadas “comen, comieron y comimos”, su respectivo lema es “comer”, que es la palabra que agrupa el significado base de todas ellas. De este modo, se puede dar apoyo a otras tareas como traductores automáticos [Goldwater; McClosky, 2005], correctores ortográficos o motores de búsqueda [Halácsy; Trón, 2006].

Su realización se da comúnmente mediante dos métodos: el uso de reglas morfológicas y diccionarios; ambos pueden ser generados mediante aprendizaje de máquina o de forma manual. La complejidad de esta labor varía de acuerdo al lenguaje que se utilice, siendo más dificultosa para lenguas aglutinantes como es el shipibo-konibo, los cuales se caracterizan por formar sus palabras mediante la unión de una raíz y un sinnúmero de afijos, y para casos en que se posea escasos recursos lingüísticos digitales [Juršic et al., 2007].

En conclusión, es necesario que se desarrolle software y técnicas capaces de ayudar al procesamiento de palabras en lenguas de bajos recursos del Perú, como es el caso de un lematizador. No obstante, esta es una tarea complicada debido a la diversidad que existe en cada lenguaje, en especial en los lenguajes predominantemente aglutinantes como es el shipibo-konibo. Entre las posibles soluciones existe el uso de diccionarios de reglas gramaticales, los cuales han probado ser muy eficientes para los casos de estas lenguas debido a la poca cantidad de datos existente. Para establecer estas reglas es necesario un análisis de la flexión de la lengua y de ver todos los posibles casos de derivaciones conjuntas que puedan aplicar a una palabra. Sin embargo, aún no se ha realizado un desarrollo concreto en lenguas amazónicas del Perú. Por estas razones, en este proyecto se buscará implementar un lematizador capaz de obtener los lemas correspondientes de las palabras flexionadas del shipibo-konibo, con un diseño capaz de ser reutilizado para otras lenguas de escasos recursos del Perú.

1.2 Objetivo General, Objetivos Específicos y Resultados Esperados

1.2.1 Objetivo general

Desarrollar una herramienta de lematización para el shipibo-konibo usando un corpus de palabras anotadas y reglas generadas por expertos en la lengua.

1.2.2 Objetivos específicos

OE1. Implementar una herramienta web para la anotación de palabras y definir los formatos de anotación de las categorías gramaticales usados por la herramienta para formar un corpus de palabras.

OE2. Entrenar modelos algorítmicos que predigan el patrón de comportamiento que permita formar reglas morfológicas de manera automática mediante el uso de los recursos anotados para la lengua.

OE3. Implementar un servicio web y una librería de funciones que permitan utilizar las herramientas generadas en el proyecto para otras aplicaciones en shipibo-konibo.

1.2.3 Resultados esperados

Para OE1:

R1.1 Alineación de las categorías y accidentes gramaticales a los definidos en el estándar propuesto por Universal Dependencies (UD).

R1.2 Herramienta de anotación web que permita generar un corpus de oraciones anotadas con la información morfológica y gramatical de las palabras.

R1.3 Corpus de palabras anotado con las categorías definidas por las UD y con los respectivos lemas de las palabras.

Para OE2:

R2.1 Modelo de clasificación entrenado que genere reglas de derivación morfológica mediante el uso un corpus de palabras anotadas.

R2.2 Componente de software que use el modelo entrenado y generado a partir del corpus.

R2.3 Validación de resultados de la herramienta final empleando métricas de precisión y de distancia entre palabras.

Para OE3:

R3.1 Servicio web que permita utilizar la herramienta final para obtener el lema de palabras en shipibo-konibo

R3.2 Librería de software que contenga las herramientas desarrolladas y el modelo entrenado anteriormente, así como las funciones necesarias para la generación de nuevos modelos para otros idiomas similares.

1.3 Herramientas, métodos y procedimientos

En esta sección se presentarán las herramientas, métodos y procedimientos necesarios para lograr los resultados esperados del presente proyecto de fin de carrera. Además, en la Tabla 1 se muestra la relación entre los resultados esperados y las herramientas a utilizarse:

Tabla 1: Relación entre Resultados Esperados y las Herramientas

Resultados Esperados	**Herramientas a usar**
R1.1 Alineación de las categorías y accidentes gramaticales a los definidos en el estándar propuesto por Universal Dependencies (UD).	● Estándar establecido por Universal Dependencies (UD)
R1.2 Herramienta de anotación web que permita generar un corpus de oraciones anotadas con la información morfológica y gramatical de las palabras.	● Etiquetador gramatical y morfológico (ChAnot) ● Python

Resultados Esperados	Herramientas a usar
R1.3 Corpus de palabras anotado con las categorías definidas por las UD y con los respectivos lemas de las palabras.	● Etiquetador gramatical y morfológico (ChAnot)
R2.1 Modelo de clasificación entrenado que genere reglas de derivación morfológica mediante el uso un corpus de palabras anotadas.	● Scikit-learn ● TiMBL ● Python
R2.2 Componente de software que use el modelo entrenado y generado a partir del corpus	● Python ● Lemmagen
R2.3 Validación de resultados de la herramienta final empleando métricas de precisión y de distancia entre palabras.	● Scikit-learn ● NLTK
R3.1 Servicio web que permita utilizar la herramienta final para obtener el lema de palabras en shipibo-konibo	● Python ● Flask
R3.2 Librería de software que contenga las herramientas desarrolladas y el modelo entrenado anteriormente, así como las funciones necesarias para la generación de nuevos modelos para otros idiomas similares.	● Python ● Scikit-learn

1.3.1 Herramientas

Python

Es un lenguaje de programación interpretado, lo que significa que se ejecuta sobre un intérprete sin necesidad de ser previamente compilado, capaz de utilizar diversos paradigmas de programación como la orientada a objetos, la imperativa o la funcional. Entre sus principales características se encuentran las siguientes [Python.org, 2016][Docs.python.org, 2016]:

- La capacidad de ser portable a diversos sistemas operativos como Windows, Unix, o OS X.
- Una sintaxis simple con un alto poder.
- Posee licencia de código abierto.
- Posee diversas librerías diseñadas para ámbitos científicos.

Esta herramienta será utilizada para el desarrollo del proyecto como base de programación e integración del resto de herramientas y para realizar el procesamiento de los corpus necesarios para el trabajo.

Scikit-learn

Scikit-learn o Sklearn es una librería libre hecha para Python que integra múltiples algoritmos de aprendizaje de máquina, supervisados y no supervisados. Su principal enfoque es llevar el aprendizaje de máquina a personas no expertas mediante el uso de un lenguaje de alto nivel, simple de usar y con una amplia documentación [Pedregosa et al., 2011].

Asimismo, esta librería posee un módulo especializado, sklearn.metrics, que cuenta con funciones que realizan el cálculo de las métricas validación de los resultados. Entre ellas se encuentran algunas como precisión, exhaustividad o valor-f. De este modo, esta herramienta, que cuenta con la implementación del algoritmo clasificatorio a utilizar, se podrá desarrollar un clasificador que utilice las reglas morfológicas y posteriormente la validación de los resultados.

Etiquetador gramatical y morfológico (ChAnot)

En el marco de este proyecto, se ha desarrollado una aplicación web en Java llamada ChAnot [Mercado et al; 2018] que permite la anotación manual de palabras en shipibo-konibo con sus respectivos lemas, afijos y categorías por expertos en la lengua, las cuales servirán como datos de entrenamiento para la realización de este proyecto, así como datos de prueba para la verificación de los resultados.

En esta herramienta, se ha realizado una alineación previa de las categorías gramaticales del shipibo-konibo a las presentes en las Universal Dependencies, de modo que exista un estándar adecuado de los datos anotados.

Gracias a su uso, se logró obtener el corpus de palabras necesarias para el desarrollo de este proyecto mediante el trabajo de anotación realizado por expertos en la lengua dentro de la herramienta.

NLTK: función edit-distance

NLTK, o "Natural Language Toolkit", es una librería libre hecha para Python, diseñada para el apoyo a la lingüística computacional. En ella se implementan diversos módulos con funciones frecuentemente utilizadas en este campo como: tokenizadores, stemmers, taggers, clasificadores [Loper; Bird, 2002], entre otras.

Asimismo, entre sus diversas funcionalidades, existe la posibilidad del cálculo de medidas de distancia entre palabras mediante el uso de la función edit-distance, la cual obtiene automáticamente la distancia entre dos palabras, las cuales, debido a su fácil uso, serán utilizadas para la validación de los resultados.

Lemmagen

Lemmagen [Jozef Stefan Institute, 2010] es una herramienta de lematización multilingüe de código abierto, la cual implementa un algoritmo de aprendizaje para la generación automática de reglas de derivación y que será utilizada como uno de los métodos a ser integrados en la herramienta final.

Estas reglas se establecen mediante el uso del formato Ripple down rules (RDR) y se generan mediante un léxico de palabras con sus respectivas categorías gramaticales y lemas. La idea principal de este enfoque es relacionar un nuevo caso al más similar que haya sido entrenado por el algoritmo, transformando así la tarea de lematización a una de clasificación [Juršic et al, 2010].

Debido a que la medida seleccionada para establecer la similitud de las palabras fue la longitud de los sufijos compartida entre dos palabras, esta herramienta funciona de mejor manera en lenguas en donde la inflexión se expresa comúnmente por sufijos, como es el caso de las lenguas indoeuropeas para las que se realizó este trabajo o como es en nuestro caso el shipibo-konibo.

TiMBL

TiMBL, o "Tilburg Memory-Base Learner", es un paquete de software de código abierto que contiene la implementación de varios algoritmos de aprendizaje basados en memoria, como IB1-IG, el cual es una implementación de un K-NN; IGTree, el cual es una aproximación de IB1-IG a un árbol de decisiones; y TRIBL, el cual es un híbrido de ambos casos [Daelemans et al., 2007]. Estos algoritmos serán utilizados para implementar clasificadores que permitan lematizar las palabras del shipibo-konibo.

Flask

Flask es un framework para programación web escrito en Python. Este no requiere otras herramientas o bibliotecas particulares para su funcionamiento por lo que se le considera como un micro framework [Ronacher, 2010]. Existen dentro de él diversas extensiones que permiten agregar características a la aplicación como si estuvieran implementadas dentro del mismo Flask, que facilitan el desarrollo de servicios web de manera rápida y sencilla.

Esta herramienta será utilizada para el desarrollo del servicio web resultante de este que contenga la herramienta de lematización final y los modelos previamente entrenados.

1.3.2 Métodos y procedimientos

Ripple down rules (RDR)

Son un tipo de reglas, que ofrecen una estructura ordenada de condiciones "si-entonces" y sus respectivas excepciones que a su vez también pueden ser nuevas reglas de este tipo, de modo que si una de las reglas se cumple, se procede a verificar si alguna de sus excepciones se satisface antes de determinar el resultado final, como se muestra de manera más clara en la Figura 1, en donde existe una regla para saber si se tiene un pájaro se puede determinar que vuela a menos que, como muestran las excepciones, este sea bebe, un pingüino o este en un avión.

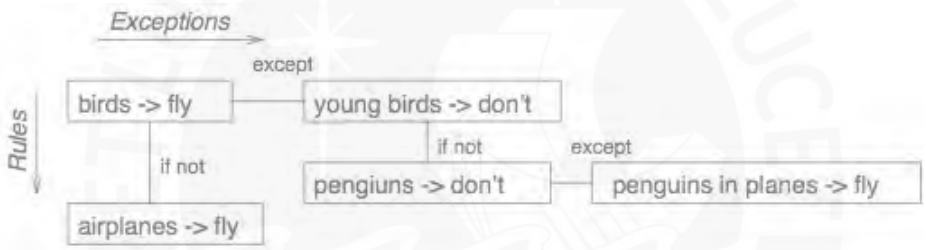

Figura 1: Ejemplo de reglas RDR. Tomado de [Scheffer, 1996 (Fig)] Algebraic foundation and improved methods of induction of ripple down rules

Mediante esta estructura se crea un concepto de localidad, debido a que cada excepción solo es aplicable al cumplimiento de la regla más cercana. Por ello, este tipo de reglas son fácilmente entendibles y modificables por el ser humano, y se pueden delimitar de manera adecuada a las reglas que en un inicio son demasiado generales, sin necesidad de alterar toda la estructura inicial [Scheffer, 1996].

1.4 Alcance

El proyecto se encuentra dentro de las áreas de las ciencias de la computación y la lingüística computacional, específicamente en el procesamiento de lenguajes naturales (PLN). Es un proyecto de investigación aplicada puesto que permitirá realizar un desarrollo

tecnológico y busca la implementación de una herramienta que permita obtener el lema de una palabra en shipibo-konibo en base a un diccionario de términos y reglas de derivación propias del lenguaje.

Además, forma parte del proyecto: 225-2015 FONDECYT: "Una plataforma de software para la traducción automática de textos entre lenguas originarias de la Amazonía peruana y español" siendo uno de los componentes a ser integrados a este proyecto de traducción automática, por lo que se cuenta con el apoyo de expertos en la lengua, así como expertos en el área de lingüística.

1.5 Limitaciones

Las principales limitaciones que posee este proyecto son:

- La disponibilidad del experto para la anotación del corpus de palabras de manera correcta en el tiempo que tome este proyecto, así como la variabilidad de las palabras que se encuentren dentro de este corpus.
- A pesar de que se trabaja sobre una base del idioma realizada por Pilar Valenzuela [Valenzuela, 2003], no existe una regularización lingüística oficial de la lengua que permita trabajar sobre un estándar preestablecido.

1.6 Riesgos

Los riesgos identificados se muestran en la Tabla 2:

Tabla 2: Riesgos del proyecto

Riesgo identificado	Impacto en el proyecto	Medidas correctivas
Demora en la formación del corpus de palabras anotadas.	Retraso al proyecto e impacto en los resultados finales debido a que son la base de este.	Establecer un cronograma de avances para el trabajo de anotación.
Problemas en la anotación de palabras.	Se pierda información importante de palabras desconocidas.	Contar con un hablante nativo que ayude en el trabajo.

1.7 Publicaciones

El presente trabajo se realiza para optar por el título de Ingeniero Informático de la Pontificia Universidad Católica del Perú y como parte de este se han realizado las siguientes publicaciones relacionadas durante su elaboración, las cuales se incluyen como anexos:

- ***Ship-LemmaTagger: Building a NLP Toolkit for a Peruvian Native Language***, **José Pereira-Noriega**, Rodolfo Mercado-Gonzales, Andrés Melgar, Marco Sobrevilla-Cabezudo y Arturo Oncevay-Marcos; Text, Speech, and Dialogue: 20th International Conference, TSD 2017. Donde se presentó los resultados derivados de la elaboración de este proyecto. (Anexo 1)

- ***ChAnot: An Intelligent Annotation Tool for Indigenous and Highly Agglutinative Languages in Peru***, Rodolfo Mercado, **José Pereira**, Marco Antonio Sobrevilla Cabezudo and Arturo Oncevay; Eleventh International Conference on Language Resources and Evaluation (LREC 2018). Donde se presentó la herramienta de anotación elaborada para la creación del corpus (ChAnot), así como su integración con el lematizador obtenido como resultado de uno de los objetivos de este proyecto. (Anexo 2)

CAPÍTULO 2: Marco Conceptual

Con el fin de entender mejor la problemática relacionada a la lematización de las lenguas originarias de la Amazonía peruana, particularmente en el caso del shipibo-konibo, se describirán ciertos conceptos utilizados en el desarrollo de este proyecto sobre las características del lenguaje y del problema en sí.

Asimismo, se explica qué es el procesamiento de lenguaje natural y uno de sus ámbitos, la lematización, y los principales enfoques utilizados en casos similares, así como las medidas que se emplean para verificar la labor de lematización.

2.1 Términos lingüísticos:

Lema o lexema:

Un lema es el término en común que puede agrupar a un conjunto de palabras distintas que han sido generadas a partir de una palabra con un significado irreducible, mediante diversos cambios flexionales [Gallmann, 1991] y derivaciones gramaticales. Por ejemplo, como se muestra en la Tabla 3, para las palabras flexionadas “Jacontani”, “Jaconbires” y “Jaconres”, su respectivo lema es “Jacon”.

Morfema:

Es la unidad mínima de la gramática la cual agrupa diversos conceptos del lenguaje como la raíz, el prefijo, el sufijo, entre otros. De este modo, un morfema es una subdivisión de la composición de las palabras. Estos se clasifican en morfemas libres, los cuales pueden ocurrir de forma independiente, y los morfemas enlazados, los cuales no pueden ocurrir sin estar junto a otros [Gallmann, 1991]. Por ejemplo, la palabra en shipibo-konibo “ainbobo”

que significa mujeres, está compuesta por dos morfemas, la raíz “ainbo” que corresponde a mujer y el sufijo “bo” que indica el plural.

Flexión:

El término flexión se refiere al proceso que sufre una palabra con el fin de transformarse en otra. Este proceso se da mediante un cambio morfológico o con la alteración de la categoría gramatical [Crystal, 2011]. Estos cambios se dan generalmente mediante el uso de sufijos, prefijos, infijos u otros cambios en la estructura de la palabra que agreguen un mayor significado a la palabra de origen. Por ejemplo, en el shipibo-konibo ocurre un caso de flexión cuando a una palabra se le agrega el sufijo “ibat” que denota una acción que sucedió el día anterior.

Lenguas de bajos recursos:

El término lengua de bajos recursos es una denominación amplia, que tiene diversas connotaciones según el contexto en que es empleado. También llamados lenguas minoritarias, lenguas menos usadas, lenguas menores o lenguas de escasos recursos [Forcada, 2006]. Para el caso de este proyecto, el término se refiere a aquellas lenguas que no tienen una presencia muy amplia en la internet, ni cuentan con los recursos necesarios para procesarlas de forma automatizada por computadoras [Forcada, 2006].

Asimismo, a veces estas lenguas no poseen un alfabeto computarizado que les permita generar software que las utilice, ni que se generen herramientas electrónicas como diccionarios, sistemas de recuperación de información o correctores ortográficos, que ayudan a la preservación de la cultura que la emplea [Streiter et al., 2006].

Lenguas aglutinantes:

Son las lenguas en los cuales la formación de palabras se da mediante la unión de morfemas independientes. Estos morfemas poseen límites claros dentro de las palabras debido a que no varían en su forma, y tienen un significado único, lo que los hace fácilmente identificables [Aikhenvald, 2007].

Por ejemplo, el shipibo-konibo tiene características aglutinantes ya que sus palabras son divisibles en una raíz y varios morfemas que le añaden significado. Esto se ve reflejado en palabras como "jacontani", "jaconbires" y "jaconres" [Faust, 1973], las cuales comparten la raíz "jaco" que significa "bien" y según los sufijos que se le agreguen varía su significado como se muestra en la Tabla 3.

Tabla 3: Significados de las derivaciones de "jaco"

Palabra	Raíz	Sufijo	Significado sufijo	Significado total
Jacontani	jacon-	-tani	apenas	un poco bien
Jaconbires	jacon-	-bires	completamente	muy bien
Jaconres	jacon-	-res	no más	bien no más

Shipibo-konibo:

El shipibo-konibo o (shipibo-conibo) es el idioma utilizado por el pueblo shipibo, el cual surge de la convergencia de tres etnias emparentadas de la Amazonía: los shipibo, los konibo y los xetebo. Esta es una lengua principalmente aglutinante con una gran variedad de afijos de distintas funciones, entre ellos los pronombres enclíticos, las posposiciones y los sufijos [Bismark, 2006], así como otras 10 clases gramaticales presentadas en la Tabla 4, identificadas por expertos en la lengua para la realización de este trabajo.

