PROYECTO DE GRADO

INSTITUTO DE COMPUTACIÓN
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA

Detección de sujetos omitidos en el español

Lucía GONZÁLEZ (lucia.gonzalez@fing.edu.uy)
Verónica MARTÍNEZ (vero.martinez.am@gmail.com)

Tutores:
Luis Chiruzzo
Dina Wonsever

Diciembre de 2018

Resumen

El siguiente documento describe el trabajo realizado durante el desarrollo de un clasificador capaz de identificar el fenómeno de los sujetos omitidos en el idioma español.

Para la realización del mismo se investigó el estado del arte de este problema de manera global y específicamente enfocado al idioma español. Se tomó como base un trabajo similar realizado sobre un corpus creado para esta tarea.

Una vez lograda la replicación del trabajo base, se procedió a extender el mismo con la creación de nuevos atributos, y el entrenamiento sobre un corpus más grande, más general, y no creado específicamente para esta tarea, el corpus AnCora.

Finalmente, se evaluó el rendimiento del clasificador elaborado, tanto sobre un fragmento de evaluación del corpus AnCora, donde se obtienen muy buenos resultados, como sobre un corpus completamente distinto. Se generó además una herramienta interactiva que simplifica su utilización. El código de la misma expone de manera simple el pipeline utilizado para la extracción de cláusulas, así como los pasos realizados para la evaluación de sus atributos y finalmente la utilización del clasificador ya entrenado para su clasificación.

Índice general

Índice de cuadros

Capítulo 1

Introducción

1.1. Descripción del problema

El idioma español pertenece a una categoría de lenguajes que permite la omisión de sujetos para la formación de oraciones válidas. Esta categoría, conocida como *pro-drop*, a la cual también pertenecen lenguas como el italiano y el portugués, se caracteriza por admitir la omisión de pronombres. Es decir, la oración *"∅ Salí a correr"* es sintácticamente correcta, por más que el pronombre *"yo"* está ausente de la misma[1].

Una explicación plausible para que en el idioma español se tomen como válidas estas oraciones es que toda la información que el oyente o lector necesita para descifrar el enunciado, ya sea número, persona, modo y/o tiempo, se encuentra presente en el verbo conjugado. Por ejemplo en la oración *"Comeremos manzanas."* se puede saber a partir del verbo conjugado *"comeremos"* el tiempo en el cual esta conjugado. En cambio lenguas no pro-drop no pueden transmitir toda esa información en una sola palabra, por ejemplo la misma oración en inglés si solo se utiliza el verbo *"eat"*, no se puede saber que persona es, ni en qué tiempo está conjugado, para conjugarlo en el mismo tiempo que tiene *"comeremos"*, se debe agregar el pronombre *"we"* y el verbo modal *"will"*, por lo que la oración quedaría *"We will eat apples."*

Se pueden encontrar muchas instancias en donde una oración es aceptable

[1]Se utiliza el símbolo ∅ para marcar el lugar que ocuparía.

independientemente de si incluye de manera explícita al sujeto o no. Por ejemplo, las oraciones *"**Yo** saqué a pasear al perro"* y *"∅ Saqué a pasear al perro"* contienen la misma información. Sin embargo, en casos como este, se generan connotaciones semánticas, dado que la inclusión del sujeto sirve para enfatizar quién realizó la acción[10]. La alternancia entre la presencia y la ausencia del sujeto no es siempre posible, y de las veces que sí lo es, no tiene por qué estar relacionada con un carácter constrastivo del discurso.[11]

Se dice entonces que una oración tiene sujeto omitido, elíptico o tácito cuando a pesar de no estar explícito se puede obtener toda su información a partir del verbo principal de la misma. Por otra parte, también existen oraciones que carecen de sujeto, estas se denominan oraciones impersonales y no se deben confundir con las que tienen sujeto omitido. Por ejemplo *"Está bailando bajo la lluvia."* no es una oración impersonal sino que existe el sujeto omitido él o ella.

Las oraciones impersonales son de tres tipos:

- Oraciones con verbos que indican fenómenos meteorológicos. Los verbos de fenómenos meteorológicos siempre forman oraciones impersonales. Por ejemplo: *"En invierno **amanece** sobre las siete"*.
- Oraciones con el pronombre se. Se construyen con el pronombre se y un verbo en la tercera persona del singular. Por ejemplo: *"**Se trata** de mejorar el proceso"*.
- Oraciones con los verbos ser, haber y hacer en tercera persona del singular. Las oraciones en las cuales se utilizan estos verbos para expresar condiciones meteorológicas o el paso del tiempo, son también oraciones impersonales. Por ejemplo: *Ya **es** muy tarde.*

Estas características del idioma español generan problemas al intentar procesar oraciones con métodos de aprendizaje automático, dado que no siempre es claro cuál es el sujeto de una oración. La detección de este fenómeno es además fundamental para resolver casos de anáfora durante procesamiento de información. Con el término "anáfora" se refiere a la referencia o repetición de algo anterior, en los casos donde se omite un sujeto, generalmente se puede encontrar el mismo en algún pasaje previo de un texto, por lo que es útil resolver estas co-referencias. Ejemplo: *Juan comió temprano. Tenía mucha hambre.*

Existen analizadores sintácticos que muchas veces poseen la información de cual es el sujeto del verbo, como el parser de dependencias de FreeLing[15], el

cuál cuando analiza la oración *Juan comió temprano.* etiqueta a *Juan* como el sujeto. Pero estas herramientas no están preparadas para los casos donde el sujeto es omitido o no existe, donde se debe realizar un proceso adicional.