Tabla 4: Clases gramaticales del shipibo-konibo

Clase Gramatical	Detalle
Adjetivo	Palabra que describe a un sustantivo o nombre.
Adverbio	Palabra inalterable que permite modificar a un adjetivo, verbo, adverbio u otras clases gramaticales.
Nombre	Palabra que sirve para designar los seres vivos o las cosas materiales o mentales.
Verbo	Palabra central del predicado que permite describir las acciones o estados del sujeto de la oración.
Conjunción	Palabra que permite unir diversas partes de una oración, e incluso múltiples oraciones.
Determinante	Palabra que permite darle significado y referencia a un nombre dentro de una frase.
Interjección	Palabra que por sí sola permite expresar un sentimiento o un estado de ánimo.
Pronombre	Palabra que permite sustituir los nombres comunes o propios de forma genérica.
Palabras interrogativas	Palabra funcional usada para formular interrogativas parciales
Postposición	Palabra invariable que sigue el llamado sintagma preposicional.
Enclítico	Palabra que se une al verbo precedente para formar una sola palabra
Sufijo	Afijo que se añade al final de una palabra
Prefijo	Afijo que se añade al inicio de una palabra

2.2 Términos relacionados al problema:

Procesamiento de lenguaje natural (NLP):

El procesamiento de lenguaje natural o por sus siglas en inglés NLP (Natural Language Processing) es el campo de la inteligencia artificial y la lingüística computacional que se encarga del desarrollo y estudio de técnicas computacionales para el análisis automático y

representación del lenguaje humano [Cambria; White, 2014], con el fin de obtener aplicaciones útiles para la interacción entre las computadoras y el lenguaje natural.

Entre sus principales labores se encuentran [Alcina; Valero, 2009]:

- Procesamiento digital de voz y reconocimiento del habla.
- Extracción de la información.
- Búsqueda y recuperación de información.
- Desambiguación lingüística.
- Etiquetado gramatical.
- Traducción automática.
- Creación de tokenizadores.
- Creación de lematizadores.

En este proyecto abordaremos la construcción de un lematizador para apoyar la labor de la traducción automática, el cual se encarga de la reducción de palabras flexionadas a una forma más simple mediante el uso de diversas técnicas.

Aprendizaje de Máquina

El aprendizaje de máquina o aprendizaje automático es el campo de estudio científico que se enfoca en el desarrollo y uso de algoritmos de inducción con el fin de generar un descubrimiento de conocimiento [Kohavi; Provost, 1998]. De esta forma, se busca construir modelos automáticos que aprenden de forma iterativa a partir de datos y que generen conocimiento.

Dentro de su aplicación, existen diversos escenarios según el tipo de datos que se posea para el aprendizaje y uno de ellos es el aprendizaje supervisado, el cual consiste en el uso de ejemplos de entrenamiento con su respectivo resultado, lo que permite predecir el resultado para nuevos datos [Mohri et al, 2012].

K-vecinos más cercanos (K-NN)

El algoritmo de k-vecinos más cercanos o K-NN es un método de clasificación supervisada que permite estimar la clase de un nuevo caso a clasificar mediante la comparación de sus características con las de casos previamente clasificados, dando como resultado la clase más frecuente a la que pertenecen sus K vecinos más cercanos [Laaksonen; Oja, 1996]. Asimismo, dependiendo del número de vecinos establecido en el análisis se puede obtener un ranking de clases resultantes de acuerdo a la cantidad de vecinos pertenecientes a cada una de dichas clases.

Lematización:

La lematización es el proceso lingüístico que consiste en obtener el lema o forma canónica de una palabra a partir de su forma flexionada; es decir, la eliminación de toda derivación realizada en la palabra mediante reglas de conjugación o unión de morfemas derivativos y flexivos, con el fin de obtener la forma simple y general de la palabra que se encuentra en un diccionario [Liu, 2011] y represente a todas sus formas flexionadas [Guerra, 2003]. Por ejemplo, para las palabras “comimos, comieron y comiendo” corresponde el lema “comer”, de modo que se redujeron todas estas conjugaciones a un lema en común, el cual es el verbo en infinitivo.

Dentro del proceso de lematización existen dos tipos de niveles posibles:

- **Lematización morfológica:** Esta es la forma más simple de lematización, debido a que solo se analizan las palabras de forma independiente sin importar el contexto en que se encuentren. Por lo tanto, este nivel puede generar dos lemas distintos a una sola palabra debido a casos de homonimia, como sucede en el caso de “ama” que posee de lemas al verbo “amar” y al sustantivo “amo”.

- **Lematización sintáctica:** En esta forma las palabras son analizadas junto a su contexto de modo que es posible identificar su respectiva categoría gramatical y significado dentro de la oración o frase en que se presenten y obtener un solo lema de acuerdo a estas características.

En el campo de la informática, la lematización es un proceso muy utilizado dentro del área de procesamiento del lenguaje natural. En su utilización, se aplican diversos enfoques para realizarla, siendo principalmente los siguientes:

- **Basado en reglas:** En este enfoque se busca crear algoritmos que reflejen las reglas gramaticales de un idioma, a través de una serie de pasos que permitan transformar las diversas derivaciones de una palabra para obtener su lema. Generalmente, la construcción de reglas se desarrolla de acuerdo a categorías gramaticales y tomando en cuenta las posibles excepciones que ocurren en la lengua, de modo que siempre se obtiene algún resultado para cualquier entrada. Sin embargo, debido a las constantes variaciones de las lenguas, la gran presencia de extranjerismos y las múltiples excepciones a las reglas, este método resulta impreciso para algunas palabras [Loponen; Järvelin, 2010].

- **Basado en diccionario:** Este enfoque busca obtener de antemano un diccionario de correlaciones entre los lemas y sus diferentes formas flexionadas, con el fin de solo reemplazar las palabras por su respectivo lema. A pesar de que este método resulte muy bueno para el caso de excepciones a la regla o palabras raras, sus resultados están muy ligados a la calidad del diccionario utilizado, y dado a que la mayoría de los lenguajes poseen una cantidad extensa de derivaciones y se encuentran en constante evolución [Delmonte, 2013], es imposible obtener un diccionario que comprenda todas las palabras de un lenguaje, por lo que su desempeño no es bueno para el caso de corpus de palabras con temas muy variados.

- **Híbrido:** Este enfoque se encarga de mezclar diversas partes de los otros dos métodos con el fin de obtener un mejor resultado [18], orientándose mayormente hacia uno de los enfoques y tomando partes de otros. Por ejemplo, algunos lematizadores trabajan mediante el uso de reglas pero utilizan diccionarios para tratar las excepciones.

Estos enfoques se pueden realizar por medio del trabajo manual generado a partir del propio conocimiento y análisis de la lengua o a través del uso técnicas estadísticas de aprendizaje de máquina que infieran las reglas de clasificación gracias al análisis de corpus anotados con sus respectivos lemas y opcionalmente ayuda humana [Loponen; Järvelin, 2010]. Sin embargo, la creación de este corpus es una labor muy ardua debido a que estos deben ser generados de manera manual [Forcada, 2006].

Corpus:

Es un conjunto computarizado de textos, generalmente verificados y provisto de anotaciones, los cuales han sido seleccionados con un criterio en específico para representar las peculiaridades de un lenguaje o algún subgrupo de este [Tognini-Bonelli, 2001], y es usado como punto de entrada para el desarrollo de diversas herramientas NLP.

Universal Dependencies:

La iniciativa de Universal Dependencies (UD) busca desarrollar un estándar de anotación lingüística coherente para ser aplicada a muchos idiomas con el fin de crear anotaciones morfológicas uniformes [Pyysalo et al., 2015]. Por este motivo, al adaptar el estándar al shipibo-konibo, se definieron 16 categorías de palabras, las cuales son: adjetivos, adverbios, conjunciones, determinantes, interjecciones, sustantivos, nombres propios,

numerales, posposiciones, pronombres, puntuaciones, símbolos, verbos, verbos auxiliares, palabras interrogativas y onomatopeyas.

De este modo, se busca facilitar la interacción y reutilización de los avances en el área de la lingüística computacional en varios idiomas, así como mejorar la comparabilidad de resultados y facilitar el surgimiento de nuevos enfoques en el análisis sintáctico automático [Pyysalo et al., 2015].

2.3 Medidas de distancia:

Debido a la necesidad de utilizar diversas medidas de distancia para comparar las características utilizadas en el algoritmo K-NN, es necesario elegir cuáles se utilizarán para contrastar los valores obtenidos. A continuación, se detallan en mayor profundidad las medidas elegidas.

Distancia de Hamming

Es una medida de distancia utilizada en vectores. Se define como la cantidad de coordenadas diferentes que existen entre los elementos de dos vectores [MacKay; David, 2003].

Modified value difference (MVDM)

Esta medida se basa en el cálculo de la similitud de dos valores mediante la comparación de ellos con su coocurrencia en su clase correspondiente, de modo que no se tiene una clasificación binaria de los elementos a comparar [Daelemans et al., 2007]. Su cálculo se da mediante la sumatoria de los valores de la distribución condicional de las clases C como se muestra en la siguiente fórmula:

$$\delta(v1, v2) = \sum_{i=1}^{n} |P(C_i|v1) - P(C_i|v2)|$$

Distancia de Levenshtein

La distancia de Levenshtein o “edit-distance”, es una medida de distancia entre palabras, la cual se determina mediante la cantidad mínima de inserciones, eliminaciones y sustituciones de caracteres requeridas para transformar una palabra en otra [Ristad; Yianilos, 1998].

Coeficiente de Dice

Esta medida determina la superposición de pares de caracteres o bigramas de dos palabras diferentes [Daelemans et al., 2007], agrupando los caracteres adyacentes en conjunto de pares relacionados. Su valor se obtiene mediante la cantidad de bigramas únicos de las palabras a analizar entre el número total de bigramas de ambas palabras como se muestra en la ecuación:

$$\delta(x_i, y_i) = 1 - \frac{2n_{x_i \cap y_i}}{n_{x_i} + n_{y_i}}$$

Distancia Euclidiana

Es la distancia de separación ordinaria entre dos elementos de un espacio euclidiano, los cuales en este caso son representados por elementos de diferentes vectores [Deza; Deza, 2009]. Su cálculo se da mediante la fórmula siguiente:

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_n)}$$

Distancia de Chebyshev

Es una medida de distancia definida como la máxima distancia entre todos los pares coordenados de un vector en cualquiera de sus dimensiones [Deza; Deza, 2009].

2.4 Medidas de evaluación:

2.4.1 Medidas tradicionales:

Con el fin de determinar qué tan bien se desempeña el proceso de lematización es necesario utilizar medidas de rendimiento como la precisión. Ella se obtiene comparando los lemas resultantes del proceso con los lemas esperados, los cuales han sido obtenidos de manera previa mediante un proceso de anotación manual. A continuación, se detalla en mayor profundidad esta medida:

Precisión

Es la relación entre el número de lemas que han sido obtenidos de manera correcta frente al número total de lemas identificados [Karashtranova et al., 2015].

2.4.2 Otras Medidas:

Edit-distance

Debido a que en un proceso de lematización solo existen dos posibles resultados (la obtención del lema correcto o la obtención de un lema incorrecto), no es fácil analizar qué tan cerca se estuvo de obtener un resultado correcto. Sin embargo, existen diversas formas de analizar la similitud entre palabras, las cuales se pueden emplear para calcular la proximidad entre los resultados obtenidos erróneamente y los resultados esperados. Por ello, este proyecto usará la distancia de Levenshtein [Levenshtein, 1966] para determinar esta similitud.

Con esta distancia, se cuenta la cantidad de inserciones, eliminaciones y sustituciones de caracteres, lo cual se puede ilustrar mediante el ejemplo de las palabras “shipibo” y “conibo”, en donde la distancia es 4 debido a que es necesario realizar 3 reemplazos entre las letras “s”, “h” e “i” por “c”, “o” y “n”, y eliminar la letra “p” para transformar la palabra.

CAPÍTULO 3: Estado del Arte

En este capítulo se realizará una revisión de la literatura relevante sobre el tema para conocer el estado actual de la lematización de lenguas de escasos recursos. Esto ayudará a entender mejor los avances relacionados al problema planteado y permitirá conocer más sobre las posibles soluciones, dificultades y criterios a tomar en cuenta para lograr una alternativa de solución al caso que se trata en este proyecto.

3.1 Objetivos de la revisión

La siguiente revisión tiene los siguientes objetivos:

- Conocer las técnicas de lematización más usadas en el caso de lenguajes con bajos recursos y en especial de lenguajes con estructuras símiles al shipibo-konibo.
- Conocer los problemas que surgieron para la lematización de lenguajes con bajos recursos y cuáles fueron las posibles soluciones planteadas.
- Conocer el alcance de otros proyectos desarrollados y analizar para qué ámbitos sus resultados son pertinentes.
- Identificar si existe la posibilidad de adaptabilidad y reutilización de las soluciones obtenidas hacia otras lenguas.

3.2 Método usado en la revisión del estado del arte

En esta revisión se usaron las publicaciones disponibles en la base de datos SCOPUS [1]y en la ACM Digital Library[2]. Estos artículos deberán incluir en su título o resumen las palabras “lemmatization” o “lemmatizer”, así como ser referentes a los lenguajes con escasos recursos o de estructura similar al shipibo-konibo. Asimismo, deberán pertenecer al área de

[1] Disponible en https://www.scopus.com/home.uri
[2] Disponible en https://dl.acm.org/

ciencias de la computación y no poseer una antigüedad mayor al año 2009 para solo considerar los últimos avances realizados para solucionar el problema.

Estas condiciones se plasmaron en la siguiente cadena de búsqueda: “TITLE-ABS-KEY(lemmatization) OR TITLE-ABS-KEY(lemmatizer) AND PUBYEAR > 2009 AND (LIMIT-TO (SUBJAREA,"COMP”))”, la cual se ejecutó en las base de datos SCOPUS y mediante el uso de filtros para el caso de ACM Digital Library.

Como criterio de inclusión se tomó los artículos que trataban sobre lenguas minoritarias y como criterio de exclusión a los trabajos que solo utilizaban algún método de lematización existente para realizar otra tarea. De este modo, se obtuvieron 128 resultados, de los cuales se descartaron 119 que no cumplían con los criterios establecidos.

3.3 Estado del arte

3.3.1 Lematizadores a partir de reglas:

A Dictionary- and Corpus-Independent Statistical Lemmatizer for Information Retrieval in Low Resource Languages [Loponen; Järvelin, 2010]:

En este proyecto se presenta un lematizador estadístico basado en reglas creadas a partir de un conjunto de pares de palabras y sus lemas respectivos (StaLe), los cuales deben ser representativos del idioma y haberse extraído de textos reales con el fin de reflejar la distribución propia del lenguaje.

Se utiliza un método basado en reglas de transformación similar al propuesto por Pirkola [Pirkola et al., 2003], las cuales consisten en una cadena de entrada y de salida, la posición de la regla y el valor estadístico de la regla, el cual consiste en el número de veces que aparece esta regla en el corpus de entrenamiento, y que tan común es esta para todos los casos posible a evaluar en donde esta sea posible de aplicar.

El trabajo se realizó de esta forma con el fin de ser reutilizado en otros lenguajes, especialmente en los de bajos recursos, debido a su independencia de diccionarios, y sin que se generen problemas por las palabras fuera del vocabulario. Sin embargo, dado que funciona mediante un proceso estadístico, algunas veces se puede obtener reglas que generen resultados incorrectos.

Para la obtención del lema correspondiente, una vez que las reglas han sido generadas, se analiza todas las reglas que pueden aplicarse a la palabra de entrada y mediante los valores estadísticos se determina cuál es la correcta. En caso de que ninguna regla sea aplicable para el caso, el lema resulta igual a la palabra de entrada.

Al evaluar este lematizador en los idiomas inglés, finés, sueco y alemán, se obtuvieron resultados muy cercanos a los del lematizador comercial basado en reglas y diccionarios TWOL [Koskenniemi, 1983], el cual es usado como base de comparación para los resultados, destacando el caso del alemán, donde los resultados de acierto fueron un 4.59% mayores (88.62%), los cuales se obtuvieron entrenado el lematizador con solo 16,086 pares de palabras.

A lemmatization method for Mongolian and its application to indexing for information retrieval [Khaltar; Fujii, 2009]:

En este trabajo se busca lematizar los verbos, sustantivos, adjetivos y numerales del idioma mongol, mediante el uso de dos conjuntos de reglas elaboradas manualmente, uno de 179 para los verbos y otro de 196 reglas aplicables al resto de categorías, llamado grupo sustantivo debido a la similitud de sufijos entre los sustantivos y el resto de las categorías, así como con el uso de un diccionario de 1,254 verbos. Asimismo, se incluyen tres clases de reglas para determinar si la palabra a tratar es prestada de otro idioma.

El proceso que se usa para obtener el lema de una palabra es primero analizarla como si fuese un verbo, aplicando todas las transformaciones que le correspondan y finalmente buscando el lema obtenido en el diccionario de verbos. Si el lema no está presente en el diccionario, se procede a analizar la palabra como un sustantivo.

La razón de este proceder es que existen sufijos comunes presentes en todas las palabras, por lo que se toma primero el caso de los verbos, que puede ser de cierta forma verificado y luego se procede con el caso de los sustantivos, que es el más general.

Del mismo modo, la elección de un diccionario de verbos sobre el de otra categoría se debe a que la probabilidad que existe de que se generen nuevos verbos en el idioma es mucho menor que la de generar nuevos sustantivos, así como el tamaño que tendría cada uno de los diccionarios.

Los resultados fueron evaluados en dos corpus distintos, uno de 183 artículos de periódicos tomados del portal web de recopilación de artículos periodísticos ‘‘Olloo”, que contienen 65,900 *tokens*, y uno de 1,102 resúmenes técnicos tomados del ‘‘Mongolian IT Park” que contienen 178,448 *tokens*. Se obtuvo mejores resultados que los lematizadores existentes, en especial para los sustantivos, en donde se obtuvo una precisión de 89.5% para los periódicos y 92.5% para el otro corpus.

Prototype-based Active Learning for Lemmatization [Daelemans et al., 2009]:

En este trabajo se investiga el impacto de la selección de los lemas de entrenamiento en la efectividad de un lematizador del idioma afrikáans basado en reglas generadas de forma automática mediante un algoritmo de k-vecinos más cercanos.

Para determinar cuáles son los datos de entrenamiento más adecuados se basan en la técnica de clasificación activa basada en prototipos (PBAC). Sin embargo, a diferencia de su uso común, en donde los datos más representativos o comunes son los más adecuados, se

plantea que para el caso de un lematizador es más importante tomar los casos menos comunes, los cuales se determinan por su longitud, su frecuencia en un corpus y su entropía.

Los datos de entrada se clasificaron en 271 clases de acuerdo a la sub-cadena común más larga que poseían, y con esos grupos se generaron las reglas de transformación, las cuales fueron experimentadas en un corpus de 72,226 palabras mediante una validación cruzada de 10 iteraciones.

Como resultado, se observó que mediante el uso de palabras poco representativas se podía reducir la cantidad necesaria de datos de entrenamiento del lematizador, determinando que para el caso de una exactitud de 89% se necesitaron 19,864 datos de los menos representativos del corpus frente a los 24,656 que se necesitaron para obtener el mismo resultado en el caso de una selección aleatoria.

Comparison of String Distance Metrics for Lemmatisation of Named Entities in Polish [Piskorski et al. 2007]:

En este proyecto se busca lematizar los diferentes nombres propios en polaco mediante el uso de diversas medidas de distancias entre palabras y evaluación de ellas con el fin de analizar cuál es la más adecuada para cada subconjunto de palabras (países, entidades y nombres de personas), mediante su aplicación a reglas.

Asimismo, se clasificaron las diferentes técnicas en subgrupos y se determinó cuáles son los que funcionan mejor para cada grupo de palabras y qué tan difícil es realizar cada uno de ellos.

A partir de todas las combinaciones de medidas se obtuvieron unos pocos cientos de ellas y se observó que las medidas de distancia entre palabras derivadas de las de Jaro, Smith-Waterman, prefijo-común o cadena común más larga, dieron un mejor desempeño frente al resto de medidas con las que se experimentó.

De este modo, mediante los resultados obtenidos para el polaco, se espera que estos puedan ser replicados para otros idiomas con una estructura altamente flexiva, similar al idioma con el que se realizaron los análisis.