1.2. Motivación

Dado que como se menciona en la sección anterior, las instancias de sujetos omitidos son sumamente comunes en el idioma español, es necesario contar con un proceso confiable capaz de detectarlas. Este es el primer paso para lograr la resolución de co-referencias.

La resolución de co-referencias es un problema importante, que afecta el problema de traducciones automáticas, o extracción de información semántica de textos no estructurados. En el primer caso, existe un problema claro al intentar realizar una traducción de un idioma que admite sujetos omitidos a uno que no, como el caso de una traducción de español a inglés. En este caso es necesario reconocer que existe una instancia de un verbo cuyo sujeto está ausente para luego poder sustituirlo con el sujeto apropiado. Por otra parte, si se intenta realizar extracción de información semántica (por ej. conceptos, eventos, o hechos) en un texto no estructurado como artículos periodísticos o libros, es necesario lograr reconstruir la relación entre la información y el sujeto al cual se refieren, dado que dicha información normalmente no se repite en cada ocasión.

Para lograr encarar los temas mencionados, el primer paso siempre resulta ser la identificación de sujetos omitidos en oraciones en español, lo cual ilustra la importancia del problema a resolver.

1.3. Objetivos

Para este trabajo se presentan los siguientes objetivos a alcanzar.

- Desarrollar un clasificador, mediante técnicas de aprendizaje automático y entrenamiento supervisado, capaz de identificar el fenómeno de los sujetos omitidos en textos en español.

- Lograr adaptar un corpus ya existente de gran tamaño para ser utilizado como entrada del clasificador a desarrollar. El mismo deberá marcar no solo aquellos verbos cuyo sujeto se encuentre omitido, sino además cuando el mismo sea explícito, o cuando se trate de construcciones impersonales.
- Sintetizar el proceso desarrollado en un módulo Python de modo que el mismo pueda extraerse y utilizarse en trabajos futuros.
- Utilizar el módulo dentro de una herramienta interactiva que permita la clasificación dinámica de textos de manera simple.

1.4. Estructura del Informe

El resto del informe se estructura de la siguiente manera: En el capítulo 2 se detallan trabajos que describen el estado del arte en referencia a la detección de sujetos omitidos tanto en español como en otros idiomas. También se detallan las herramientas utilizadas durante la realización del proyecto y los corpus utilizados.

En el capítulo 3 se describe el desarrollo de la replicación de uno de los trabajos detallados en el capítulo 2. En el mismo se presentan los pasos seguidos para su realización, así como los resultados obtenidos y su comparación con el trabajo original.

En el capítulo 4 se presenta el desarrollo principal de este trabajo, evaluando sobre los dos corpus de desarrollo utilizados, presentando los resultados obtenidos. Se describen también los cambios y mejoras realizados al modelo. En este capítulo se incluye además la evaluación del modelo desarrollado sobre un nuevo corpus, completamente distinto al utilizado para el desarrollo, y los resultados obtenidos sobre el mismo.

El capítulo 5 describe el análisis de errores realizado sobre el resultado final del trabajo. En éste se describe el proceso general de evaluación, así como un análisis general de los errores cometidos por el clasificador según ciertos atributos particulares.

Finalmente, el capítulo 6 presenta las conclusiones así como también el trabajo futuro propuesto.

Capítulo 2

Marco de trabajo

En este capítulo se exponen conocimientos previos al desarrollo del trabajo. Primero se detalla el estado del arte del problema, luego las herramientas utilizadas para la resolución del mismo, así como los corpus en los que se evalúa y los clasificadores que se utilizan.

2.1. Antecedentes en trabajos similares

Como punto inicial se realiza una investigación sobre la detección de sujetos omitidos. Teniendo en cuenta que este fenómeno no se presenta únicamente en el idioma español sino que existen otros idiomas donde se manifiesta, y anticipándose a una posible escasez de trabajos sobre el tema, dicha búsqueda no se limita únicamente al idioma español.

A continuación se incluye algunos de los trabajos encontrados en nuestra investigación y un análisis de las principales características de cada uno. Comenzaremos analizando el sistema Elliphant para el español que juega un rol importante en el proyecto, siguiendo con un trabajo que intenta realizar la detección de sujetos omitidos para el idioma polaco y finalizando con un predictor de sujetos omitidos desarrollado para un traductor.

2.1.1. Detección de sujeto omitido e impersonal para idioma español

En el trabajo *Elliphant: Improved Automatic Detection of Zero Subjects and Impersonal Constructions in Spanish (2012)* [17], se encontró la aproximación más cercana al objetivo de este proyecto de grado. El mismo distingue entre oraciones donde el sujeto esta explícito o implícito y oraciones impersonales que nunca tendrán sujeto.

Se utiliza un corpus generado específicamente para esta tarea, el cual consiste en dos categorías de textos que fueron anotados de manera manual. Se utilizan textos legales y documentos psiquiátricos. Las dos clases de fuentes utilizadas provienen de dominios altamente especializados, y particularmente en el caso de los textos legales, con una forma sumamente particular de expresarse. Se puede discutir entonces si estas fuentes son las más convenientes para generar una herramienta capaz de reconocer sujetos omitidos o construcciones impersonales en textos más generales.

Para la construcción del clasificador los autores se definen 14 features (atributos) para evaluar:

- PARSER: presencia o ausencia de sujeto en la cláusula, según el parser.
- CLAUSE: tipo de cláusula: principal, relativas comenzando con conjunción compleja, con una conjunción simple o con símbolos de puntuación.
- LEMMA: información léxica extraída del verbo, el lema del verbo infinitivo.
- NUMBER, PERSON: información morfológica del verbo.
- AGREE: concordancia del verbo con el verbo que lo precede en una oración, así como con el verbo principal de la oración, en tiempo, persona, número.
- NH_{PREV}: Candidatos a sujeto de la cláusula representados por número de sintagmas nominales en la cláusula que preceden al verbo.
- NH_{TOT}: Total de candidatos a sujeto de la cláusula representados por número de sintagmas nominales en la cláusula.
- INF: Cantidad de verbos en infinitivo de la cláusula.

- SE: Feature binaria que indica la presencia del pronombre "se" cuando ocurre inmediatamente antes o después de un verbo (o con un máximo de un token entre el verbo y él)
- A: Feature binaria que indica la presencia o ausencia de la preposición *a* en la cláusula. A veces se genera la distinción entre una construcción impersonal y un cláusula pasiva con sujeto omitido por la aparición de la preposición *a*.
- POS_{pre}, POS_{pos}: part-of-speech de ocho tokens (4-grama anterior y 4-grama posterior de la instancia)
- $VERB_{type}$: Clasificado como copulativo, pronominal, transitivo o con uso impersonal. Si caen bajo más de una categoría, se los clasifica con un tipo nuevo correspondiente a la combinación de los tipos del verbo.

Los autores utilizan la plataforma WEKA[5] para realizar la evaluación de su clasificador. Como primer paso evalúan la utilidad de las features definidas intentando determinar si existen features redundantes, así como para identificar las que más información aportan. Estas pruebas llevan a los autores a concluir que no hay features redundantes, ya que la ausencia de cualquiera de ellas resulta en una reducción de accuracy. Sin embargo, también encuentran que por sí solas, las features no aumentan de manera notoria el clasificador, siendo PARSER la más influyente, pero aún así no generando grandes pérdidas de accuracy. De esto concluyen que es necesario combinar las features para mejorar el clasificador.

Evaluando las features en grupos de seis, encuentran que el más efectivo, seleccionado por la técnica WEKA SymetricalUncertAttribute, es el grupo $\{NH_{PREV}$, PARSER, NH_{TOT}, POS_{pos}, PERSON, LEMMA$\}$ con accuracy 83,5 %. De éstas, las features seleccionadas más frecuentemente son PARSER, POS_{pos}, y NH_{TOT}, y se obtiene 83,6 % de accuracy cuando se utilizan únicamente esas tres. Es razonable suponer que alguna de las otras features del grupo genera algún tipo de confusión, ya que reducen la accuracy de las tres mejores features.

Otro análisis de features realizado consiste en distinguir entre *features simples* y *features complejas*, tomando como las simples aquellas que no requieren más información que la provista por los tags generados por el parser, estas features son {PARSER, LEMMA, PERSON, POS_{pos}, POS_{pre}}. El conjunto {CLAUSE, AGREE, NH_{PREV}, NH_{TOT}, $VERB_{type}$} es considerado complejo. La dificultad para evaluar los miembros del último grupo de features se debe a que los autores crearon módulos para obtener y clasificar las cláusulas de su corpus, así como

para identificar y contabilizar las *noun phrases* o sintagmas nominales. Si bien evaluando con las features complejas únicamente se obtienen mayor accuracy que haciendo la misma prueba con las features simples (82,6 % vs 79,9 %), las mismas no logran identificar ninguna instancia de construcciones impersonales.

Considerando la evolución del desempeño del clasificador al aumentar la cantidad de instancias utilizadas, los autores señalan que este alcanza su máximo rendimiento al llegar al 90 % de los datos, aunque se altere el orden de las instancias. Una observación interesante a hacer sobre esta evaluación es la forma de la curva para cada categoría de oración. La precisión para reconocer oraciones con sujetos explícitos es casi constante al agregar más instancias, incluso disminuyendo ligeramente al pasar el 90 % de los datos utilizados. Las oraciones con sujeto omitido por otra parte, aumentan rápidamente su precisión entre el 10 y 20 % de las instancias, y se mantienen constantes hasta llegar al 90 %, donde crece un poco más. Totalmente distinto es el caso de las construcciones impersonales, cuya curva de precisión es oscilante, disminuyendo fuertemente entre 60 y 80 % del corpus, luego de haber crecido rápidamente, antes de recuperarse al llegar al total de los datos. Los autores atribuyen este comportamiento a la falta de ejemplos de esta clase entre los datos.