***UTA Stemming and Lemmatization Experiments in the FIRE Bengali Ad Hoc Task [L*oponen et al., 2013]:**

En este proyecto se analizó la eficiencia de tres normalizadores, un stemmer y dos lematizadores, GRALE and StaLe, para la tarea de recuperación de información en el lenguaje bengalí.

StaLe es un lematizador estadístico basado en reglas diseñado para operar en lenguajes de escasos recursos. Para el caso de la adaptación al bengalí, fueron necesarios 11,000 sustantivos aleatorios tomados de un corpus con sus respectivos lemas, los cuales se generaron manualmente, obteniendo resultados similares a un lematizador competitivo (entre 88%-108% de los resultados obtenidos con el lematizador comercial TWOL [Koskenniemi, 1983]).

En este experimento se evaluó su precisión mediante la relevancia de los resultados obtenidos al consultar un conjunto de 120,347 documentos con 50 consultas diferente, dando como resultado para el mejor de los casos un MAP (*Mean Average Precision*) de 0.5058.

Utilizing Vector Models for Automatic Text Lemmatization [Gallay; Šimko, 2016]:

En este proyecto de lematización para el eslovaco se mencionan los problemas que existen para generar lematizadores de lenguajes altamente flexivos como el analizado. En el caso de un enfoque de reglas, debido a la gran cantidad de reglas de derivación que presenta el lenguaje y sus múltiples excepciones, esta labor resulta inviable. En el caso de un enfoque

por diccionarios, dado que la generación automática de este no es una tarea sencilla, se utilizan diccionarios generados de forma manual o semiautomática, lo que tampoco es viable.

Con el fin de evitar los problemas mencionados, se plantea el uso de modelos de espacio vectorial para realizar la lematización de forma automática, de forma que se posean operaciones de desplazamiento vectoriales capaces de generar los lemas correctos mediante la caracterización de las variaciones morfológicas, obteniéndose operaciones de forma similar a: vector(rey) − vector(hombre) + vector(mujer) = vector(reina)[3], y pares de referencia generales entre las palabras y sus respectivos lemas de acuerdo a las categorías gramaticales.

Esta labor se divide en cuatro partes. La primera es la selección de los pares de referencia más relevantes para el caso, para lo que se analiza el sufijo común de mayor tamaño, la similitud coseno y las categorías gramaticales presentes en la palabra y en los pares de referencia. La segunda es la obtención de los posibles lemas basados en lo pares de referencia anteriores mediante su respectivo desplazamiento vectorial. La tercera es la del cálculo de los pesos de los lemas anteriores mediante el uso de varias técnicas como la distancia de Levenshtein, la de Jaro-Winkler o la de similitud de prefijos. La cuarta y última es la selección del lema más apropiado basado en los resultados anteriores mediante la suma normalizada de los pesos de acuerdo al lema común obtenido por distintos pares de referencia.

En el mejor de los casos, se obtuvo un acierto de hasta un 80%. Sin embargo, el corpus usado para el entrenamiento fue muy grande (829’771,945 *tokens*), lo que impactaría en el caso de un lenguaje con menos recursos.

3 De esta forma, al utilizar vectores de grandes dimensiones, se pueden aislar ciertas características morfológicas de las palabras en dimensiones independientes, las cuales pueden ser utilizadas en operaciones vectoriales para encontrar similitudes entre palabras distintas. Para un mayor detalle ver la publicación de T. Mikolov [Mikolov et al., 2013].

LemmaGen: Multilingual Lemmatisation with Induced Ripple-Down Rules [Juršic et al, 2010]:

En este trabajo se evalúa el desempeño del lematizador LemmaGen para 12 idiomas europeos, el cual emplea el aprendizaje de máquina para generar reglas de tipo Ripple-Down (RDR), las cuales se utilizan para evaluar las palabras de acuerdo a los sufijos que poseen.

Cada regla posee la forma de: "RDR ::= SI condición ENTONCES resultado EXCEPTO excepciones", y en conjunto forman una estructura similar a la de un árbol de decisiones, en donde primero se establecen los casos más generales y según se encuentren excepciones a estos casos se van agregando a la estructura de forma iterativa, de modo que se obtiene un conjunto de reglas fácilmente entendible y altamente eficiente.

Debido a que las reglas son generadas automáticamente a partir de un corpus de palabras, lemas y opcionalmente las respectivas categorías gramaticales de la palabra, el lematizador es capaz de ser reutilizado para diferentes lenguas.

En las pruebas realizadas para los 12 idiomas, destaca el caso del serbio y del búlgaro, en donde a partir de 20,294 y 55,200 palabras respectivamente, se obtuvo para el caso sin utilizar las categorías gramaticales un 97.9% y 93.7% de efectividad sobre el conjunto de entrenamiento, y 65.3% y 70.4% respectivamente sobre un conjunto de prueba.

Asimismo, al momento de tomar en cuenta las categorías gramaticales se obtuvo un 99.8% y 99.5% de efectividad sobre el conjunto de entrenamiento, y 86.1 % y 94.1% respectivamente sobre un conjunto de prueba.

En ambos casos, los resultados fueron superiores al lematizador utilizado como referencia (CST) y entre los 12 idiomas probados, el búlgaro se ubica en el puesto 7 de efectividad y el serbio en el 12, lo cual se debe presuntamente al tamaño de corpus utilizado.

3.3.2 Lematizadores a partir de diccionarios:

***UTA Stemming and Lemmatization Experiments in the FIRE Bengali Ad Hoc* Task [Loponen et al., 2013]:**

GRALE es un lematizador basado en grafos hecho especialmente para el bengalí, en donde primero se genera una lista de posibles sufijos de un corpus mediante el uso de n-gramas y luego se determina la correctitud de estos por un experto en la lengua. Posteriormente, se relacionan las palabras mediante grafos dirigidos según un nodo pueda generar a otro mediante el uso de uno de los sufijos determinados, y en donde los nodos con grados de entrada igual a 0 son tomados como lemas.

Sin embargo, para casos en que el lema no está presente en el corpus de entrenamiento, se pueden generar errores dado a que no se logra encontrar una derivación adecuada, lo que sucede raramente en el idioma bengalí dada su naturaleza de poseer una cantidad limitada de lemas.

En este experimento se evaluó su precisión mediante la relevancia de los resultados obtenidos al consultar un conjunto de 120,347 documentos con 50 consultas diferente, dando como resultado para el mejor de los casos un MAP (*Mean Average Precision*) de 0.5001.

3.3.3 Lematizadores híbridos:

Design & Development of a Rule Based Urdu Lemmatizer [Gupta et al., 2016]:

En este proyecto se desarrolló un lematizador para una lengua hablada en la India y Pakistán, el urdu, el cual se basa en reglas y un diccionario de palabras excepcionales a las reglas, en donde no existe la necesidad de retirar ningún sufijo.

Se utilizaron 128 reglas generadas a partir de 128 sufijos extraídos de 250,000 palabras, las cuales se encargan de eliminar el sufijo correspondiente y añadir alguna letra a la palabra

en caso sea necesario. Asimismo, las palabras que no cumplían las reglas de remoción de sufijos fueron añadidas a un diccionario de excepciones.

En el flujo utilizado en el análisis de las palabras, lo primero fue verificar si esta se encuentra presente en el diccionario y en caso contrario se procede al análisis de ella mediante las reglas. Si ninguna regla resulta pertinente para el caso, se determina la palabra de entrada como el lema final.

Al evaluar el lematizador en un conjunto de 1,000 palabras, se obtuvo una precisión de 90.3%.

Hybrid Lemmatizer for Estonian [Tkachenko et al., 2014]:

En este trabajo de lematización para el estonio, se utiliza un método híbrido en el cual se posee un diccionario de términos con sus respectivos lemas y un conjunto de reglas con el fin de obtener resultados para palabras fuera del vocabulario.

Para lematizar cada palabra primero se procede a buscar si esta existe en el diccionario. En caso exista más de un resultado, se determina el lema mediante una función clasificatoria que emplea un modelo de canal ruidoso en donde se analiza la frecuencia del lema en los datos de entrenamiento. En cambio, si no existe resultado alguno se procede a generar todos los posibles lemas mediante las reglas y a partir de la función clasificatoria se determina el más adecuado.

Bajo este enfoque, se obtuvo una exactitud de 91%, usando como datos de entrada un corpus de 513,000 *tokens* y un diccionario generado a partir de 100,000 lemas, y dividiéndolos en ⅔ para el entrenamiento y ⅓ para pruebas.

3.4 Resumen:

Tabla 5: Resumen de estudios revisados sobre lematización

Estudio	Idioma	Tipo	Corpus	Resultado
A Dictionary- and Corpus-Independent Statistical Lemmatizer for Information Retrieval in Low Resource Languages	Inglés, alemán, sueco y finés	Reglas	16,086 palabras para el alemán	88.62% para el alemán
A lemmatization method for Mongolian and its application to indexing for information retrieval	Mongol	Reglas	Dos corpus de 65,900 y 178,448 palabras	89.5% y 92.5% de precisión por corpus
Prototype-based Active Learning for Lemmatization	Afrikáans	Reglas	19,864 palabras de un corpus de 72226	89% de efectividad
Comparison of String Distance Metrics for Lemmatisation of Named Entities in Polish	Polaco	Reglas	8 sets de entre 5,941 y 1,000 palabras	33% a 98% dependiendo de la clase y método usado
UTA Stemming and Lemmatization Experiments in the FIRE Bengali Ad Hoc Task	Bengalí	-GRALE: Diccionario -StaLe: reglas	-Corpus de 123,047 documentos de unas 362 palabras -11,000 palabras al azar del corpus	-MAP de 0.5001 -MAP de 0.5058
Utilizing Vector Models for Automatic Text Lemmatization	Eslovaco	Otro / Reglas	829’771,945 *tokens*	80%. de correctitud
LemmaGen: Multilingual Lemmatisation with Induced Ripple-Down Rules	12, entre ellos serbio y búlgaro	Reglas	20,294 para el serbio, 55,200 para el búlgaro	86.1% para el serbio, 94.1% para el búlgaro
Design & Development of a Rule Based Urdu Lemmatizer	Urdu	Híbrido	250,000 palabras	90.3% de precisión sobre 1000 palabras

Estudio	Idioma	Tipo	Corpus	Resultado
Hybrid Lemmatizer for Estonian	Estonio	Híbrido	513,000 *tokens* y 100,000 entradas para el diccionario	91% de exactitud

3.5 Conclusiones:

Al implementar un lematizador, se utilizan 3 técnicas principales: el uso de diccionarios de términos, el uso de reglas de derivación o una combinación de ambas. Para el caso de lenguas de escasos recursos o de lenguas altamente flexionadas se observa que el método más utilizado es el de reglas.

Asimismo, debido a la escasez de datos y de personas expertas en el lenguaje, la generación de las reglas se hace de forma automática o semi-automática mediante algoritmos de aprendizaje y técnicas estadísticas, evitando de esta manera la necesidad de un gran esfuerzo en la recopilación de los datos necesarios que puede presentar cada lengua en particular para generar estas reglas.

Finalmente, a pesar de las limitaciones, se han obtenido resultados aceptables en la generación de lematizadores con corpus pequeños, e incluso, existe la posibilidad de reutilización de ciertas herramientas y métodos dada la forma común de derivación que poseen algunos lenguajes.

CAPÍTULO 4: Metodología

Este capítulo tiene como finalidad explicar la metodología seleccionada, KDD o Knowledge Discovery in Databases, para realizar cada uno de los objetivos específicos de este proyecto, detallando los pasos que se siguieron para su realización.

El término "KDD" se usa para referirse al proceso general de descubrimiento de conocimiento útil a partir de datos, el cual consiste por lo general de los siguientes cinco pasos, mostrados en la Figura 2: selección de datos, preprocesamiento, transformación, minería de datos e interpretación de los resultados [Fayyad et al., 1996].

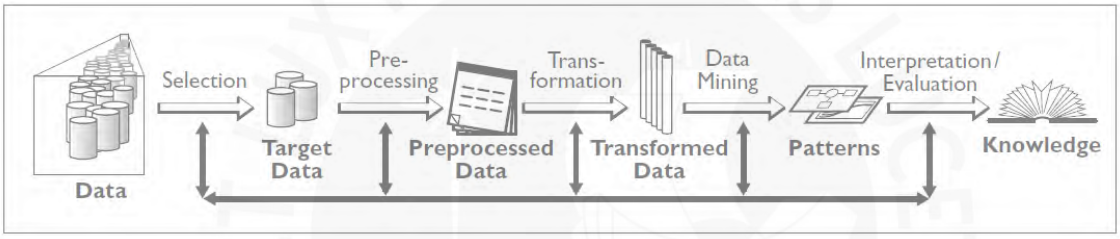

Figura 2: Proceso KDD. Tomado de [Fayyad et al., 1996] The KDD process for extracting useful knowledge from volumes of data

En el caso de este proyecto, cada uno de los pasos corresponde a una parte de la metodología, teniéndose la siguiente relación:

1. Selección de datos: Corresponde a la fase de elaboración de los corpus de palabras anotadas con sus respectivos lemas y categorías gramaticales.
2. Preprocesamiento: Corresponde a la fase de verificación de los corpus anotados por los expertos, observando si en ellos no existen inconsistencias o datos indebidos, y que para todos los casos exista su lema y categoría correspondiente.
3. Transformación: Corresponde a la fase de generación de clases y características necesarias para ejecutar cada uno de los algoritmos de clasificación.

4. Minería de datos: Corresponde a la fase de entrenamiento de los algoritmos, a la generación de los clasificadores resultantes y a su integración.

5. Interpretación de resultados: Corresponde a la etapa de verificación de los datos obtenidos frente a los resultados dados por los casos de prueba.

4.1 Selección de datos

Dada la necesidad de tener un conjunto de palabras para el entrenamiento y generación de los clasificadores, se procedió a anotar el lema y la categoría gramatical de las palabras presentes en un corpus de oraciones tomadas de un diccionario de lengua shipibo [Lauriout et al, 2003] y de ejemplos presentes en diversos textos educativos de las series "Naturaleza y Vida Social" [Ministerio de Educación Pública, 1960][Ministerio de Educación Pública, 1963] y "Quirica" [Ministerio de Educación, 1982][Instituto Lingüístico de Verano, 1979], elaborados por el Ministerio de Educación y el Instituto Lingüístico de Verano, así como del libro "Quimisha Incabo ini yoia (Leyendas de los shipibo-conibo sobre los tres Incas)" [Bardales Rodríguez, 1979].

Esta labor fue realizada por expertos en la lengua y mediante el uso del anotador gramatical ChAnot, el cual contiene las categorías gramaticales del shipibo-konibo que fueron previamente alineadas a las presentes en las Universal Dependencies. Esto permitió que las anotaciones de las características morfosintácticas de las palabras queden almacenadas en archivos de formato XML, como los mostrados en el Anexo 3, en donde se posee de manera jerárquica las oraciones que han sido anotadas y las características de cada uno de sus elementos.

De esta tarea se ha logrado anotar hasta la fecha un conjunto de 15,477 palabras con sus respectivos lemas y categorías, pero esta es una labor que continuará hasta la finalización del

proyecto de traducción automática (225-2015 FONDECYT) de la cual este proyecto de fin de carrera es parte. Estas palabras fueron utilizadas para los siguientes pasos.

4.2 Preprocesamiento

En esta fase se procedió a procesar los archivos XML del corpus para obtener el conjunto de datos necesarios para el entrenamiento de los clasificadores, lo que consistió en transformar el formato del archivo original al formato de corpus requerido para este trabajo, los cuales se pueden observar en el Anexo 4, en donde cada línea presenta una palabra, su lema y su categoría gramatical, las cuales servirán para formar las clases de manera automática y para obtener las características necesarias de cada palabra.

Del mismo modo, se descartó los datos no consistentes e innecesarios, que en este caso fueron las anotaciones de signos de puntuación, las palabras que no poseían un lema, las palabras que contenían errores (denotados por la palabra "ERROR" en la anotación), los números cardinales y los conjuntos de palabras-lema repetidos, obteniendo de esta forma un conjunto de 6844 anotaciones distintas y sin errores a partir de las 15,477 iniciales.

4.3 Transformación

En esta fase se procedió a transformar el corpus en datos coherentes con los formatos necesarios para los algoritmos de clasificación, para lo que se realizaron dos labores: la generación de clases, las cuales son las reglas de transformación que se le aplicará a cada palabra para obtener su respectivo lema, y la selección de características, las cuales servirán de datos de entrada que permitan clasificar cada palabra en una regla respectiva.

4.3.1 Generación de clases

A partir del corpus generado con el formato presente en el Anexo 4, se procedió a la generación de las clases a utilizar en los clasificadores.

Para esto se utilizó un formato particular de anotación, en donde se muestra cuál es la parte que se le debe quitar a la forma flexionada y la parte que se le debe agregar a la palabra para formar el lema. De este modo, existen cuatro tipos de reglas posibles:

- Las que remueven una parte de la palabra y agregan otra: Por ejemplo, para el par palabra-lema (oxawe, oxati), la regla correspondiente tiene la forma de “we>ti”.
- Las que solo remueven una parte de la palabra: Por ejemplo, para el par palabra-lema (miara, mia), la regla correspondiente tiene la forma de “ra>”.
- Las que solo agregan una parte a la palabra: Por ejemplo, para el par palabra-lema (awi, awin), la regla correspondiente tiene la forma de “>n”.
- Las que no realizan ninguna transformación: Por ejemplo, para el caso en donde se analiza un lema como en el par palabra-lema (joti, joti), la regla correspondiente tiene la forma de “>”.

De este modo, se procedió a obtener la cadena común más larga de cada par palabra-lema presente en el corpus, y se sustrajo los caracteres faltantes para formar la regla de derivación de forma automática, obteniendo de esta labor un conjunto de 94 clases diferentes, las cuales se muestran en el Anexo 6.

4.3.2 Selección de características

Posteriormente, se procedió a la selección de características a analizar por los clasificadores, para lo que se planteó un formato particular de alineación de los caracteres.

Este formato consiste en un vector en el cual cada uno de sus elementos es una de las letras de la palabra y el último de ellos es su categoría gramatical. Sin embargo, dado que el

shipibo-konibo es una lengua aglutinante, se procedió a invertir el orden de las letras para dar un mayor énfasis a la alineación de los posibles sufijos presentes en las palabras, los cuales a su vez se ven representados en las clases generadas anteriormente. Asimismo, en caso de que se posea una palabra que no tenga todos los elementos necesarios para el vector, se procede a rellenar los campos faltantes con ceros.

Por ejemplo, para el caso de la palabra "ainbo", su vector de características de tamaño 10 es: [o,b,n,i,a,0,0,0,0,0].

Cabe resaltar que se decidió experimentar con vectores de tamaño 5, 10 y 15 tras el análisis del tamaño de las palabras presentes en el corpus y el de sus respectivos sufijos, determinando que el tamaño del vector que brindaba un mejor resultado final era el de 10. De este modo, se logró que no se pierdan características importantes mediante el uso de un vector muy pequeño o se agregue demasiado ruido a la clasificación mediante el uso de un vector de tamaño muy grande en comparación al tamaño de las palabras presentes en el corpus.

4.3.3 Resumen

En la Tabla 6 se muestra un cuadro descriptivo del corpus obtenido tras el preprocesamiento.

Tabla 6: Cuadro descriptivo del corpus

Categorías	Número de palabras
Verbos	2,990
Sustantivos	2,443
Adjetivos	479
Adverbios	298
Nombres Propios	158
Pronombres	119
Conjunciones	98
Palabras Interrogativas	74
Posposiciones	65
Numeral	25
Determinantes	24
Interjecciones	23

Categorías	Número de palabras
Verbos Auxiliares	22
Onomatopeyas	16
Símbolos	10

4.4 Minería de datos

Tras la fase de generación de recursos, se utilizó los datos obtenidos para generar modelos de clasificación utilizando tres herramientas distintas: usando el clasificador K-NN presente en Sklearn, los algoritmos implementados en la herramienta TiMBL y la herramienta Lemmagen.