Se realiza además una evaluación de performance según el género de texto, donde los autores concluyen que los textos legales son más homogéneos y más informativos que los textos de salud. Esta conclusión se basa en los resultados que obtienen al entrenar su clasificador únicamente con un género y evaluarlo en el mismo y en los textos del otro género. En esta prueba observan que el clasificador entrenado únicamente con textos legales tiene mejor precisión al evaluar todas la categorías que aquel entrenado únicamente en textos de salud. Una posible razón dada para esto es la heterogeneidad que tienen los textos de salud, así como la gran cantidad de acrónimos y nombres particulares que utilizan.

2.1.2. Detección de sujeto nulo para polaco

En [7] se detalla un trabajo similar al que se busca realizar, esta vez sobre el idioma polaco, donde se encontró que el problema del sujeto omitido ocurre no solo en el español y lenguas romances, sino que es una característica compartida con las lenguas baltoeslavas, dado que la flexión de los verbos es capaz de proveer la misma información. En este artículo se menciona que este fenómeno

ocurre en aproximadamente el 30 % de los verbos en polaco, 30,42 % en italiano, y 41,17 % en español.

La ausencia de un corpus suficientemente grande es una de las dificultades que se menciona en el artículo para crear un clasificador útil para la detección de sujetos nulos. En el trabajo para el polaco se utilizaron 779 documentos anotados de manera manual, divididos equitativamente para el desarrollo y la evaluación del clasificador. Se menciona que el corpus de desarrollo tiene 10801 verbos, mientras que el de evaluación tiene 11000.

Otro problema mencionado es la falta de un parser de dependencias suficientemente maduro, ya que se utiliza el primero creado para polaco, presentado en 2012[21]. Si se utiliza como única herramienta para la detección de sujetos omitidos reporta peores resultados que utilizando un clasificador baseline.

Para el desarrollo de su clasificador, el autor define cuatro grupos de features a evaluar: Verbo, Tokens vecinos, Largo, y Existencia de candidato a sujeto. Estas mismas evalúan atributos del verbo de la cláusula (como número), información sobre los tokens que rodean al verbo (como POS tags), cantidad de tokens en la oración y en la cláusula que contiene al verbo, y evaluación de construcciones particulares propias del idioma, respectivamente.

Utiliza como base un clasificador que siempre predice que la cláusula tiene sujeto explícito, el cual presenta accuracy de 71,13 % sobre el corpus de desarrollo. Por otra parte, el clasificador desarrollado obtiene resultados de accuracy de [82.49 %, 82.99 %]. Con el clasificador entrenado, evaluando sobre el corpus de evaluación obtiene accuracy de 83,38 %. El recall para los sujetos omitidos es de 67,39 %, mientras que la precision es de 71,97 %.

Un aspecto final mencionado refiere a la curva de aprendizaje del clasificador desarrollado. En la misma se observa que mejora su accuracy al utilizar una mayor cantidad de datos, pero que la misma se aproxima a una asíntota sobre el total del corpus de entrenamiento. Por esto el autor sugiere que el modelo presentado no parece capaz de aumentar de manera significativa su accuracy, aún si existieran más datos para entrenarlo.

2.1.3. Predictor general de sujeto omitido

Otro paper de interés es el *Language independent null subject prediction for statistical machine translation (2015)*[8], que toma un camino completamente distinto para atacar el problema de los sujetos omitidos. Fue realizado por un grupo de autores trabajando para Google Japón. El enfoque de estos autores se centra en la dificultad para realizar traducciones desde los que llaman *null subject languages* o NSLs, como el español, a idiomas *non-NSLs* como el inglés. Las dos razones por las que consideran este un problema importante a resolver son que los modelos de traducción estadísticos no están diseñados para realizar traducciones cuando la frase original y su traducción no se alinean correctamente, y por otra parte, que este no es un fenómeno raro, ni restringido a un solo idioma, como ya se mencionó. Es por esto que buscan generar un predictor de sujeto omitido independiente del idioma. Se mencionan trabajos previos de otros autores que abordan este problema, pero en todos los casos se concentran en un idioma específico. Esto dificulta la escalabilidad de incorporar estos clasificadores como un paso de preprocesamiento en una traducción.

Los autores identifican dos tareas para corregir estas traducciones: detectar si el sujeto fue omitido, y estimar el tipo de sujeto que debería aparecer.