4.4.1 K-NN con Sklearn

En primer lugar, se buscó entrenar un clasificador K-NN implementado en la librería Sklearn de Python, para lo que se usaron los vectores de características y las respectivas clases generadas para cada conjunto de palabras del corpus. Asimismo, se usaron las siguientes medidas de distancia: Hamming, Euclidiana y Chebyshev; y la cantidad de vecinos utilizados (parámetro “k” del algoritmo) fueron 9, 11, 13 y 15.

4.4.2 TiMBL

Para el caso de TiMBL, se utilizaron los vectores de características generados para cada conjunto de palabras del corpus y sus respectivas clases, y entrenando 3 clasificadores basados en métodos diferentes.

En primer lugar, se utilizó un algoritmo K-NN llamado IB1, el cual utiliza una distancia de superposición para definir la similitud entre los datos utilizados para el entrenamiento, lo que genera resultados más exactos pero bajo la premisa de un mayor consumo de memoria.

En segundo lugar, se utilizó un algoritmo basado en un árbol de decisión llamado IGTREE, el cual utiliza una medida llamada Information Gain generada a partir de la

entropía de los datos de entrenamiento para generar su estructura base y añadir nuevos ejemplos a esta. Esta medida es utilizada para determinar el orden en que se van añadiendo las instancias al árbol de clasificación, de modo que se logra generalizar las clases de clasificación obteniendo de este modo una mayor velocidad en la clasificación, pero debido a que las instancias que poseen valores menores no son almacenadas se disminuye la precisión final del clasificador. Sin embargo, al momento de procesar entradas que no coinciden con ninguna de las hojas generadas en su entrenamiento, se procede a utilizar una clasificación por defecto a base del último nodo no terminal del árbol.

En tercer lugar, se utilizó un algoritmo llamado TRIBL, el cual consiste en un híbrido de los dos mencionados anteriormente con el fin de optimizar la velocidad de clasificación mediante el uso de un árbol para la separación de las clases más generales y mantener la precisión utilizando el algoritmo K-NN para los subcasos de cada clase.

Para IB1 y TRIBL, los cuales permiten variar sus medidas de distancia, se utilizaron como medidas de distancia las de Hamming, MVDM, Levenshtein y Dice, y se varió el valor de “k” entre 9, 11, 13 y 15.

4.4.3 Lemmagen

En el caso de Lemmagen, se utilizó directamente el corpus de palabras anotado para generar un árbol de clasificación basado en reglas de tipo Ripple-Down, el cual busca dar a las palabras una regla de transformación adecuada según los sufijos que ellas posean, como se muestra en la Figura 3, en donde se observa que para las palabras que terminen en “o” corresponde la regla ““”-->””, a menos que la palabra termine en “bo” o en “onko”, para las cuales existen otras reglas con sus respectivas excepciones.

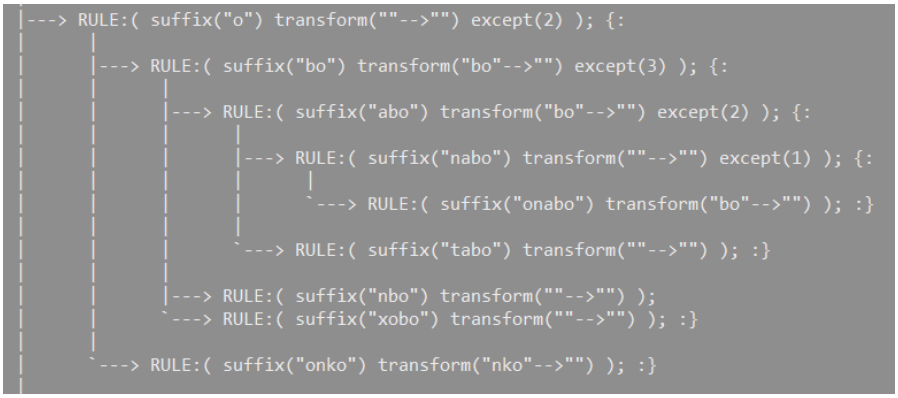

Figura 3: Ejemplo de árbol generado por Lemmagen

A partir de este árbol, se logra obtener reglas de derivación morfológica de manera automática, las cuales se encuentran ordenadas por su generalidad y se permite clasificar nuevas instancias de manera veloz y con alta precisión.

4.5 Interpretación de resultados

En esta etapa se procedió a realizar el análisis de los resultados obtenidos para cada clasificador, determinando su precisión correspondiente, así como el promedio de distancia entre los resultados obtenidos y los esperados.

4.5.1 Integración

Finalmente, se integraron los clasificadores realizados en una herramienta de lematización para el shipibo-konibo, en donde se utilizaron funciones que permitan realizar la transformación inversa de la palabra derivada a su respectivo lema utilizando la clase resultante de los clasificadores generados por Sklearn y TiMBL.

Sin embargo, debido a que la regla obtenida no siempre es aplicable a la palabra clasificada, se optó por obtener las 10 primeras reglas más cercanas a la palabra a evaluar, de modo que se pueda analizar las siguientes reglas encontradas hasta encontrar alguna que aplique, como por ejemplo para el caso de la palabra “piwe”, si se obtiene en primer lugar una regla como “ra>ti”, debido a que la palabra no termina en “ra”, esta regla no es aplicable

para esa palabra y se procede a analizar la siguiente regla según la distancia de la palabra de entrada al vector correspondiente a la próxima.

Para esta integración se utilizó un sistema de votación entre el mejor caso de cada uno de los 5 modelos generados, y en caso de empate en la votación obtenida de los clasificadores se selecciona el resultado del modelo que obtuvo una mayor precisión de forma independiente.

Asimismo, debido a que existe una lista cerrada de sufijos existentes para el shipibo-konibo, la cual se detalla en el Anexo 6, se procedió a poner una verificación de entrada, en donde se compara la terminación de la palabra de entrada con la lista de estos posibles sufijos, de modo que si esta no posee un sufijo existente, se asume que la palabra ya corresponde a un lema, y en caso contrario, se procede a aplicar el clasificador integrado para obtener su posible lema.

CAPÍTULO 5: Experimentación

En este capítulo se explica de forma detallada el proceso realizado para el entrenamiento de cada clasificador, así como la comprobación de los resultados obtenidos por cada uno de ellos y el desarrollo de las herramientas finales que forman parte de los objetivos de este proyecto.

5.1 Método de validación

Para el entrenamiento de cada clasificador se procedió a dividir los corpus en dos partes iguales de manera aleatoria, manteniendo en cada una de ellas la misma cantidad de casos por clase, y realizando esta labor 100 veces con conjuntos distintos, de modo que los resultados finales fueron obtenidos mediante el cálculo de la media aritmética de cada iteración con un margen de error igual al de dos veces la desviación estándar.

Se optó por usar este método de validación debido a que en los datos usados para el entrenamiento y evaluación de los clasificadores se contó con una gran cantidad de clases, las cuales debían ser evaluadas de manera proporcional para obtener resultados representativos a cada una de ellas, y que estas no se vean repartidas solo en los subconjuntos de entrenamiento o de evaluación.

5.2 Línea base (*Baseline)*

Con el fin de obtener una base de comparación para contrastar los resultados de cada clasificador, se procedió a generar una línea base mediante el uso de dos clasificadores simples. El primero de ellos consistió en un clasificador K-NN implementado con Sklearn en donde las reglas de transformación solo se encargaban de remover los sufijos que estaban presentes en las palabras, más no realizaban ningún paso adicional para obtener el lema

adecuado, y el segundo solo buscaba la regla más común aplicable a la palabra como resultado.

De este modo, se entrenó a los clasificadores usando una medida de distancia euclidiana, con un número de 9 vecinos, siguiendo el procedimiento planteado anteriormente y tomando la misma cantidad de palabras de entrenamiento que para el resto de los casos, lo que arrojó como resultado base los siguientes valores mostrados en la Tabla 7 y 8:

Tabla 7: Resultados de línea base solo removiendo sufijos

Línea base 1	
Precisión	Distancia entre palabras
0.327 ± 0.07	**1.637**

Tabla 8: Resultados de línea base con clase más común

Línea base 2	
Precisión	Distancia entre palabras
0. 331 ± 0.05	**1.853**

5.3 Clasificación

5.3.1 Clasificador con K-NN con Sklearn

Para este caso, fue necesario transformar el vector de características a un formato numérico, para lo que se representaron las letras de la palabra con su valor correspondiente en código ASCII. Luego de esta fase de generación de características y clases para el corpus de palabras se procedió con el entrenamiento del algoritmo.

Como cantidad de vecinos, se probaron los valores 9, 11, 13 y 15, utilizando pesos uniformes para cada característica y como medida de distancia la Euclidiana, la de Hamming y la de Chebyshev, lo cual arrojo los resultados mostrados en la Tabla 9:

Tabla 9: Resultados de Sklearn

K	Distancia	Precisión	Distancia entre palabras
9	Euclidiana	**0.702 ± 0.03**	**1.562**
	Hamming	**0.713 ± 0.04**	**1.479**
	Chebyshev	**0.698 ± 0.03**	**1.544**
11	Euclidiana	**0.707 ± 0.03**	**1.459**
	Hamming	**0.717 ± 0.05**	**1.469**
	Chebyshev	**0.704 ± 0.04**	**1.572**
13	Euclidiana	**0.713 ± 0.04**	**1.359**
	Hamming	**0.718 ± 0.03**	**1.428**
	Chebyshev	**0.709 ± 0.06**	**1.399**
15	Euclidiana	**0.711 ± 0.03**	**1.454**
	Hamming	**0.715 ± 0.03**	**1.448**
	Chebyshev	**0.705 ± 0.05**	**1.495**

5.3.2 Clasificador con TiMBL

Como se mencionó anteriormente, para esta herramienta se entrenaron 3 algoritmos distintos: IB1, IGTREE y TRIBL.

Para el caso de IB1, la cantidad de vecinos utilizada fue la misma que para el clasificador con Sklearn, utilizando pesos uniformes para cada característica y como medida de distancia la de Hamming, MVDM, Levenshtein y Dice en los que aplicaban, lo cual arrojo los resultados mostrados en la Tabla 10.

Tabla 10: Resultados de IB1

K	Distancia	Precisión	Distancia entre palabras
9	Hamming	**0.639 ± 0.04**	**1.841**
	MVDM	**0.677 ± 0.05**	**1.635**
	Levenshtein	**0.679 ± 0.03**	**1.701**
	Dice	**0.637 ± 0.03**	**1.614**
11	Hamming	**0.641 ± 0.04**	**1.796**
	MVDM	**0.678 ± 0.07**	**1.673**
	Levenshtein	**0.682 ± 0.06**	**1.642**
	Dice	**0.642 ± 0.03**	**1.679**
13	Hamming	**0.645 ± 0.05**	**1.662**
	MVDM	**0.689 ± 0.06**	**1.748**
	Levenshtein	**0.699 ± 0.04**	**1.559**
	Dice	**0.645 ± 0.03**	**1.697**
15	Hamming	**0.642 ± 0.03**	**1.745**
	MVDM	**0.684 ± 0.04**	**1.762**

K	Distancia	Precisión	Distancia entre palabras
	Levenshtein	**0.695 ± 0.05**	**1.856**
	Dice	**0.638 ± 0.03**	**1.682**

Del mismo modo, para el caso de IGTREE, el cual no utiliza parámetros de número de vecinos ni de distancias, se obtuvieron los resultados mostrados en la Tabla 11.

Tabla 21: Resultados de IGTREE

Precisión	Distancia entre palabras
0.687 ± 0.04	**1.493**

Finalmente, para el caso de TRIBL, se usaron los mismos parámetros y tras realizar experimentos con diversos valores, se utilizó un valor igual a la mitad de los casos a analizar (1,711) para determinar el momento en que se pasa de IGTREE a IB1, lo cual generó los resultados mostrados en la Tabla 12.

Tabla 12: Resultados de TRIBL

K	Distancia	Precisión	Distancia entre palabras
9	Hamming	**0.625 ± 0.04**	**1.841**
	MVDM	**0.667 ± 0.05**	**1.675**
	Levenshtein	**0.645 ± 0.03**	**1.742**
	Dice	**0.612 ± 0.03**	**1.614**
11	Hamming	**0.633 ± 0.04**	**1.796**
	MVDM	**0.668 ± 0.07**	**1.683**
	Levenshtein	**0.658 ± 0.06**	**1.726**
	Dice	**0.622 ± 0.03**	**1.679**
13	Hamming	**0.638 ± 0.05**	**1.662**
	MVDM	**0.672 ± 0.06**	**1.656**
	Levenshtein	**0.679 ± 0.04**	**1.698**
	Dice	**0.631 ± 0.03**	**1.697**
15	Hamming	**0.632 ± 0.03**	**1.745**
	MVDM	**0.671 ± 0.04**	**1.692**
	Levenshtein	**0.665 ± 0.05**	**1.811**
	Dice	**0.628 ± 0.03**	**1.682**

5.3.3 Clasificador con Lemmagen

Tras el entrenamiento de este clasificador se obtuvo un árbol de clasificación con un promedio de 1,741 nodos, de los cuales 1,150 son hojas y 591 son nodos internos, lo cual permite representar las diferentes reglas de derivación gramatical.

De este modo, para este clasificador se obtuvieron los resultados mostrados en la Tabla 13.

Tabla 13: Resultados de Lemmagen

Lemmagen	
Precisión	Distancia entre palabras
0.613 ± 0.04	**1.537**

Asimismo, se puede observar una parte de uno de los árboles generados por la herramienta en el Anexo 5.

5.3.4 Resumen

Se pudo observar que los resultados obtenidos de manera independiente por cada uno de estos clasificadores superaron de manera clara a los obtenidos por la línea base e incluso que los modelos desarrollados mediante el uso de clasificadores K-NN superan a la herramienta Lemmagen debido a la flexibilidad que dan estos para poder obtener más de un solo resultado.

Asimismo, en la Figura 4 se muestra cómo varió la precisión de cada uno de los mejores clasificadores generados, Sklearn con Hamming, IB1 con Levenshtein y MVDM, y TRIBL con Levenshtein y MVDM, según el número de vecinos utilizados, mostrando que el valor que da mejores resultados para todos los clasificadores es el de 13, y que el modelo que brinda la mejor precisión es el de Sklearn.

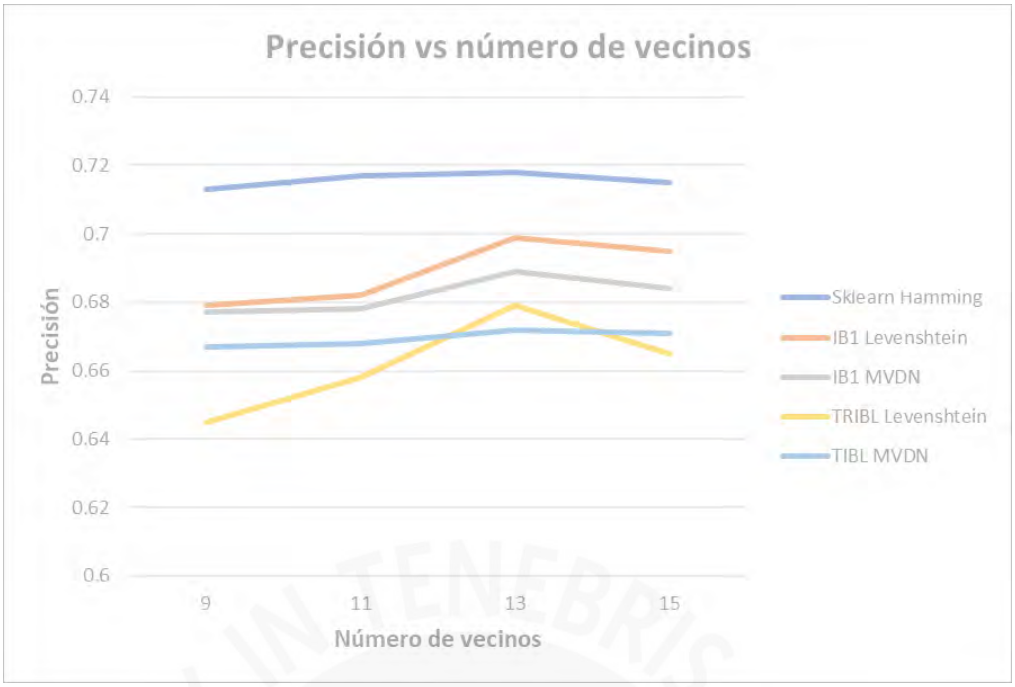

Figura 4: Precisión vs número de vecinos

También, en la Figura 5 se puede observar que la variación de la distancia promedio obtenida entre los resultados obtenidos por los clasificadores y los resultados reales es menor en 4 de los 5 mejores clasificadores cuando se usa un número de 13 vecinos para su evaluación, lo que indica que este de número de vecinos es el indicado para obtener una menor distancia entre palabras.

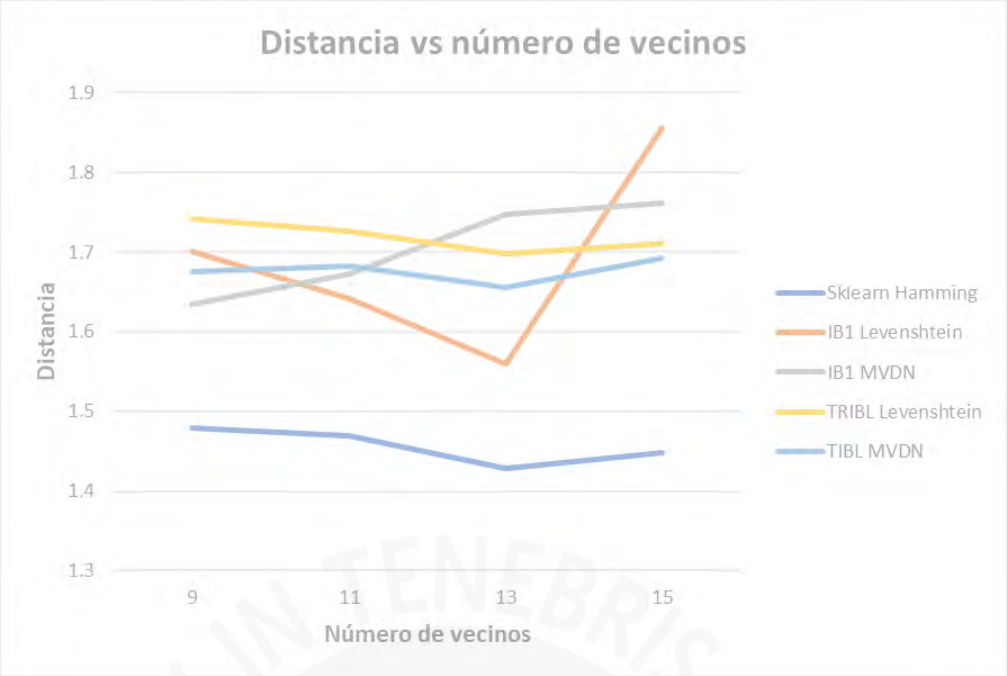

Figura 5: Distancia vs número de vecinos

Finalmente, debido a que con el valor de 13 vecinos se obtiene una convergencia para ambos resultados, este es el que será utilizado en la integración final de los clasificadores, con el fin de lograr el mejor desempeño en ellos.

5.4 Integración

Tras el proceso de integración de los algoritmos, en donde se utilizó el modelo entrenado que dio los mejores resultados para cada una de las herramientas, y aplicando el procedimiento explicado anteriormente, en donde se procede a analizar si la palabra de entrada posee alguno de los posibles sufijos presentes para el shipibo-konibo, y luego pasar por el sistema de votación para obtener el lema resultante, se obtuvieron los resultados mostrados en la Tabla 14.

Tabla 14: Resultados finales del lematizador

Resultados de la integración	
Precisión	Distancia entre palabras
0. 736 ± 0.04	**1.249**

Los cuales, en comparación a la línea base planteada, superan en gran medida los valores obtenidos en ella, por lo que se puede apreciar que sí es posible generar un lematizador con resultados adecuados para el shipibo-konibo, e incluso uno que supere a algún lematizador previamente desarrollado para otros idiomas, como es el caso de Lemmagen.

5.5 Herramientas finales

Finalmente, tras los procesos de validación e integración de los algoritmos, se procedió a generar un servicio web y una librería que permitan utilizar los modelos entrenados de una manera más sencilla y de fácil acceso para el público en general.

5.5.1 Servicio web

Con el fin de poder ir observando los resultados obtenidos en este proyecto, así como facilitar la anotación de nuevas palabras para ampliar el corpus de entrenamiento mediante el uso de sugerencias en la herramienta de anotación ChAnot, se crearon diferentes servicios web que permiten utilizar los diversos modelos entrenados, así como la herramienta final para obtener el lema de palabras en shipibo-konibo.

Estos servicios web fueron realizados en Python con el framework de desarrollo Flask, de modo en que cualquier usuario interesado en obtener el lema de una palabra en shipibo-konibo solo debe de acceder a una ruta web y colocar la palabra a analizar, permitiendo que de esta forma se obtenga una manera sencilla de utilizar la herramienta generada.

Los servicios web generados se encuentran alojados dentro de un servidor interno de la universidad[4], el cual puede ser accedido de manera libre por cualquier persona que se encuentre en la red interna.

5.5.2 Librería de software

Adicionalmente, con el fin de permitir que la herramienta sea de acceso público y pueda ser utilizada en diversas laboras mediante una fácil instalación, se generó una librería de software en Python llamada “chana-library”[5]. Esta contiene diversas herramientas de procesamiento de lenguaje natural para el idioma shipibo-konibo, entre las cuales se encuentra un módulo especializado en lematización con el modelo generado en este proyecto.

En este módulo se encuentran las funciones necesarias para obtener el lema de una palabra en shipibo-konibo mediante el uso del modelo de aprendizaje generado, así como diversas funciones adicionales que permiten realizar esta labor, como son la obtención de la regla necesaria para obtener el lema correspondiente a una palabra o la verificación de si una palabra cuenta con uno de los sufijos existentes dentro del idioma.

Asimismo, en este módulo se encuentran las herramientas necesarias para crear un nuevo lematizador para este u otros idiomas similares mediante el uso de una lista de palabras con sus respectivos lemas. El proceso aplicado en la creación de estos nuevos modelos sigue la misma lógica que se utilizó para el lematizador del shipibo-konibo.

4 Las rutas de acceso se encuentran disponibles en: http://chana.inf.pucp.edu.pe/index.php/api/
5 Esta librería se encuentra disponible en el siguiente enlace: https://github.com/iapucp/chana-library y su documentación detallada en: https://chana.readthedocs.io/en/latest/index.html

Capítulo 6: Conclusiones y Trabajos Futuros

En este capítulo se presenta de forma detallada las conclusiones obtenidas en el desarrollo de este proyecto de fin de carrera, así como los posibles trabajos futuros y limitaciones encontradas durante el desarrollo.

6.1 Conclusiones

El objetivo principal de este proyecto de fin de carrera fue desarrollar una herramienta de lematización para el shipibo-konibo usando un corpus de palabras anotadas y reglas generadas por expertos en la lengua. Sin embargo, para llevar este objetivo a cabo se requería plantear un método que permitiera optimizar la cantidad pequeña de recursos que se poseía para la lengua y la falta del gran conocimiento de expertos que permitiera generar las reglas a ser utilizadas.

Debido a esto, la herramienta final fue generada mediante el uso de dos métodos, el entrenamiento y uso de Lemmagen, un lematizador ya desarrollado para otras lenguas, y con de diversos clasificadores K-NN entrenados para predecir una regla de transformación, las cuales fueron generadas de manera automática, que permita obtener el lema de una palabra en shipibo-konibo tras la aplicación de esta. Ambos métodos fueron posteriormente integrados en un solo proceso, en donde mediante un sistema de votación y la verificación de la existencia de sufijos en la palabra a analizar, se procedía a obtener el lema correspondiente.

Con este fin, en primer lugar, se tuvieron que definir los formatos de anotación de las categorías gramaticales para formar el corpus de palabras anotadas, para lo cual se utilizó una herramienta de anotación ChAnot. Esta permitió que se obtenga la información necesaria de las palabras y lemas junto a sus respectivas categorías para el shipibo-konibo, las cuales ya estaban alineadas a las Universal Dependencies dentro de la misma herramienta. Esta

información de las anotaciones se encontró almacenada en los archivos XML resultantes de la herramienta, los cuales fueron procesados para obtener el formato necesario en el desarrollo de este proyecto, presente en los Anexos 3 y 4, que fueron utilizados para el posterior entrenamiento y evaluación de Lemmagen y de los clasificadores K-NN, logrando de esta manera satisfacer el primero de los objetivos específicos planteados.

En segundo lugar, debido que para el segundo objetivo específico se planteó generar un patrón de comportamiento que permita formar reglas morfológicas de manera automática, mediante el uso del corpus anotado se procedió a obtener las reglas de transformación necesarias para transformar cada palabra en su respectivo lema. Estas reglas fueron posteriormente utilizadas en conjunto con la información presente de cada palabra en el corpus para entrenar y evaluar los diversos clasificadores. Estos fueron elaborados mediante el uso del algoritmo K-NN presente en la librería Sklearn y los diferentes algoritmos presentes en TiMBL, los cuales permitieron obtener estos patrones de comportamiento que permiten obtener la regla morfológica correspondiente a cada palabra para transformarla en su respectivo lema.

Del mismo modo, mediante el uso del corpus obtenido se procedió a realizar las transformaciones necesarias a este para adecuarse al formato utilizado por la herramienta Lemmagen, de modo que esta pueda ser entrenada para su posterior uso y evaluación, logrando utilizar la herramienta de manera exitosa

Posteriormente, debido a que, en la realización de las pruebas de evaluación de desempeño de las herramientas de manera independiente se obtuvieron resultados cuyos valores superaron a los planteados en la línea base para cada una de ellas, se concluye que estos patrones de comportamiento, así como el uso de la herramienta Lemmagen, lograron alcanzar los resultados esperados para cada uno de ellos. Igualmente, tras la integración de las diversas herramientas mediante un sistema de votación, se obtuvo un resultado total de 0.736

de precisión y de 1.249 de distancia entre los lemas resultantes y los lemas correctos, valores que superaron al resto de resultados obtenidos por cada herramienta de forma independiente, por lo que se puede determinar que este objetivo también fue satisfecho.

En tercer lugar, dado que el tercer objetivo fue implementar un servicio web y una librería de funciones que permitan utilizar las herramientas generadas en el proyecto para otras aplicaciones en shipibo-konibo, se desarrolló la librería "chana-library", así como se implementaron diversos servicios web dentro de un servidor de la universidad. En ambos casos, se utilizaron los modelos generados y se procedió a crear las funciones necesarias para realizar el proceso de lematización, las cuales fueron utilizadas dentro de la herramienta de anotación ChAnot, así como para otras tareas, lo que permitió cumplir el tercer de los objetivos específicos planteados.

Por lo previamente mencionado sobre cada uno de los resultados obtenidos dentro de cada uno de los objetivos de este proyecto de fin de carrera, se puede llegar a las siguientes conclusiones:

Primeramente, al haber alcanzado satisfactoriamente cada uno de los objetivos específicos del proyecto, se puede concluir que se consiguió el completar el objetivo general del proyecto: Desarrollar una herramienta de lematización para el shipibo-konibo usando un corpus de palabras anotadas y reglas generadas por expertos en la lengua.

Del mismo modo, se muestra que existen herramientas existentes que pueden ser reutilizadas de manera adecuada, como es el caso de Lemmagen, obteniendo resultados satisfactorios en la generación de lematizadores para lenguas de escasos recursos como el shipibo-konibo, debido a que esta posee una naturaleza altamente aglutinante, la cual también se encuentra presente en otras lenguas de gran uso y con herramientas ya desarrolladas para ellas.

Después, se observó que el método que obtuvo mejores resultados fue el del clasificador realizado con la herramienta Sklearn y la distancia de Hamming. Esto se debe a que en comparación a el resultado único que da una herramienta empaquetada como es Lemmagen, mediante el método utilizado en este proyecto se puede obtener más de un posible resultado en forma de regla de transformación con ella, lo que trae como consecuencia una mayor probabilidad de obtener un resultado adecuado en caso de que exista algún error en el primer resultado obtenido, frente a una herramienta con un resultado único.

De similar manera, debido a que aún se cuenta con un corpus de entrenamiento relativamente pequeño y que en cada idioma existen diversas palabras que poseen terminaciones que a pesar de tener igual forma que algún sufijo son excepciones a la regla, mediante el uso del método empleado con los clasificadores se puede obtener un resultado más adecuado ya que con este método se compara toda la estructura de la palabra y no solo la terminación de ella como hace Lemmagen. Por lo que se concluye que el uso de métodos que brinden mayor flexibilidad en sus resultados y en su forma de obtenerlo, permitirá obtener mejores resultados en la herramienta final.

Del mismo modo, para estos clasificadores que implementan K-NN se observó que las medidas de distancia entre palabras o caracteres influyen dando mejores resultados frente a las medidas utilizadas para determinar distancias entre valores numéricos. Esto se debe a la naturaleza del problema, en donde los datos a analizar son en un inicio caracteres, por lo que al momento de transformarlos a un formato numérico se pierden las características propias de cada uno de ellos y la similitud y cercanía que existe en un dominio cerrado como es un alfabeto frente a uno abierto como el de los números el cual variará de acuerdo al valor relativo que reciba cada carácter al momento de ser transformado por la propia implementación de cada herramienta.

Finalmente, para los clasificadores basados en K-NN, se observa que los resultados obtenidos variaron de acuerdo con el número de vecinos que fueron utilizados para su evaluación, por lo que se concluye que debe existir de esta forma un equilibrio entre la cantidad de datos de entrenamiento y la cantidad de vecinos a utilizar para obtener el mejor resultado posible, así como la cantidad de muestras que presente cada regla de transformación dentro del corpus para evitar casos en donde se presenten errores por la similitud que pueda existir entre ellas.

6.2 Trabajos futuros

Durante el desarrollo de este proyecto de fin de carrera se han podido observar ciertos factores que pueden contribuir a la realización de futuras mejoras en el desarrollo para la obtención de mejores resultados, entre los cuales se encuentran:

- En primer lugar, se observó durante el desarrollo paulatino de los experimentos de este proyecto que el tamaño del corpus utilizado influye directamente a los resultados obtenidos (se obtuvo una precisión de 0.63 en un corpus de 460 palabras), por lo que con la recopilación de nuevas palabras anotadas se pueden conseguir mejores resultados.
- En segundo lugar, debido a que los resultados obtenidos en la integración final muestran la experimentación con diversos parámetros para los clasificadores (número de vecinos a usar, tamaño del vector de características a evaluar y medidas de distancia entre palabras), los cuales fueron optimizados para este corpus en especial, en caso se decida utilizar mejorar el tamaño del corpus se pueden realizar nuevos experimentos con nuevos parámetros que se adecuen al nuevo caso.
- En tercer lugar, debido a que el corpus a utilizar fue previamente procesado para eliminar las anotaciones duplicadas y los algunos errores presentes en la

anotación, se podría determinar cuáles son las clases que presenten un conjunto bajo de muestras, con el fin de realizar un diccionario que procese automáticamente estos casos y de esta manera evitar errores en estos casos, así como para el caso de palabras que sean excepciones a las reglas de transformación encontradas.

- En cuarto lugar, dado que aún no se cuenta con un alfabeto estandarizado para la escritura del shipibo-konibo, es posible que dentro del corpus se encuentren casos de sufijos que a pesar de que sean los mismo hayan sido escritos de distintas formas, por lo que se podría realizar un preprocesamiento de estos casos.
- En quinto lugar, dado que se usaron diversas implementaciones de un mismo algoritmo para la realización de los clasificadores, y que el resultado del mejor de estos no difiere en gran medida del resultado de la integración final, es necesario evaluar si es que con un corpus de mayor tamaño es conveniente el uso de solo uno de los clasificadores en lugar de realizar una integración entre ellos.

De igual manera, existe la posibilidad de elaborar de diversos trabajos relacionados para mejorar el desempeño de la herramienta o utilizarla para diversas aplicaciones, entre los cuales se presentan:

- En primer lugar, debido a que siempre están presentes los extranjerismos dentro de los lenguajes, como trabajo futuro es posible integrar este proyecto a un identificador de lenguaje, de modo que solo sean evaluadas las palabras que corresponden al shipibo-konibo y no las de otras lenguas, las cuales empeoran el desempeño final de la herramienta.
- En segundo lugar, como se mencionan en los experimentos realizados por [Pereira et al., 2017], es posible agregar la etiqueta gramatical de cada palabra como una nueva categoría, de modo que mediante el uso en conjunto con un etiquetador

gramatical para el shipibo-konibo, es posible desarrollar una herramienta que brinde mejores resultados.

- En tercer lugar, dada la naturaleza aglutinante que presenta el shipibo-konibo, es posible reentrenar los clasificadores desarrollados con corpus de otros lenguajes de escasos recursos que posean esta misma característica y de este modo obtener lematizadores para ellos de manera sencilla y así contribuir con la conservación de estos idiomas y con su inclusión en el ámbito computacional, tarea que ya ha sido avanzada dentro de la librería desarrollada.

Referencias:

[Aikhenvald, 2007]
Aikhenvald, A. Y. (2007). Typological distinctions in word-formation. *Language typology and syntactic description*, *3*, 1-65.

[Alcina; Valero, 2009]
Alcina, A., & Valero, E. (2009). *Terminología y sociedad del conocimiento*. Peter Lang.19-25.

[Bardales Rodríguez, 1979]
Bardales Rodríguez, C. (1979). Quimisha Incabo Ini Yoia Leyendas de los Shipibo-Conibo sobre los Tres Incas. Comunidades y Culturas Peruanas, 12. Yarinacocha: Instituto Lingüístico de Verano. 2da edición

[Bismark, 2006]
Bismark, P. V. (2006). Aspectos morfosintácticos de la relativización en shipibo-konibo (pano). *Boletim do Museu Paraense Emílio Goeldi Ciências Humanas*, *1*(1), 123-134.

[Cambria; White, 2014]
Cambria, E., & White, B. (2014). Jumping NLP curves: a review of natural language processing research [review article]. *Computational Intelligence Magazine, IEEE*, *9*(2), 48-57.

[CONGRESO DE LA REPÚBLICA, 2011]
CONGRESO DE LA REPÚBLICA 2011 LEY Nº 29735 Ley que regula el uso, preservación, desarrollo, recuperación, fomento y difusión de las lenguas originarias del Perú. Lima, 5 de julio.

[Crystal, 2011]
Crystal, D. (2011). *Dictionary of linguistics and phonetics* (Vol. 30). John Wiley & Sons.

[Daelemans et al., 2009]
Daelemans, W., Groenewald, H. J., & Van Huyssteen, G. B. (2009). Prototype-based active learning for lemmatization.

[Daelemans et al., 2007]
Daelemans, W., Zavrel, J., Van der Sloot, K., & Van den Bosch, A. (2007). Timbl: Tilburg memory-based learner. *Version*, *6*, 07-03.

[Delmonte, 2013]
Delmonte, R. (2013). Italian Lemmatization by Rules with Getaruns. En *Evaluation of Natural Language and Speech Tools for Italian* (pp. 239-248). Springer Berlin Heidelberg.

[Deza; Deza, 2009]
Deza, M. M., & Deza, E. (2009). Encyclopedia of distances. En *Encyclopedia of Distances* (pp. 1-583). Springer Berlin Heidelberg.

[Docs.python.org, 2016]
Docs.python.org. (2016). General Python FAQ — Python 3.5.2 documentation. Recuperado de https://docs.python.org/3/faq/general.html#what-is-python [Accedido 26 Mayo 2016].

[Faust, 1973]
Faust, N. (1973). Lecciones para el aprendizaje del idioma shipibo-conibo. *Documento de trabajo*, *1*.

[Fayyad et al., 1996]
Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM, 39(11), 27-34.

[Forcada, 2006]
Forcada, M. L. (2006). Open source machine translation: an opportunity for minor languages. En *Proceedings of the Workshop "Strategies for developing machine translation for minority languages", LREC* (Vol. 6, pp. 1-6).

[Gallmann, 1991]
Gallmann, P. (1991). Wort, Lexem und Lemma. *Rechtschreibwörterbücher in der Diskussion: Geschichte–Analyse–Perspektiven*, 261-80.

[Gallay; Šimko, 2016]
Gallay, L., & Šimko, M. (2016). Utilizing Vector Models for Automatic Text Lemmatization. En *SOFSEM 2016: Theory and Practice of Computer Science* (pp. 532-543). Springer Berlin Heidelberg.

[Gasser, 2011]
Gasser, M. (2011). ANTIMORFO 1.2 User's Guide.

[Goldwater; McClosky, 2005]
Goldwater, S., & McClosky, D. (2005, October). Improving statistical MT through morphological analysis. En *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 676-683). Association for Computational Linguistics.

[Guerra, 2003]
Guerra, A. M. M. (2003). *La microestructura del diccionario: la definición* (pp. 127-150). Ariel.

[Gupta et al., 2016]
Gupta, V., Joshi, N., & Mathur, I. (2016). Design and Development of a Rule-Based Urdu Lemmatizer. In *Proceedings of International Conference on ICT for Sustainable Development* (pp. 161-169). Springer Singapore.

[Halácsy; Trón, 2006]
Halácsy, P., & Trón, V. (2006, September). Benefits of deep NLP-based Lemmatization for Information Retrieval. En *CLEF* (Working Notes).

[Homola, 2011]
Homola, P. (2011). Parsing a Polysynthetic Language. En *RANLP* (pp. 562-567).
[Instituto Lingüístico de Verano, 1979]
Instituto Lingüístico de Verano, Perú (1979). Quirica 9.

[Jozef Stefan Institute, 2010]
Jozef Stefan Institute. (2016). LemmaGen. Recuperado de http://lemmatise.ijs.si/ [Accedido 5 Nov. 2016].

[Juršic et al., 2007]
Juršič, M., Mozetič, I., & Lavrač, N. (2007). Learning ripple down rules for efficient lemmatization. En *Proceedings of the 10th international multiconference information society, IS* (pp. 206-209).

[Juršic et al., 2010]
Juršic, M., Mozetic, I., Erjavec, T., & Lavrac, N. (2010). Lemmagen: Multilingual lemmatisation with induced ripple-down rules. *Journal of Universal Computer Science*, *16*(9), 1190-1214.

[Karashtranova et al., 2015]
Karashtranova, E., Iliev, G., Borisova, N., Chankova, Y., & Atanasova, I. (2015). Evaluation of the Accuracy of the BGLemmatizer. *arXiv preprint arXiv:1506.04229*.

[Khaltar; Fujii, 2009]
Khaltar, B. O., & Fujii, A. (2009). A lemmatization method for Mongolian and its application to indexing for information retrieval. *Information Processing & Management*, *45*(4), 438-451.

[Kohavi; Provost, 1998]
Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning*, *30*(2-3), 271-274.

[Koskenniemi, 1983]
Koskenniemi, K.: Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. Ph.D. Thesis, University of Helsinki, Department of General Linguistics, Helsinki (1983)

[Laaksonen; Oja, 1996]
Laaksonen, J., & Oja, E. (1996, June). Classification with learning k-nearest neighbors. En *Neural Networks, 1996., IEEE International Conference on* (Vol. 3, pp. 1480-1483). IEEE.

[Lauriout et al., 2003]
Lauriout, E., Day, D., & Loriot, J. (1993). Diccionario shipibo-castellano.

[Levenshtein, 1966]
Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady. 10 (8): 707–710.