Para realizar la primer tarea, se realiza la traducción a inglés en primer lugar, y se utilizan analizadores disponibles para evaluar la presencia del sujeto. La regla general para decidir si una oración no tiene sujeto es la ausencia de un sujeto para el verbo en una estructura de predicado-argumento. Se manejan de forma especial los casos de las oraciones imperativas, ya que las mismas, por construcción, carecen de un sujeto.

Para la segunda tarea, los autores definen un método para predecir el sujeto ausente. Esto se hace modelando un problema de optimización que busca el pronombre $z \in \{\text{I, you, we, it, they, he/she, imperative}\}$ que maximice una cierta función S de puntuación. Ésta recibe como parámetros z, la oración original, la traducción generada, el árbol de dependencia parseado para la traducción, el verbo sin sujeto y la palabra de la fuente que se alinea con el verbo traducido sin sujeto. El puntaje S se divide en una suma ponderada de tres puntajes y un corrector de parcialidad. Estos son el puntaje sobre el origen, el puntaje sobre la traducción y el puntaje para el modelo de lenguaje. El primero se computa como la probabilidad que el pronombre z puede generarse a partir del verbo sin sujeto en la oración origen. Esta probabilidad se obtiene de un corpus paralelo con alineación automática. El puntaje sobre la traducción se calcula de forma similar,

calculando la probabilidad de que el pronombre z se genere del verbo traducido y el árbol de dependencia. El último se computa midiendo como cambia la puntuación del modelo de lenguaje luego de agregar z como pronombre del verbo y modificando el verbo para que concuerde con el pronombre agregado, en caso de ser necesario.

Como nota interesante, se menciona que el fenómeno de las omisiones de los sujetos en las oraciones ocurre con mayor frecuencia en textos cortos, y de tipo conversacionales. Se recalca además, que idiomas como el español, al ser NSLs de tipo consistente, donde información del sujeto se incluye en la conjugación del verbo, utilizan más fuertemente información de la fuente para evaluar si se esta omitiendo el sujeto de la oración.

En cuanto a resultados, los autores logran mejorar la calidad de las traducciones, en mayor o menor medida, de 18 NSLs a inglés mediante este método. Cabe notar que la resolución de los casos donde la traducción agrega un pronombre equivocado para reemplazar al sujeto ausente para español empeora las traducciones obtenidas de manera notoria.

2.2. Conceptos gramaticales

Todo sujeto en una oración está relacionado con un verbo de la misma, pero existen casos donde en una sola oración se encuentran más de un verbo finito con sujeto, por ejemplo en la oración *El perro ladra y el gato maúlla*. Si se tratara la oración entera, se estaría perdiendo información, ya que se estaría buscando un único sujeto mientras que la oración planteada presenta más de uno.

En el contexto de este trabajo se considera que una **cláusula** es un segmento de texto, subconjunto contiguo de la oración, el cuál contiene un verbo finito principal que puede o no tener un sujeto asociado, ya sea explícito u omitido, o una construcción impersonal. Entonces para la oración del ejemplo *El perro ladra y el gato maúlla* , se pretende extraer las cláusulas *El perro ladra*, *el gato maúlla*.

Por otra parte, también se considera el caso presentado por la oración *Juan come manzanas rojas, pero no come manzanas verdes*. Las cláusulas esperadas de la misma son *Juan come manzanas rojas* y *No come manzanas verdes*, donde la primera tiene como sujeto explícito a *Juan*, mientras que en la segunda se observa un ejemplo de un sujeto omitido.

2.3. Herramientas

En esta sección se listan los principales recursos utilizados para la realización de este trabajo. Incluye tanto las herramientas de software utilizadas, como los corpus sobre los que se trabaja, tanto para el desarrollo del modelo, como para su evaluación.

2.3.1. NLTK

El *Natural Language Toolkit* [2], o NLTK, es un conjunto de módulos Python enfocados al procesamiento del lenguaje natural. Como tal, ofrece una gran cantidad de herramientas para esta tarea. Para este trabajo se utilizaron principalmente los módulos de clasificadores. Particularmente se utilizaron dos, `MaxEntClassifier` y `NaiveBayesClassifier`. Esta librería fue de gran utilidad en los comienzos de este proyecto, sin embargo, al avanzar con el mismo y buscando mejorar los resultados obtenidos, fue necesario cambiar a una herramienta con más recursos. Esta migración fue posible principalmente y se realizó de manera sencilla, ya que no se utilizaron herramientas de la librería NLTK para realizar el procesamiento de los textos, sino que otras herramientas complementarias realizaron las tareas tokenización y etiquetado de *Part-of-Speech* (PoS), por lo que la migración fue exclusivamente de los clasificadores.

2.3.2. Scikit-learn

Scikit-learn[16] es una librería Python enfocada en herramientas de aprendizaje automático. Durante el desarrollo de este trabajo se utilizaron varios de los componentes que la misma ofrece, particularmente en reemplazo de NLTK. Se aprovechó la mayor variedad de clasificadores de distintas familias disponibles, así como varias de las medidas implementadas en su módulo de métricas.