[Liu, 2011]
Liu, B. (2011). Association rules and sequential patterns. In *Web Data Mining* (pp. 17-62). Springer Berlin Heidelberg.

[Loper; Bird, 2002]
Loper, E., & Bird, S. (2002). NLTK: The natural language toolkit. En *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*(pp. 63-70). Association for Computational Linguistics.

[Loponen et al., 2013]
Loponen, A., Paik, J. H., & Järvelin, K. (2013). UTA Stemming and Lemmatization Experiments in the FIRE Bengali Ad Hoc Task. En *Multilingual Information Access in South Asian Languages* (pp. 258-268). Springer Berlin Heidelberg.

[Loponen; Järvelin, 2010]
Loponen, A., & Järvelin, K. (2010). A dictionary-and corpus-independent statistical lemmatizer for information retrieval in low resource languages. En *Multilingual and Multimodal Information Access Evaluation* (pp. 3-14). Springer Berlin Heidelberg.
[MacKay; David, 2003]
MacKay, David J. C.. Information Theory, Inference, and Learning Algorithms Cambridge: Cambridge University Press, 2003. ISBN 0-521-64298-1

[Mercado et al, 2018]
ChAnot: An intelligent annotation tool for indigenous and highly agglutinative languages in Peru. Mercado, R., Pereira, J., Sobrevilla-Cabezudo, M. & Oncevay-Marcos, A. (2018). En Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). (In-press)

[Mikolov et al., 2013]
Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. En *Advances in neural information processing systems* (pp. 3111-3119).

[MINEDU, 2015]
MINISTERIO DE EDUCACIÓN (MINEDU) 2015 RESOLUCIÓN MINISTERIAL Nº 303-2015-MINEDU. Lima, 12 de junio

[Ministerio de Educación, 2016]
Ministerio de Educación, Perú. Dirección General de Educación Intercultural, Bilingüe y Rural. http://www.minedu.gob.pe/digeibir/. Accedido en: 6 de abril de 2016.

[Ministerio de Educación, 1982]
Ministerio de Educación con la colaboración del Instituto Lingüístico de Verano, Lima, Perú (1982). Quirica 8.

[Ministerio de Educación Pública, 1960]
Ministerio de Educación Pública, Lima, Perú (1960). Naturaleza y vida social 3.

[Ministerio de Educación Pública, 1963]
Ministerio de Educación Pública, Lima, Perú (1963). Naturaleza y vida social 2.

[Mohri et al, 2012]
Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

[Pedregosa et al., 2011]
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12, 2825-2830.

[Pereira et al., 2017]
Pereira-Noriega, J., Mercado-Gonzales, R., Melgar, A., Sobrevilla-Cabezudo, M., & Oncevay-Marcos, A. (2017, August). Ship-LemmaTagger: Building an NLP Toolkit for a Peruvian Native Language. En *International Conference on Text, Speech, and Dialogue* (pp. 473-481). Springer, Cham.

[Pirkola et al., 2003]
Pirkola, A., Toivonen, J., Keskustalo, H., Visala, K., & Järvelin, K. (2003). Fuzzy translation of cross-lingual spelling variants. En *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 345-352). ACM.

[Piskorski et al. 2007]
Piskorski, J., Sydow, M., & Wieloch, K. (2007). Comparison of string distance metrics for lemmatisation of named entities in polish. En *Human Language Technology. Challenges of the Information Society* (pp. 413-427). Springer Berlin Heidelberg.

[Python.org, 2016]
Python.org. (2016). Welcome to Python.org. Recuperado de https://www.python.org/about [Accedido 26 Mayo 2016].

[Pyysalo et al., 2015]
Pyysalo, S., Kanerva, J., Missilä, A., Laippala, V., & Ginter, F. (2015). Universal dependencies for Finnish. En *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania* (No. 109, pp. 163-172). Linköping University Electronic Press.

[Ristad; Yianilos, 1998]
Ristad, E. S., & Yianilos, P. N. (1998). Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *20*(5), 522-532.

[Ronacher, 2010]
Ronacher, A. (2010). Welcome—flask (a python microframework). Recuperado de http://flask.pocoo.org [Accedido 12 de Mayo 2017]

[Scikit-learn.org, 2016]
Scikit-learn.org. (2016). *1.4. Support Vector Machines — scikit-learn 0.17.1 documentation*. Recuperado de http://scikit-learn.org/stable/modules/svm.html [Accedido 28 mayo 2016].

[Scheffer, 1996]
Scheffer, T. (1996). Algebraic foundation and improved methods of induction of ripple down rules. En *In*.

[Scheffer, 1996 (Fig)]
Scheffer, T. (1996). Algebraic foundation and improved methods of induction of ripple down rules. [Figura] En *In*.

[Singh, 2008]
Singh, A. K. (2008). Natural Language Processing for Less Privileged Languages: Where do we come from? Where are we going? En *IJCNLP* (pp. 7-12).

[Streiter et al., 2006]
Streiter, O., Scannell, K. P., & Stuflesser, M. (2006). Implementing NLP projects for noncentral languages: instructions for funding bodies, strategies for developers. *Machine Translation*, *20*(4), 267-289.

[Sullón et al., 2013]
Sullón Acosta, K. N., Huamancayo Curi, E., Mori Clement, M., & Carbajal Solis, V. (2013). Documento nacional de lenguas originarias del Perú.

[Ting et al., 2014]
Ting, M., Kadir, R. A., Sembok, T. M. T., Ahmad, F., & Azman, A. (2014). Adaptive learning for lemmatization in morphology analysis. En *Intelligent Software Methodologies, Tools and Techniques* (pp. 343-357). Springer International Publishing.

[Tkachenko et al., 2014]
Tkachenko, A., Petmanson, T., & Laur, S. (2014). Hybrid Lemmatizer for Estonian. En *Baltic HLT* (pp. 244-247).

[Tognini-Bonelli, 2001]
Tognini-Bonelli, E. (2001). *Corpus linguistics at work* (Vol. 6). John Benjamins Publishing. 52-55.

[Tufiş; Ceauşu, 2009]
Tufiş, D., & Ceauşu, A. (2009, June). Factored phrase-based statistical machine translation. En *Speech Technology and Human-Computer Dialogue, 2009. SpeD'09. Proceedings of the 5-th Conference on* (pp. 1-7). IEEE.

[Valenzuela, 2003]
Valenzuela, P. M. (2003). Transitivity in shipibo-konibo grammar (Doctoral dissertation, University of Oregon Eugene

Anexo 1: Ship-LemmaTagger: Building an NLP Toolkit for a Peruvian Native Language

Ship-LemmaTagger: Building an NLP Toolkit for a Peruvian Native Language

José Pereira-Noriega[1], Rodolfo Mercado-Gonzales[2], Andrés Melgar[2], Marco Sobrevilla-Cabezudo[2], and Arturo Oncevay-Marcos[2]

[1] Facultad de Ciencias e Ingeniería, Pontificia Universidad Católica del Perú, Lima, Peru
jpereira@pucp.pe

[2] Departamento de Ingenier´ıa, Research Group on Pattern Recognition and Applied Artificial Intelligence, Pontificia Universidad Católica del Perú, Lima, Peru
{rmercado,amelgar,msobrevilla,arturo.oncevay}@pucp.edu.pe

Abstract. Natural Language Processing deals with the understanding and generation of texts through computer programs. There are many different functionalities used in this area, but among them there are some functions that are the support of the remaining ones. These methods are related to the core process- ing of the morphology of the language (such as lemmatization) and automatic identification of the part-of-speech tag. Thereby, this paper describes the implementation of a basic NLP toolkit for a new language, focusing in the features mentioned before, and testing them in an own corpus built for the occasion. The obtained results exceeded the expected results and could be used for more complex tasks such as machine translation.

Keywords: Part-of-speech tagging, Lemmatization, Low resource language, Shipibo-konibo

1 Introduction

Traditionally, both Part-of-speech tagging (POS-tagging) and Lemmatization were made by the use of hand-crafted rules [6]. However, there are several recent studies showing that machine learning approaches are suitable to solve these tasks without taking any effort in defining all the rules and exceptions needed for a particular language.

Specifically, in the case of an agglutinative language like Shipibo-konibo, the labor of building rules is not feasible due to all of the possible combinations of affixes. Also, due to the lack of an established order in the words of a sentence in this language, the labor of developing rules for POS-tagging is particularly time-consuming.

Nevertheless, in order to use machine learning approaches, it is necessary to have an annotated corpus. In this way, since it is easier to build those datasets than the rules, it was decided to follow this learning approach for the develop of our NLP tools for this low resource language.

The paper is organized as follows: in Section 2 are presented some works related to lemmatization and POS-tagging for agglutinative and low-resourced languages. After that, Section 3 describes the case study of this research: the Shipibo-konibo language.

Later, the corpus annotation process is presented in Section 4. Then, Section 5 ex- plains the functionalities developed in this work. Finally, the experiments performed are included in Section 6, and Section 7 presents some conclusions and potential future works.

2 Related Works

In the case of the Shipibo-konibo language, there have not been any direct attempts to solve the problem of POS-tagging or lemmatization. Moreover, this language does not even have an annotated corpus or any computational tool. However, there are some studies for similar agglutinative and low-resourced languages that show some progress in solving these tasks.

For the POS-Tagging task, in languages like Bhojpuri [11] and Bengali [3], the supervised learning approach had a great performance (between 86% and 90% of accuracy). The experiments made for these languages were performed with Support Vector Machines trained models. Also in similar languages like Nepali, approaches based in Hidden Markov Models were used with a little lower results [10].

Regarding the lemmatization task, in languages like Urdu [4] or Mongol [7], it is shown that a rule-based approach can be really effective in solving this problem. However, these studies used manually generated rules, a big corpus, and dictionaries of words to deal correctly with exceptions.

Although, due to the particular agglutinative characteristics of the Shipibo-konibo language, the labor of making manual derivation rules is not feasible. Therefore, it is also possible to develop rules automatically, like it is shown for the Afrikaans [2] and for some European languages [6]. However, since the corpus built for this study is currently smaller than the ones used for those languages, lower results were expected for this work.

3 The Shipibo-konibo Language

Shipibo-konibo (SHP) is the sixth language with highest number of native speakers in Peru. It is a language spoken by about 150 communities (mainly in the Amazon region) and is taught in almost 300 public schools in Peru (schools with a bilingual education program) [8] [1]. However, it does not have any own computational-linguistic resources yet, and this is the reason why it is considered a low-resourced language from a computational perspective, like most of the peruvian native languages.

SHP is an agglutinative language which relies in the use of around 114 suffixes plus 31 prefixes [12] and their combinations for word derivation. However, there is not an official grammar established, so, in order to develop computational-linguistic resources it was a must to relied on the assistance of linguistic experts and bilingual speakers.

4 Corpus: Building and Annotation

Because there is no annotated corpus for SHP, a new one was built with the required data for the job. This task was achieved with the development of an annotation tool

called ChAnot[3], the help of linguists with a vast knowledge of the language and some native speakers. It is important to note that they had no experience in annotation tasks. The final corpus for this study is available in a project site[4].

ChAnot is an annotation tool that allows to process a text by sentences and perform morphological (lemma and affixes), morpho-syntactic (POS-tag) and named-entity annotation. This tool was developed to make easier the process of creating an annotated corpus for peruvian low-resourced languages. Unlike annotators tools that allow highlighting parts of the document to annotate some information, the focus of this tool is to process a sentence sequentially word for word, allowing the splitting of its affixes and an specific annotation.

On the other side, a suitable tagset for the language was needed, and since Shipibo-konibo and most native languages in Peru do not have an official tagset, a linguist team defined a new one based on the standard tagset of Universal Dependencies [9] and linguistic studies regarding the language [12]. The tagset match with the UD standard names can be seen in the tool website. With the help of this tool, it was possible to develop a corpus of 219 annotated sentences, where each word of the sentence contains: an annotation of the lemma, POS-tag, sub-POS-tag, and a list of all the affixes that appears in the word.

This corpus was used entirely for the training of the POS-tagger. The distribution of the amount of words per tag in the Shipibo-konibo tagset is shown in Table 1.

Table 1: Structure of the corpus used in the POS-tagging task

POS Category	Quantity
Adjective	66
Adverb	40
Particle	1
Conjunction	38
Determiner	53
Interjection	6
Noun	368
Proper Noun	15
Numeral	6
Interrogative Word	59
Adposition	14
Pronoun	65
Punctuation	311
Verb	361
Auxiliary	95

Table 2: Structure of the corpus used in the Lemmatization task

POS Category	Quantity
Adjective	309
Adverb	130
Particle	1
Conjunction	29
Determiner	4
Interjection	11
Noun	1474
Proper Noun	0
Numeral	6
Interrogative Word	30
Adposition	22
Pronoun	32
Verb	1490
Auxiliary	6

[3] Available in: chana.inf.pucp.edu.pe/chanot
[4] Available in: chana.inf.pucp.edu.pe/resources

Furthermore, with the help of a Shipibo-konibo dictionary (which entries included POS-tags information), it was possible to identify the derived wordforms of lemmas that were presented in the examples of the use of each entry. In that way, the corpus of the lemmatization task could get more annotated examples. At the end, the corpus achieved a total of 3544 unique input words (with their correspondent lemma and POS-tag) distributed by word category as it is shown in Table 2.

5 Ship-LemmaTagger

5.1 Part-of-speech Tagging

Part-of-speech (POS) tagging is the process of assigning a part of speech to each word in a corpus [5]. For this process, it was defined a tagset aligned to the standard tagset of Universal Dependencies, and after that a supervised learning approach was considered.

The workflow for this step is shown in Figure 1. Firstly, a sentence is received as an input. Then, a tokenization step is performed, where the sentence is split in tokens (words, numbers, symbols or signs).Each token in a sentence is checked to observe whether it is a numeral, a symbol or a punctuation sign. If the token is one of the three possibilities mentioned before, the POS-tag is assigned directly, otherwise the trained supervised model comes into action.

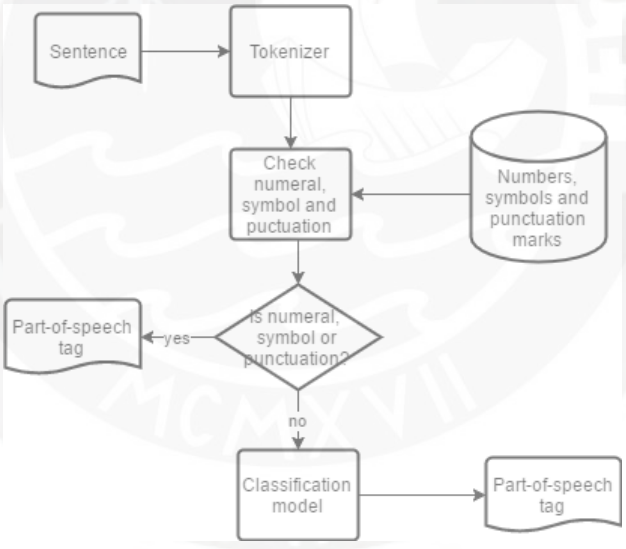

Fig. 1: Part-of-speech tagging process

For the classification task, the approach of Ekbal et al. [3] was taken in consideration, since they trained a Support Vector Machine (SVM) algorithm using different features such as word information (initial and final substring of the word, which could

be called prefixes and suffixes) and contextual information (previous word, previous POS-tags and next word). The complete list of the generated features is as follows:

- Current token
- Previous token
- Next token
- A binary value that indicates whether it is the first token of sentence or not.
- A binary value that indicates whether it is the last token of sentence or not.
- A binary value that indicates whether the first character of the token is capitalized.
- A binary value that indicates whether all characters of the token are capitalized.
- Prefixes (initial characters) of length 1, 2, 3 and 4.
- Suffixes (last characters) of length 1, 2, 3 and 4.
- Two previous POS-tags.

For instance, in the sentence ”Manaxawe betan chaxo inˊı” (that means ”The motelo and the deer”), the features regarding the information of word ”Manaxawe” are 1 (first token), 0 (not last token), 1 (first character capitalized), 0 (some characters are not capitalized), ”m”, ”ma”,”man”, ”mana” (prefixes) and ”e”,”we”,”awe”,”xawe” (suffixes). Meanwhile, the features for the contextual information of the word ”chaxo” are ”betan” (previous token), ”inˊı” (next token), conjunction (previous POS-tag) and noun (POS-tag before previous POS-tag).

5.2 Lemmatization

The lemmatization process follows the workflow shown in Figure 2. First, an individual input token is analyzed in order to determine if it could possess a suffix. This is done by contrasting the end of the word versus a list of all the existent suffixes identified in the Shipibo-konibo language.

In case there is a potential suffix present in the token, a possible rule is inferred with the use of a trained classification model. Once the potential rule is obtained, it is analyzed whether it could be applied for the input token to get the lemma. If the rule could no be used (there is no match) we retrieve the same word as the final lemma.

Regarding the rule prediction task, the approach of W. Daelemans [2] was followed, training a K-NN classification model using a number of features corresponding to the size of the biggest word of the corpus. In this feature vector, each character of the word is mapped to a dimension according to the position of the character in the word. Furthermore, since the language is highly agglutinative on the side of the suffixes, it was decided to reverse the order of the characters in a word to get an alignment between suffixes. On the other side, the derived rules of transformation were considered as the classes of the model.

The lemmatization rules are composed similarly to the ones shown in the previous work: two-elements tuples with (1) the string to be removed and (2) the string to be added to get the final lemma. In both cases, if there is no need to add or remove a string from the input word, the corresponding part of the tuple is left blank.

Additionally, since there are some particular suffixes that only appear in certain words categories, it was decided later to include the POS-tag as an additional feature.

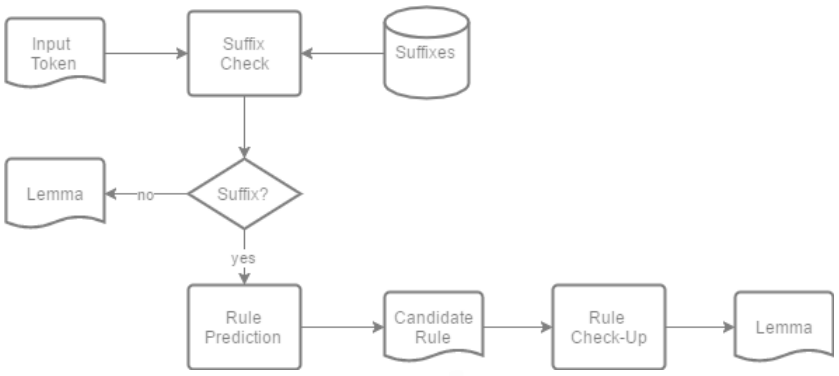

Fig. 2: Lemmatization process

For instance, for the word “ainbobo”, that means “women” in Shipibo-konibo, and its lemma “ainbo”, the features vector would be: [’o’ , ’b’ , o’ , ’b’ , ’n’ , ’i’ , ’a’ , ’noun’] and the rule of transformation would be [bo -] because we need to remove the substring ”bo” from the input word and add a null string (” ”) to get the lemma.

6 Experimentation and Results

Regarding the POS-tagger experiments, two methods were used: an SVM and a Decision Tree model. For the validation step, the corpus was split in sub-datasets for training (70%) and testing (30%). After repeating this split process 10, 50 and 100 times, the average accuracy of our part-of-speech tagger was obtained. The results are presented in Table 3.

Additionally, experiments with ensemble learning methods were tested, but the scores were lower than the expected. Finally, the best overall accuracy was 0.848, obtained with the SVM algorithm (kernel=RBF, C=1, gamma=0.1).

Table 3: Accuracy for part-of-speech tagging experiments

Algorithm \ Iterations	10	50	100
SVM	0.847	0.847	**0.848**
Decision Tree	0.808	0.810	0.811

For the lemmatization task using the K-NN algorithm, the performance was validated by splitting the corpus in two equal parts for training and testing (50-50). This division was made by stratifying every class of the corpus in two parts, in order to avoid the disproportion of some word categories with little data. This process was performed

100 times with random divisions each time, and the average accuracy obtained is presented in Table 4.

The experiment was fulfilled using different numbers of neighbors and distance metrics in order to find the optimal result. In this way, the best parameters configura- tion (neighbors=5, distance=Manhattan) achieved an overall accuracy of 0.593. This is caused by the presence of high number of features for this task, and with the Man- hattan measure, the relation between near features is isolated and the alignment of the characters obtained more relevance. Also, it is important to notice that the number of neighbors needed for the optimal result should not be too high, since that configuration could bias the results towards the rules with higher appearances in the corpus.

However, this result was not completely satisfactory in itself, but considering that only half of the corpus was used for training, it was a good step to then test it together with the POS-tagger.

Table 4: Accuracy results for the lemmatizer

Metric \ # of k	1	3	5	7	9	11	13
Euclidean	0.482	0.531	0.558	0.536	0.557	0.534	0.521
Chebyshev	0.486	0.514	0.543	0.539	0.558	0.539	0.541
Manhattan	0.502	0.539	**0.593**	0.562	0.547	0.556	0.551

Finally, both procedures were merged by using the best trained model of the POS- tagging step as an additional feature for the lemmatization. This new lemmatizer model was trained with the whole corpus obtained from the dictionary, and it was tested on the annotated sentences with ChAnot, that include a set of different words. With this procedure, it was obtained a new accuracy value of 81.4% for the trained lemmatizer as it is shown in Table 5.

Table 5: Accuracy results for the joint process

Metric \ # of k	1	3	5	7	9	11	13
Euclidean	0.805	0.565	0.507	0.486	0.474	0.483	0.476
Chebyshev	0.762	0.531	0.498	0.492	0.474	0.471	0.465
Manhattan	**0.814**	0.574	0.525	0.492	0.489	0.492	0.487

7 Conclusions and future work

This study focus on the developing of a basic NLP toolkit for a new language. As this language (SHP) is an agglutinative one, some approaches in similar contexts were taken in consideration in order to build a solid feature vector to fit learning models for the POS-tagger and Lemmatizer tasks.

The first results were uneven, highlighting the good performance of the POS-tagger. However, despite having achieved an individual low result for the lemmatization task, the integration with the POS-tagging process (as an input feature) led to very promising results in general. Likewise, since the approach used was a corpus-based, the continuous growth of the annotated corpus could lead to better accuracy results for both tasks.

As future work, semi-supervised learning methods will be considered for upcoming experiments. This approach could take advantage of the large unannotated corpus avail- able and, with the integration of the predictive models in the annotation tool, it could support the development of more linguistic resources for this language.

Acknowledgments For this study, the authors appreciate the linguistic team effort that made

possible the corpus annotation, and also acknowledge the support of the “Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica” (CONCYTEC Perú) under the contract 225-2015-FONDECYT.

References

1. Acosta, S., Natalia, K., Huamancayo Curi, E., Mori Clement, M., Carbajal Solis, V.: Docu- mento nacional de lenguas originarias del perú (2013)
2. Daelemans, W., Groenewald, H.J., Van Huyssteen, G.B.: Prototype-based active learning for lemmatization (2009)
3. Ekbal, A., Bandyopadhyay, S.: Part of speech tagging in bengali using support vector ma- chine. In: Information Technology, 2008. ICIT’08. International Conference on. pp. 106–111. IEEE (2008)
4. Gupta, V., Joshi, N., Mathur, I.: Design and development of a rule-based urdu lemmatizer. In: Proceedings of International Conference on ICT for Sustainable Development. pp. 161–169. Springer (2016)
5. Jurafsky, D., Martin, J.H.: Speech and language processing, vol. 3. Pearson (2014)
6. Jurs˘ic, M., Mozetic, I., Erjavec, T., Lavrac, N.: Lemmagen: Multilingual lemmatisation with induced ripple-down rules. Journal of Universal Computer Science 16(9), 1190–1214 (2010)
7. Khaltar, B.O., Fujii, A.: A lemmatization method for mongolian and its application to index- ing for information retrieval. Information Processing & Management 45(4), 438–451 (2009)
8. Ministerio de Educación del Perú: Minedu oficializa alfabetos de 24 lenguas originarias a ser utilizados por todas las entidades públicas. `http://www.minedu.gob.pe/n/noticia.php?id=33082`, accessed: 2016-31-03
9. Nivre, J., de Marneffe, M.C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C.D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., et al.: Universal dependencies v1: A multilingual treebank collection. In: Proceedings of the 10th International Conference on Language Re- sources and Evaluation (LREC 2016). pp. 1659–1666 (2016)
10. Paul, A., Purkayastha, B.S., Sarkar, S.: Hidden markov model based part of speech tagging for nepali language. In: Advanced Computing and Communication (ISACC), 2015 Interna- tional Symposium on. pp. 149–156. IEEE (2015)
11. Singh, S., Jha, G.N.: Statistical tagger for bhojpuri (employing support vector machine). In: Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on. pp. 1524–1529. IEEE (2015)
12. Valenzuela, P.: Transitivity in shipibo-konibo grammar. Ph.D. thesis, University of Oregon (2003)

Anexo 2: ChAnot: An Intelligent Annotation Tool for Indigenous and Highly Agglutinative Languages in Peru

ChAnot: An Intelligent Annotation Tool for Indigenous and Highly Agglutinative Languages in Peru

Rodolfo Mercado-Gonzales, José Pereira-Noriega, Marco Sobrevilla, Arturo Oncevay
Research Group on Pattern Recognition and Applied Artificial Intelligence
Departamento de Ingeniería, Pontificia Universidad Católica del Perú, Lima, Peru
{rmercado, msobrevilla, arturo.oncevay}@pucp.edu.pe, jpereira@pucp.pe

Abstract

Linguistic corpus annotation is one of the most important phases for solving Natural Language Processing (NLP) tasks, as these methods are deeply involved with corpus-based techniques. However, meta-data annotation is a highly laborious manual task. A supportive alternative requires the use of computational tools. They are likely to simplify some of these operations, while can be adjusted appropriately to the needs of particular language features at the same time. Therefore, this paper presents ChAnot, a web-based annotation tool developed for Peruvian indigenous and highly agglutinative languages, where Shipibo-Konibo was the case study. This new tool is able to support a diverse set of linguistic annotation tasks, such as word segmentation, POS-tag markup, among others. Also, it includes a suggestion engine based on historic and machine learning models, and a set of statistics about previous annotations.

Keywords: Annotation Tool, Corpus Annotation, Peruvian Indigenous Languages, Shipibo-Konibo

1. Introduction

The research field of Natural Language Processing (NLP) aims the analysis automation, representation and generation of human language through computational techniques (Cambria and White, 2014). These methods usually are based on supervised machine learning approaches, which employ the analysis of corpus, composed of annotated samples, and other strategies for reaching a particular NLP goal. The samples within a corpus need to be enriched with additional information or meta-data, plus the correct solution regarding the NLP task, generating an annotated linguistic corpus (Müller and Strube, 2006).

Nevertheless, the development of an annotated linguistic corpus might be a highly time-consuming task. The process requires mainly human effort, which is performed by linguists or experts in a particular language. While the corpus is more specialized or the language is less widespread, the expertise requirements will increase. In that way, an easier specialized annotation tool is what is needed. (Müller and Strube, 2006).

This paper introduces ChAnot, a new web-based annotation tool, which is focused in Peruvian indigenous and highly agglutinative languages. ChAnot enables the corpus annotation for various NLP tasks, such as Morphological Analysis (lemma and affixes segmentation), Part-of-Speech tagging, Name Entity Recognition and Syntactic Analysis (using a BRAT[1] interface (Stenetorp et al., 2012)). The interactive interface and architecture are adjusted appropriately to the needs of the Peruvian indigenous languages and humans annotators. Moreover, ChAnot includes a suggestion engine based in machine learning models and a set of important statistics about historic annotation.

The text below is organized as follow. Section 2 presents some existing annotation tools. Then, the main and distinctive functionalities in ChAnot are detailed in Section 3. Later, Section 4 presents briefly features regarding the Peruvian indigenous languages, focusing in Shipibo-Konibo, which is the case study language. Finally, conclusions and future works are discussed.

2. Related Work

Due to the increasing of NLP applications during the last years, an important number of annotation tools have been developed. Each tool has features according to specific objectives. Therefore, there are annotation tools supporting a set of NLP tasks independently of the language, while other are focused in a specific kind of languages.

One of the mainly important annotation software is MMAX2[2], which is a GUI-based tool that, like most of the other tools, lets the users to select a portion of text and annotate some properties over it (text span annotation). This process enables the markup of POS tag, word senses, coreference, dependency relations, among others. Besides, the XML format is used to store the meta-data (Müller and Strube, 2006).

Most of the current annotation tools are web-based, language independent, use machine learning for managing suggestions and support a variety set of text annotation tasks. Within this range, BRAT is likely to be the most popular one, which additionally includes high-quality annotation visualization and is fully configurable (Stenetorp et al., 2012). Another popular tool is WebAnno[3], which additionally offers annotation project management (including management of users and roles) (Yimam et al., 2013).

Likewise, as it was mentioned before, there are some annotation tools developed according to features of a specific language, such as Fassieh (Attia et al., 2009). This GUI-based tool lets the user to perform morphological, POS-tag, phonetic and semantic annotations for Arabic texts.

As it may be observed, the described tools have many features in common. Generally speaking, they are very helpful for different annotation tasks. Nevertheless, a customized

[1] Available in: http://brat.nlplab.org/

[2] Available in: http://mmax2.net/

[3] Available in: https://webanno.github.io

tool for the features and reality of Peruvian indigenous languages is what is pursuit in this study. The main motivations and specific functionalities are described in the next section.

3. ChAnot Annotation Tool

The main reason for developing a new tool was the need to exploit morphological-rich languages, due that word segmentation together with morpheme meta-data annotation (for POS-tagging or disambiguation) is not possible in most of the other tools, as far as it was noticed in the previous part. Another motivation is the lack of experience of the human annotators (linguists and native speakers), who have never done this kind of work before, and even some of them are beginners in the use of software.

In this context, ChAnot[4] is an intelligent web-based annotation tool focused in Peruvian indigenous and agglutinative languages, which allows text processing through morphological (lemma and affixes), morpho-syntactic (POS tag), named entity and syntactic annotation (with an interface integrated with BRAT).

The name of the tool is composed from the terms *Chana*, the native name of a bird that represents knowledge in the Shipibo-Konibo culture, and the first part of *anotador*, which means annotator in Spanish. The latter is the main official language in Peru, and the reason why the interface has included Spanish terminology.

Likewise, ChAnot was implemented using a client-server architecture that can be accessible from any modern web browser, using a back-end implemented in Java. The annotated data is stored on a MySQL database for most tasks, except for the syntactic annotation which is saved in BRAT format.

3.1. ChAnot Workflow

ChAnot was designed to perform two main annotation phases. The first phase is a morphological and morpho-syntactic annotation. Then, using the previous information, and with the help of a BRAT interface, enables a syntactic annotation phase. NER task annotation is also available, although not integrated in the main work flow.

- **Input Phase**
  ChAnot receives a plain text file in encoding UTF-8 with a sentence per line. Each sentence has two parts (separated by a vertical bar): the own sentence in indigenous language followed by the translation of this sentence in Spanish (translation is not necessary), as it is shown in Figure 1.

```
Rámara pápa Iquitoain iki | Ahora papá está en Iquitos
Bakisha ea kái | Voy a ir mañana
```

Figure 1: Input format sample

- **Annotation Phase**
  After uploading the input file to ChAnot, the user must select an annotation task to perform. The unique restriction is asked for performing syntactic annotation, as it is required a previous morphological and POS-tag annotation.

- **Output Phase**
  Finally, the annotation is saved in a database. It can also be exported in a XML file structured by sentences, words and affixes. Figure 2 presents an output XML sample.

Figure 2: Output format sample

3.2. ChAnot Functional Features

Unlike other tools, ChAnot enables a complete morphological annotation (lemma plus affixes segmentation with annotation) and posses a communication interface with BRAT for dependency syntax annotation using the previous meta-data. The main features of ChAnot are detailed below.

- **Accessibility**
  Since the intended users of ChAnot are people who are not very familiar with computers or technology, it is a must to have a tool that can be accessible from anywhere and without the need to install any complicated software. In order to accomplish that, ChAnot is accessible from any modern web browser.

- **User Management**
  Each human annotator has a private account to work in ChAnot. This account let them to manage their annotation files through a menu (see Figure 3). There is no crossover of information and no possibility of getting corrupted data by external users either.

- **Statistics**
  ChAnot generates a set of important statistic per annotation file and per users, such as the current number of annotated sentences or words, and even the average time that was spent for the annotation task of each sentence. These statistics could be very useful as a complexity metric in the evaluation of the historically annotated sentences. Besides, it will be useful to evaluate the progress of human annotators for further analysis.

- **Interactive Interface**
  In order to ease annotation tasks for users, the interfaces were designed to be intuitive and for transmitting information trough different colors. In most of the ChAnot interfaces, green color means processed or completely annotated, yellow refers to work in progress, while red represents a not processed task or data. Figures 3 and 5 illustrates the color usage.

  Likewise, all text and messages in this tool are in Spanish, because it is the most spoken language in Peru.

[4]Source code and video demo available in: chana.inf.pucp.edu.pe/resources/chanot

Figure 3: A user's main menu interface with a list of documents to annotate. The different colors represents the progress in each file.

- **Automatic Tokenization**
  ChAnot automatically tokenizes the input sentences in words, numbers, symbols and punctuation symbols. The split is performed for morphological, POS-tag and named entity annotation. This feature differs from most of other tools, which are mainly based on text span annotation.

- **BRAT Interface**
  Additionally, it was decided to take advantage of the BRAT capabilities in dependencies annotation for syntax, so a direct connection was configured from ChAnot.

  In the interface, each annotated sentence of the file is transformed into the BRAT files format. For that purpose, each word is split in its morphemes according the morphological annotation in ChAnot, which is entirely preserved (see the details in subsection 3.3). Figure 4 presents an annotation task in BRAT, where the POS-tags and word segmentation was obtained from the ChAnot interface.

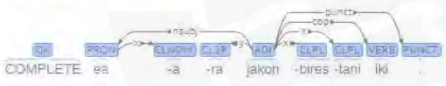

Figure 4: Dependencies annotation with a BRAT interface.

  It is important to highlight, that some dependency rules for annotation were introduced in the interface, in order to reduce the workload for the annotators in BRAT.

3.3. NLP Annotation Tasks in ChAnot

- **Morphological Annotation**
  In the morphological scope, annotation through ChAnot allows a complete word segmentation. For each word in a sentence, the lemma is introduced, and if applicable all the affixes could be added. The lemmas and morphemes must be annotated with their respective category, which are customizable regarding the language. Besides, the relative position of the sub-word units in the term is a requirement for finishing each word annotation. Figure 5 shows an example for a word with two affixes.

  As it is noted, the annotation includes the most relevant morphological information of each word plus the full meaning of each affix and their categories. The corpus can be exploited for other tasks such as word segmentation or morphological disambiguation.

- **POS-Tag Annotation**
  ChAnot allows Part-of-Speech tagging using a two-level predefined POS-tagset. The first level provides a general information about grammatical properties of a word, while the second describes more granulated information regarding the different kinds of the upper level.

  The tagset can be modified according the preferences or some specific language features. For the case study (Section 4), the definition of a POS-tagset aligned to the UD standard (Nivre et al., 2016) was an important factor.

- **Named Entity Annotation**
  As a complementary function, it is possible to identify if there is a named entity (NE) in the text. In ChAnot, the annotation is performed by introducing the specific NE category and a relative position tag, that discriminates single and multi-word entities in the sentence.

- **Syntactic Dependencies Annotation**
  A Universal Dependency (UD) Treebank (Nivre et al., 2016) is the main resource-like goal for the Peruvian indigenous languages (work in-progress). For that purpose, ChAnot includes a BRAT interface, as it was previously described.

3.4. Automatic Suggestions in ChAnot

ChAnot has integrated three different machine learning-based models that assist the workload of the annotators. One for the automatic identification of POS-tags, the second for the lemma prediction and a third for named entities recognition. The first two models are part of the ShipLemmaTagger toolkit (Pereira-Noriega et al., 2017), while the latter one is a newly hybrid model that uses a combination of rules and predictions based on previous annotations. All of these predictive models are implemented as Python web services in the server side. Furthermore, they are automatically updated in a periodic scheme with data coming from new annotations that are made by the users. As these models are frequently adjusted, the quality improvement of their suggestions is what is expected.
Furthermore, there are two kind of suggestions embedded in ChAnot: historic and machine learning-based. The former recalls the previous annotations (lemma, tags and affixes) that any user performed in the past, and presents the suggestion marking the word with a yellow fill. The latter one works as it was described before, and the suggestions are presented for entirely new words, highlighting them with a red fill. Figure 5 presents a sample.

4. Case Study: Shipibo-Konibo (shp)

4.1. Background

Peru presents a diverse culture map including many indigenous communities and cultures, who are minorities in the country. In order to support the preservation of their traditions and languages, the Ministry of Culture of Perú has identified 19 linguistic families including 47 indigenous languages, and 24 of them have been made official for government-service purposes. Among them, Shipibo-Konibo is the sixth language with the highest number of native speakers, with about 22 517 speakers, and is taught

Figure 5: Morphological annotation sample (*Who made a competition?*): the word *Tsoabaonki* was split in *tsoa +baon +ki*, and each sub-word unit received additional information. Regarding the colors, the 3rd and 4th terms (red) are not annotated but they contain machine learning-based suggestions, while the 5th term (yellow) includes a historic-based suggestion.

in 299 public schools through bilingual educational programs (Ministerio de Cultura del Perú, 2016).

Shipibo-Konibo is a highly agglutinative language, with more than 100 suffixes and about 13 prefixes for word inflection (Valenzuela, 2003). Besides, there are not too many academic experts with experience in computational annotation tasks. In that context, ChAnot reflects as much information of this language for solving specific NLP tasks, while at the same time tries to be as easy as possible for unexperienced annotators.

4.2. Corpus Annotation and Evaluation

Using ChAnot, the experts could develop a corpus of 1630 annotated sentences, where each word within them contains: annotation of lemma, POS-tag, sub-POS-tag, and a list of all the affixes that compose the word. These affixes include category and relative positions. Besides, 204 and 78 sentences got named entity and dependencies (with the BRAT interface) annotations, respectively. However, the latter two tasks are not part of the analysis.

An evaluation of the machine learning-based suggestion engine was performed. It was simulated an increasing annotation scheme with automatically updated models in a 300-sentence block. In this sense, the accuracy achieved a peak of 74% for lemmatization, and 89% for POS-tagging (see Figure 6).

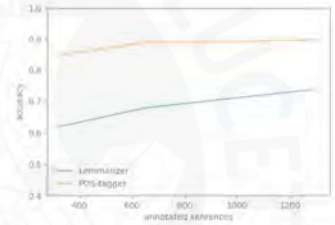

Figure 6: Evaluation of the lemmatizer and POS-tagger integrated in ChAnot

5. Conclusions and Future Work

The need for an annotation tool customized around the features of Peruvian indigenous languages allowed the design and implementation of ChAnot. The tool is likely to speed up the construction of linguistic corpora, by presenting suggestions based on past annotations samples, as well as predicting suggestions for new words with machine learning models that are frequently adjusted with newly annotated data. Furthermore, the wide range of functional features and annotations tasks are core points of the tool, which has the potential to scale easily for more complex jobs.

Future work is focused in the functional aspect. First, crowdsourcing, as it is included in WebAnno (Yimam et al., 2013), may be a relevant feature for enable the participation of more experts. However, the simultaneously integration of crowdsourcing and active learning schemes is what is really desired. The initial steps has been already made, as there are now some predictive models embedded in the tool. Another functional feature would be the inclusion of a spell-checker for the indigenous language, as it might work as a previous validation step for the required text to annotate (Alva and Oncevay, 2017).