2.3.3. Clatex

El sistema Clatex[20] permite la identificación y extracción de cláusulas en textos en español. Este trabajo se realiza a partir de las proposiciones encon-

tradas en las oraciones, así como la jerarquía que existe entre ellas. Utiliza una combinación de un etiquetador probabilístico, así como varios conjuntos de reglas formales enfocados en distintos aspectos del problema a analizar. Este sistema, implementado en lenguaje Prolog[1], se incorpora al presente proyecto para realizar la extracción de cláusulas que serán clasificadas.

2.3.4. FreeLing

La herramienta FreeLing[15] [3] [9] consiste de una librería escrita en el lenguaje C++[2], pero que cuenta con una interfaz Python, dedicada al análisis del lenguaje, ofreciendo herramientas para la realización de sus principales tareas. Por ejemplo tokenización, análisis morfológico, predicción probabilística de categorías de palabras, manejo de clíticos y contracciones, etc. FreeLing es una de las herramientas más importantes utilizadas para este trabajo ya que se utilizó para todos los etiquetados de categorías gramaticales, o *Part-of-Speech (PoS) tags*, análisis de dependencias en las cláusulas, lematización, entre otras tareas, necesarias para el cálculo de features y procesamiento de los corpus.

2.3.5. NumPy

La librería NumPy[14] brinda herramientas para el manejo de estructuras complejas de datos, así como la generación de números aleatorios. Estas fueron las principales funcionalidades que se utilizaron en este trabajo. La generación de números aleatorios se utiliza para la división final de los corpus de desarrollo y evaluación. También fue necesario manejar los vectores obtenidos al utilizar Word2Vec, y facilitar la organización de los datos necesarios para ajustar el codificador de features mencionado en la sección 2.3.2. Se utilizaron además herramientas de manejo de memoria disponibles en esta librería, dado que el tamaño de los datos utilizados varía en tamaño según el corpus y las features aplicadas, sobrepasando las capacidades de memoria disponibles en los ambientes de desarrollo utilizados. Con NumPy, fue posible trabajar con las estructuras de datos creadas sobre la totalidad de los datos disponibles.

[1] http://www.swi-prolog.org/ - Último acceso 2018/11/25
[2] http://www.cplusplus.com/doc/tutorial/ - Último acceso 2018/11/25

2.4. Corpus

En esta sección se describen los corpus utilizados para el desarrollo y evaluación del trabajo realizado en el transcurso de este proyecto.

2.4.1. AnCora

El corpus AnCora[19], el cual es el corpus objetivo del proyecto, consta de una colección de 1635 documentos formados principalmente por textos periodísticos, los cuales suman en conjunto un total de 17375 oraciones. Dichos documentos se encuentran en formato XML y están anotados tanto morfológica, sintáctica y semánticamente. Son de mayor interés para el proyecto las anotaciones que tiene sobre sujeto, tanto explícito como implícito.

Al analizar el corpus se observa que cada oración del documento se encuentra entre los tags "sentence" y que sus nodos hijos pueden ser de diferentes tipos de constituyentes dependiendo de las palabras que estos contengan. Por ejemplo una oración puede tener nodos hijos que representen un sintagma adjetival, un sintagma nominal, un grupo verbal, entre otros.

Por otro lado se observa que cuando se encuentra el atributo "func" y es igual a "suj" se está ante la presencia de un sujeto. En estos casos el nodo puede estar determinando tanto un sujeto explícito como uno implícito, para diferenciarlos AnCora le agrega a este último caso el atributo "elliptic" al nodo.

Observando distintos ejemplos se encuentra que además de existir marcas para los sujetos explícitos e implícitos, existe una marca para los verbos impersonales mediante el atributo "func" con valor "impers".

Finalmente, se decide transformar los documentos que componen al corpus AnCora, los cuales están escritos en formato XML, a formato JSON. Este cambio se decide dado que el formato original no es particularmente cómodo para recorrer o procesar. El formato JSON presenta además otras ventajas, no solo en las bibliotecas que existen para poder manipularlo, sino que también a simple vista es notoriamente más sencillo de leer.

A continuación se muestran tres fracciones simplificadas del corpus de AnCora, se omiten tags que no son utilizados para este trabajo con el fin de aumentar

la legibilidad.

El primer ejemplo se trata de una oración con sujeto explícito, pudiendo notarlo al encontrar un nodo con el atributo “func” en “suj”.

```
<sentence>
  <sn func='suj'>
    <grup.nom num='s' gen='m'>
      <n wd='Hiddink' ne='person' pos='np00000' lem='Hiddink'/>
    </grup.nom>
  </sn>
  <grup.verb>
    <v wd='sigue' mood='indicative' lem='seguir' pos='vmip3s0'
    person='3' num='s' tense='present' />
  </grup.verb>
  <sadv func='cc'>
    <grup.adv>
      <r wd='atentamente' lem='atentamente' pos='rg'/>
    </grup.adv>
  </sadv>
  <sn func='cd'>
    <spec>
      <d wd='el' pos='da0ms0' lem='el' num='s' gen='m'/>
    </spec>
    <grup.nom>
      <n wd='entrenamiento' lem='entrenamiento' pos='ncms000'
      num='s' gen='m'/>
      <sp func='cn'>
        <prep>
          <s wd='de' lem='de' pos='sps00'/>
        </prep>
        <sn>
          <spec>
            <d wd='sus' lem='su' person='3' num='p' pos='dp3cp0'
            gen='c'/>
          </spec>
          <grup.nom num='p' gen='m'>
            <n wd='jugadores' pos='ncmp000' lem='jugador' num='p'
            gen='m'/>
          </grup.nom>
        </sn>
      </sp>
```

```
    </grup.nom>
  </sn>
  <f wd='.' lem='.' pos='fp'/>
</sentence>
```