Furthermore, there are new tasks that are expected to be annotated in the short term. Alignment is one of them, due to the presence of the translated text in Spanish (from parallel corpora). That could help enormously in the corpus-based machine translation experiments that has been performed recently (Galarreta et al., 2017). Finally, other NLP level tasks may be supported in the future, such as the semantic layer with word sense disambiguation annotations.

6. Bibliographical References

Alva, C. and Oncevay, A. (2017). Spell-checking based on syllabification and character-level graphs for a Peruvian agglutinative language. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 109–116.

Attia, M., Rashwan, M. A., and Al-Badrashiny, M. A. (2009). Fassieh, a semi-automatic visual interactive tool for morphological, PoS-Tags, phonetic, and semantic annotation of Arabic text corpora. *IEEE transactions on audio, speech, and language processing*, 17(5):916–925.

Cambria, E. and White, B. (2014). Jumping NLP curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57.

Galarreta, A. P., Melgar, A., and Oncevay, A. (2017). Corpus creation and initial SMT experiments between Spanish and Shipibo-Konibo. In *RANLP (In-press)*.

Ministerio de Cultura del Perú. (2016). Base de datos de pueblos indígenas u originarios - Pueblos indígenas del Perú. http://bdpi.cultura.gob.pe/lista-de-pueblos-indigenas. Accessed: 2016-31-03.

Müller, C. and Strube, M. (2006). Multi-level annotation of linguistic data with MMAX2. *Corpus technology and language pedagogy: New resources, new tools, new methods*, 3:197–214.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R. T., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal Dependencies v1: A multilingual treebank collection. In *LREC*.

Pereira-Noriega, J., Mercado-Gonzales, R., Melgar, A., Sobrevilla-Cabezudo, M., and Oncevay-Marcos, A. (2017). Ship-LemmaTagger: Building an NLP toolkit for a Peruvian native language. In *International Conference on Text, Speech, and Dialogue*, pages 473–481. Springer.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: A web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Valenzuela, P. (2003). *Transitivity in Shipibo-Konibo grammar*. Ph.D. thesis, University of Oregon.

Yimam, S. M., Gurevych, I., de Castilho, R. E., and Biemann, C. (2013). WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *ACL (Conference System Demonstrations)*, pages 1–6.

Anexo 3: Ejemplo de anotación generado en formato XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<corpus>
   <lastOracion>23</lastOracion>
   <oracion id="0" texto="Awakanronki ronón ewaki kóshi méenike" traduccion="Cuentan que en la antigüedad la sachavaca y la boa hicieron una competencia de fuerza">
      <palabra catGram="Nombre" id="0" lema="Awakan" subCatGram="contable" token="Awakanronki">
         <afijo texto="ronki" tipo="clít. de segunda posición"/>
      </palabra>
      <palabra catGram="Nombre" id="1" lema="rono" subCatGram="contable" token="ronón">
         <afijo texto="n" tipo="sufijo nominal"/>
      </palabra>
      <palabra catGram="Postposición" id="2" lema="sufijo aumentativo de ronon" subCatGram="comitiva" token="ewaki">
         <afijo texto="ki" tipo="clít. de segunda posición"/>
      </palabra>
      <palabra catGram="Adjetivo" id="3" lema="kóshi" subCatGram="calificativo" token="kóshi"/>
      <palabra catGram="Verbo" id="4" lema="méeti" subCatGram="transitivo" token="méenike">
         <afijo texto="ke" tipo="sufijo verbal"/>
         <afijo texto="ni" tipo="sufijo verbal"/>
      </palabra>
   </oracion>
   <oracion id="1" texto="Awai xetánra chópa kénéati iki" traduccion="Con el pico de la gaviota se pintan diseños en la tela">
      <palabra catGram="Nombre" id="0" lema="Awai" subCatGram="contable" token="Awai"/>
      <palabra catGram="Nombre" id="1" lema="xéta" subCatGram="contable" token="xetánra">
         <afijo texto="ra" tipo="clít. de segunda posición"/>
         <afijo texto="n" tipo="clít. nominal"/>
      </palabra>
      <palabra catGram="Nombre" id="2" lema="chópa" subCatGram="contable" token="chópa"/>
      <palabra catGram="Verbo" id="3" lema="kénéa" subCatGram="transitivo" token="kénéati">
         <afijo texto="ti" tipo="sufijo verbal"/>
      </palabra>
      <palabra catGram="Verbo" id="4" lema="iki" subCatGram="transitivo" token="iki"/>
   </oracion>
   <oracion id="2" texto="Jonínra aínbo bíke awájonra raónxon" traduccion="Un hombre consiguió a su mujer por esposa, porque la curó con awajonra">
      <palabra catGram="Nombre" id="0" lema="Joni" subCatGram="contable" token="Jonínra">
         <afijo texto="ra" tipo="clít. de segunda posición"/>
         <afijo texto="n" tipo="clít. nominal"/>
      </palabra>
      <palabra catGram="Nombre" id="1" lema="aínbo" subCatGram="contable" token="aínbo"/>
      <palabra catGram="Verbo" id="2" lema="biti" subCatGram="transitivo" token="bíke">
         <afijo texto="ke" tipo="sufijo verbal"/>
      </palabra>
      <palabra catGram="Nombre" id="3" lema="awájonra" subCatGram="contable" token="awájonra"/>
      <palabra catGram="Verbo" id="4" lema="ráonti" subCatGram="transitivo" token="raónxon">
         <afijo texto="xon#2" tipo="sufijo verbal"/>
      </palabra>
   </oracion>
<oracion id="3" texto="Moátianronki isónki awántsonin róo tetónki tsákanike ; jáskatironki róo teton áni lki" traduccion="En la antigüedad el maquisapa le tiró el tutumo al cotomono en el cuello; por eso el cotomono se quedó así">
```

Anexo 4: Muestra de corpus utilizado para Sklearn y TIMBL

sámo	sámo	nombre	
sánitarion	sánitario	nombre	
sánto	sánto	nombre	
sáquiicai	sáquiiti	verbo	
sárajinto	sárajinto	nombre	
sárisariicai	sárisariiti	verbo	
sárorantin	sároranti	verbo	
sároro	sároro	nombre	
sáya	sáya	nombre	
sébe	sébe	nombre	
séicai	séiti	verbo	
sénbique	sébiti	verbo	
sénenain	sénenain	adverbio	
sénenainco	sénenainco	postposición	
sénoque	sénoti	verbo	
sépaque	sépati	verbo	
séqueque	séqueti	verbo	
sése	sése	adjetivo	
séseai	séseati	verbo	
séseai	séseti	verbo	
síbara	síbara	nombre	
síbuaamabi	síhuati	verbo	
sícati	sícati	verbo	
síhuatai	sihuáti	verbo	
síi	síin	nombre	
síique	síiti	verbo	
sínatai	sináti	verbo	
sínco	sínco	nombre	
sínopepo	sínopepo	nombre	
sínta	sínta	nombre	
sío	sío	nombre	
síquinoninra	síquino	nombre	
síra	síra	nombre	
síranai	siránti	verbo	
síribinque	síribinti	verbo	
síro	síro	adjetivo	
síroaque	síroati	verbo	
sóai	sóati	verbo	
sóatibo	sóati	nombre	
sócota	sócota	adjetivo	
sóitai	soíti	verbo	
sóoique	sóoiti	verbo	
sóro	sóro	nombre	
sórotai	soróti	verbo	
sótanai	sotánti	verbo	
tabla	tabla	nombre	
taca	táca	nombre	
tacácho	tacácho	nombre	
tacárinin	tacári	nombre	
tacáshitai	tacáshiti	verbo	
tae	tae	nombre	
taen	tae	nombre	
taen	taen	nombre	
taenko	tae	nombre	
tahuín	tahuín	nombre	
taka	taka	nombre	
takan	taka	nombre	
tako	tako	nombre	

takoki tako nombre
tama tama nombre
tama táma nombre
taménoai taménoti verbo
taménooque taménooti verbo
tan tan onomatopeya
tana tanati verbo
tanaaki tanati verbo
tanake tánati verbo
tanakin tanati verbo
tanhuámpari tahuámpari nombre
tankinki tankinki nombre
tano tano adjetivo
tanon tano adjetivo
tansharina tansharina nombre
tansóoicai tansóoiti verbo
tantia tántiti verbo
tantii tantiti verbo
tantiparinon tantiti verbo
tantiwe tántiti verbo
tanánon tanati verbo
tapamanribi tapaman nombre
tapo tapo nombre
tapomanra tapón nombre
tapon tapon nombre
tapon tapón nombre
taponribi tapon nombre
taponya tapón nombre
tapán tapán nombre
tapíriba tapíriba nombre
tapó tapó nombre
tarampa tarampa nombre
tarampanin tarampanin nombre
tari tari nombre
tariati tariati verbo
tariaxon tariati verbo
tariki tari nombre
tarironki tari nombre

Anexo 5: Parte del árbol generado por Lemmagen

```
`---> RULE:( suffix("") transform(""-->"") except(13) ); {:
    |
    |---> RULE:( suffix(";") transform(";"-->"") except(2) ); {:
    |    |
    |    |---> RULE:( suffix("que;") transform("que;"-->"ti") );
    |    `---> RULE:( suffix("ai;") transform("ai;"-->"ti") ); :}
    |
    |---> RULE:( suffix("a") transform(""-->"") except(19) ); {:
    |    |
    |    |---> RULE:( suffix("aa") transform("a"-->"ti") except(11) ); {:
    |    |    |
    |    |    |---> RULE:( suffix("baa") transform("a"-->"ti") except(2) ); {:
    |    |    |    |
    |    |    |    |---> RULE:( suffix("ribaa") transform("ribaa"-->"ti") );
    |    |    |    `---> RULE:( suffix("ríbaa") transform("ríbaa"-->"ti") except(1) ); {:
    |    |    |         |
    |    |    |         `---> RULE:( suffix("iríbaa") transform("ríbaa"-->"ki") ); :}
    |    |    |          :}
    |    |    |
    |    |    |---> RULE:( suffix("caa") transform(""-->"ti") );
    |    |    |---> RULE:( suffix("eaa") transform("eaa"-->"éati") except(1) ); {:
    |    |    |    |
    |    |    |    `---> RULE:( suffix("taweaa") transform("taweaa"-->"weati") ); :}
    |    |    |
    |    |    |---> RULE:( suffix("haa") transform("a"-->"ti") except(2) ); {:
    |    |    |    |
    |    |    |    |---> RULE:( suffix("chaa") transform("chaa"-->"") );
    |    |    |    `---> RULE:( suffix("rihaa") transform("rihaa"-->"nhati") ); :}
    |    |    |
    |    |    |---> RULE:( suffix("jaa") transform(""-->"") );
    |    |    |---> RULE:( suffix("skaa") transform("aa"-->"áa") except(1) ); {:
    |    |    |    |
    |    |    |    `---> RULE:( suffix("awekeskaa") transform("awekeskaa"-->"áwekeskaati") ); :}
    |    |    |
    |    |    |---> RULE:( suffix("amaa") transform("a"-->"ti") except(3) ); {:
    |    |    |    |
    |    |    |    |---> RULE:( suffix("iamaa") transform("amaa"-->"ti") );
    |    |    |    |---> RULE:( suffix("mamaa") transform("mamaa"-->"") );
    |    |    |    `---> RULE:( suffix("yamaa") transform("yamaa"-->"ti") except(3) ); {:
    |    |    |         |
    |    |    |         |---> RULE:( suffix("yamaa") entireword transform("a"-->"ti") );
    |    |    |         |---> RULE:( suffix("wetsayamaa") transform("wetsayamaa"-->"ti") );
    |    |    |         `---> RULE:( suffix("nyamaa") transform("yamaa"-->"ti") except(2) ); {:
    |    |    |              |
    |    |    |              |---> RULE:( suffix("inanyamaa") transform("anyamaa"-->"ánti") );
    |    |    |              `---> RULE:( suffix("xonyamaa") transform("xonyamaa"-->"ti") ); :}
    |    |    |               :}
    |    |    |          :}
    |    |    |
    |    |    |---> RULE:( suffix("inaa") transform(""-->"ti") );
    |    |    |---> RULE:( suffix("raa") transform("a"-->"ti") except(2) ); {:
    |    |    |    |
    |    |    |    |---> RULE:( suffix("eraa") transform("eraa"-->"érati") );
    |    |    |    `---> RULE:( suffix("yoraa") transform("a"-->"ti") except(2) ); {:
    |    |    |         |
    |    |    |         |---> RULE:( suffix("eyoraa") transform("eyoraa"-->"éti") );
    |    |    |         `---> RULE:( suffix("kanyoraa") transform("kanyoraa"-->"ti") ); :}
    |    |    |          :}
```

```
|     |     |
|     |     |---> RULE:( suffix("saa") transform(""-->"iti") );
|     |     `---> RULE:( suffix("otaa") transform("otaa"-->"ótati") ); :}
|     |
|     |---> RULE:( suffix("ba") transform(""-->"") except(3) ); {:
|     |     |
|     |     |---> RULE:( suffix("iweaba") transform("iweaba"-->"íweaba") );
|     |     |---> RULE:( suffix("iba") transform("iba"-->"") except(2) ); {:
|     |     |     |
|     |     |     |---> RULE:( suffix("aiba") transform("aiba"-->"") except(1) ); {:
|     |     |     |     |
|     |     |     |     `---> RULE:( suffix("paiba") transform("iba"-->"") ); :}
|     |     |     |
|     |     |     `---> RULE:( suffix("boiba") transform("boiba"-->"") ); :}
|     |     |
|     |     `---> RULE:( suffix("ába") transform("ába"-->"aba") ); :}
|     |
|     |---> RULE:( suffix("ca") transform(""-->"") except(2) ); {:
|     |     |
|     |     |---> RULE:( suffix("aca") transform("ca"-->"ti") except(3) ); {:
|     |     |     |
|     |     |     |---> RULE:( suffix("paca") transform(""-->"") );
|     |     |     |---> RULE:( suffix("taca") transform("aca"-->"áca") except(1) ); {:
|     |     |     |     |
|     |     |     |     `---> RULE:( suffix("itaca") transform("itaca"-->"ítaca") ); :}
|     |     |     |
|     |     |     `---> RULE:( suffix("xaca") transform(""-->"") ); :}
|     |     |
|     |     `---> RULE:( suffix("bóca") transform("óca"-->"ocánra") ); :}
|     |
|     |---> RULE:( suffix("ea") transform("a"-->"ti") except(7) ); {:
|     |     |
|     |     |---> RULE:( suffix("ea") entireword transform("a"-->"") );
|     |     |---> RULE:( suffix("kea") transform(""-->"") except(1) ); {:
|     |     |     |
|     |     |     `---> RULE:( suffix("skea") transform("a"-->"ti") ); :}
|     |     |
|     |     |---> RULE:( suffix("mea") transform("mea"-->"") except(2) ); {:
|     |     |     |
|     |     |     |---> RULE:( suffix("amamea") transform("mamea"-->"n") );
|     |     |     `---> RULE:( suffix("ámea") transform("ámea"-->"a") ); :}
|     |     |
|     |     |---> RULE:( suffix("kenea") transform(""-->"ti") );
|     |     |---> RULE:( suffix("sea") transform("ea"-->"ca") );
|     |     |---> RULE:( suffix("itea") transform(""-->"") );
|     |     `---> RULE:( suffix("awea") transform("awea"-->"áweti") ); :}
|     |
|     |---> RULE:( suffix("ha") transform(""-->"") except(2) ); {:
|     |     |
|     |     |---> RULE:( suffix("echa") transform("echa"-->"écha") );
|     |     `---> RULE:( suffix("sha") transform(""-->"") except(2) ); {:
|     |           |
|     |           |---> RULE:( suffix("risha") transform("isha"-->"íshti") );
|     |           `---> RULE:( suffix("ssha") transform("sha"-->"") ); :}
|     |            :}
|     |
|     |---> RULE:( suffix("ia") transform(""-->"") except(12) ); {:
|     |     |
|     |     |---> RULE:( suffix("aia") transform("a"-->"") );
|     |     |---> RULE:( suffix("bia") transform(""-->"") except(2) ); {:
```

```
|   |   |   |
|   |   |   |---> RULE:( suffix("ribia") transform("ribia"-->"ti") except(1) ); {:
|   |   |   |   |
|   |   |   |   `---> RULE:( suffix("oribia") transform("oribia"-->"óti") ); :}
|   |   |   |
|   |   |   `---> RULE:( suffix("?bia") transform("a"-->"ti") ); :}
|   |   |
|   |   |---> RULE:( suffix("oshia") transform("a"-->"ti") except(1) ); {:
|   |   |   |
|   |   |   `---> RULE:( suffix("toshia") transform("oshia"-->"i") ); :}
|   |   |
|   |   |---> RULE:( suffix("kia") transform("a"-->"") except(1) ); {:
|   |   |   |
|   |   |   `---> RULE:( suffix("ishkia") transform("ishkia"-->"íshkti") ); :}
|   |   |
|   |   |---> RULE:( suffix("mia") transform(""-->"") except(2) ); {:
|   |   |   |
|   |   |   |---> RULE:( suffix("amia") transform("a"-->"") );
|   |   |   `---> RULE:( suffix("imia") transform("imia"-->"ími") ); :}
|   |   |
|   |   |---> RULE:( suffix("nia") transform(""-->"") except(3) ); {:
|   |   |   |
|   |   |   |---> RULE:( suffix("enia") transform("a"-->"ti") );
|   |   |   |---> RULE:( suffix("inia") transform(""-->"") except(2) ); {:
|   |   |   |   |
|   |   |   |   |---> RULE:( suffix("hinia") transform("a"-->"ti") );
|   |   |   |   `---> RULE:( suffix("winia") transform("inia"-->"íniti") ); :}
|   |   |   |
|   |   |   `---> RULE:( suffix("onia") transform("a"-->"") except(1) ); {:
|   |   |       |
|   |   |       `---> RULE:( suffix("konia") transform("konia"-->"") ); :}
|   |   |       :}
|   |   |
|   |   |---> RULE:( suffix("oia") transform("a"-->"ti") except(1) ); {:
|   |   |   |
|   |   |   `---> RULE:( suffix("poia") transform("a"-->"") ); :}
|   |   |
|   |   |---> RULE:( suffix("pia") transform("a"-->"ti") except(1) ); {:
|   |   |   |
|   |   |   `---> RULE:( suffix("bápia") transform("a"-->"") ); :}
|   |   |
|   |   |---> RULE:( suffix("sia") transform("a"-->"á") );
|   |   |---> RULE:( suffix("tia") transform(""-->"") except(2) ); {:
|   |   |   |
|   |   |   |---> RULE:( suffix("antia") transform("antia"-->"ántiti") );
|   |   |   `---> RULE:( suffix("?tia") transform("a"-->"") ); :}
|   |   |
|   |   |---> RULE:( suffix("quia") transform(""-->"") except(2) ); {:
|   |   |   |
|   |   |   |---> RULE:( suffix("nquia") transform("a"-->"") );
|   |   |   `---> RULE:( suffix("?quia") transform("a"-->"ti") ); :}
|   |   |
|   |   `---> RULE:( suffix("?ia") transform("a"-->"ti") ); :}
|   |
```

Anexo 6: Muestra de clases encontradas en el corpus

n>
>
n>
tian>
al>ái
a>á
>
nin>
ra>
>
>
nique>ti
ahuínhu>aínhn
aon>o
aonki>o
>
ai>aí
an>
bo>
boki>
ki>
meax>
tsiki>
yabicho>
nra>
k>ti
kbo>ti
ki>ti
kibo>ti
>
akaianronki>a
kmponike>ti
kn>ti
kn>ti
knbo>ti
ai>ti
kni>ti
akanaian>a
akanaianki>a
kana>ti
knike>ti
>
akan>a
we>ti
knwe>ti
knyor>ti
ksir>ti
akasktai>
akas>a
akkanai>
kx>ti
kx>ti