En el siguiente ejemplo se encuentra, al igual que en el anterior, un nodo con el atributo "func" en "suj", pero además contiene el atributo "elliptic", lo que indica que se trata de una oración que contiene un sujeto omitido.

```
<sentence>
  <S>
    <f wd='"' lem='"' pos='fe' />
    <sadv func='cc'>
      <grup.adv>
        <r wd='Por_ahora' lem='por_ahora' pos='rg' />
      </grup.adv>
      <f wd=',' lem=',' pos='fc'/>
    </sadv>
    <sn elliptic='yes' func='suj' />
    <grup.verb>
      <v wd='prefiero' mood='indicative' lem='preferir'
      pos='vmip1s0' person='1' num='s' tense='present' />
    </grup.verb>
    <neg func='mod'>
      <r wd='no' lem='no' pos='rn'/>
    </neg>
    <S impersonal='yes' func='cd'>
      <infinitiu>
        <v wd='hablar' mood='infinitive' pos='vmn0000'
        lem='hablar' />
      </infinitiu>
      <sp func='creg'>
        <prep>
          <s wd='de' lem='de' pos='sps00'/>
        </prep>
        <sn>
          <spec>
            <d wd='ese' pos='dd0ms0' lem='ese' num='s'
            gen='m'/>
          </spec>
          <grup.nom>
            <n wd='asunto' pos='ncms000' lem='asunto'
```

```
            num='s' gen='m'/>
          </grup.nom>
        </sn>
      </sp>
    </S>
    <f wd='"' lem='"' pos='fe'/>
    <f wd=',' lem=',' pos='fc'/>
  </S>
  <grup.verb>
    <v wd='concluyó' mood='indicative' lem='concluir'
    pos='vmis3s0'
    person='3' num='s' tense='past' />
  </grup.verb>
  <sn func='suj'>
    <grup.nom>
      <n wd='Toni' ne='person' pos='np0000p' lem='Toni'/>
    </grup.nom>
  </sn>
  <f wd='.' lem='.' pos='fp'/>
</sentence>
```

Por último se presenta un fragmento de una oración impersonal, donde se aprecia la marca que lo distingue, un nodo con el atributo "func" en "impers".

```
<S>
  <morfema.verbal func='impers'>
    <p wd='Se' pos='p0000000' gen='c' lem='se' num='c' />
  </morfema.verbal>
  <grup.verb>
    <v wd='dice' mood='indicative' lem='decir' pos='vmip3s0'
    person='3' num='s' tense='present' />
  </grup.verb>
  <S>
    <conj>
      <c wd='que' lem='que' pos='cs'/>
    </conj>
    <sp func='cc'>
      <prep>
        <s wd='en' lem='en' pos='sps00'/>
      </prep>
      <sn>
        <spec>
```

```
          <d wd='el' pos='da0ms0' lem='el' num='s' gen='m'/>
        </spec>
        <grup.nom>
          <n wd='fútbol' pos='ncms000' lem='fútbol' num='s'
          gen='m'/>
        </grup.nom>
      </sn>
    </sp>
    <neg func='mod'>
      <r wd='no' lem='no' pos='rn'/>
    </neg>
    <grup.verb>
      <v wd='hay' mood='indicative' pos='vaip3s0' lem='haber'
      person='3' num='s' tense='present'/>
    </grup.verb>
    <sn func='cd'>
      <grup.nom >
        <n wd='romanticismo' pos='ncms000' lem='romanticismo'
        num='s' gen='m'/>
      </grup.nom>
    </sn>
  </S>
  <f wd=',' lem=',' pos='fc'/>
</S>
```

De XML a JSON

Para transformar los documentos se deben tener en cuenta algunos puntos, como por ejemplo mantener la estructura de la colección y no perder el orden de las oraciones en el documento ni de las palabras en una oración. Se deben guardar los datos que nos brinda AnCora sobre los sujetos en la oración marcando el verbo al cual están relacionados, así como también los atributos de las palabras encontradas, ya que estos serán útiles en un futuro.

Se crea una función que tiene como entrada un documento XML de AnCora y retorna un documento JSON con el mismo nombre, dicha función analiza el documento procesando cada oración por separado y guardando el resultado en una colección, la cual luego será el documento JSON a retornar, de esta manera se logra mantener el orden de las oración dentro del documento.

El proceso de cada oración se realiza mediante una función recursiva la cual retorna en una lista ordenada donde cada elemento de la misma es un objeto que representa una palabra de la oración. Además cada objeto contiene todos los atributos que ya contenía en el documento XML y para el caso en el que la palabra sea un verbo con un sujeto asociado, se le agrega la información del sujeto encontrado, ya sea explícito, implícito o impersonal. A continuación se presenta un pseudo-código de la función.

```
function procesar_oracion(nodo, resultado)
    buscando_sujeto = FALSE
    buscando_verbo = FALSE
    posicion_verbo = -1
    sujeto_encontrado = {}

    // PASO BASE
    si el nodo contiene el atributo "wd"
        // el nodo es una palabra
        se crea un objeto con todos los atributos de el nodo y
        se almacena al final de la lista resultados

    // PASO RECURSIVO
    para todo x tal que x es hijo de nodo
        res = procesar_oracion(x, [])
        se concatena res a la lista resultados

        si el nodo es un sujeto
            se almacenan todos los atributos de el nodo sujeto
            en la variable sujeto_encontrado

            si buscando_sujeto
                // se habia encontrado un verbo pero no el sujeto
                // asociado
                agrego al objeto resultado[posicion_verbo] la key
                "suj" con valor sujeto_encontrado
                buscando_sujeto = FALSE
            sino
                // se encontro un sujeto pero no su verbo asociado
                buscando_verbo = TRUE

        si el nodo es un grupo verbal
            se almacena en posicion_verbo la posicion en la lista
            resultados de el verbo encontrado

            si buscando_verbo
                // se habia encontrado un sujeto pero no el verbo
```

```
                // asociado
                agrego al objeto resultado[posicion_verbo] la key
                "suj" con valor sujeto_encontrado
                buscando_verbo = FALSE
            sino
                // se encontro un verbo pero ningun sujeto
                buscando_sujeto = TRUE
    retorna resultados
```

Procesamiento para Clatex y obtención de cláusulas

Se utiliza Clatex para obtener las cláusulas de AnCora, la herramienta espera como entrada una palabra por línea, junto a su lema, part-of-speech tag, y probabilidad estimada del tag seleccionado. Toda esta información se encuentra en los documentos originales del corpus y convenientemente también se encuentran en los nuevos documentos JSON, haciendo que la tarea de transformar los documentos en entradas validas para Clatex sea un trabajo sencillo.

2.4.2. Elliphant

El corpus Elliphant forma parte del trabajo analizado en la sección 2.1.1. El mismo consiste de 5212 oraciones en 17 documentos, pertenecientes a dos categorías: textos legales y textos del área de la salud.

Este corpus tiene formato XML, donde cada oración se encuentra contenida en tags `sentence`, y cada uno de sus tokens se encuentra contenido en tags `token`. Cada tag `token` contiene la información de cada una de las palabras que lo componen (texto y lema), así como información morfológica del mismo (por ejemplo, su PoS-tag). En el caso de los verbos conjugados se incluye la información principal utilizada, la clasificación relativa a la presencia de un sujeto.

Dado que para la realización de el presente trabajo no es necesaria la mayor parte de la información incluida, se decide simplificar el contenido del corpus. En primer lugar se realiza un pre-procesamiento donde se extrae el texto plano de cada una de las oraciones que lo componen. El siguiente paso implica la obtención de las cláusulas de cada una de las oraciones. Para esto se aplica el

mismo proceso utilizando FreeLing y Clatex mencionado en la sección anterior. Debido a la necesidad de trabajar sobre el texto extraído de las tags del corpus, fue necesario volver a relacionar los verbos principales de cada cláusula con su respectiva clasificación de clase de sujeto asignada en el corpus. Este proceso requirió un par de iteraciones debido a algunas modificaciones que sufrieron los textos a ser procesados y que dificultó la tarea de emparejamiento. El resultado final es un corpus que no es más que un listado de tuplas que contiene el texto llano de la cláusula y la clasificación de su verbo. En iteraciones posteriores, se agrega también a la tupla el verbo principal de la cláusula según se identifica mediante FreeLing. Las razones de este cambio se detallaran en la sección 4.3.4.

El siguiente ejemplo muestra una representación simplificada de una oración del corpus Elliphant. Al igual que en los ejemplos para el corpus de AnCora, se omiten tags que no son utilizados para este trabajo. Dentro de los nodos `token` correspondientes a verbos, se incluye la clasificación de sujeto, dentro del nodo `tags`, con la etiqueta `subject`. En el ejemplo presentado a continuación se encuentran dos verbos etiquetados, uno de tipo *ZERO* y otro de tipo *SUBJECT*.

```
<sentence id="w1398">
   <token id="w1399">
      <text>Continúan</text>
      <lemma>continuar</lemma>
      <depend head="w1398">main</depend>
      <tags>
         <syntax>@MAIN</syntax>
         <morpho>V IND PRES PL P3</morpho>
         <subject>ZERO</subject>
      </tags>
   </token>
   <token id="w1400"> <text>contando</text> </token>
   <token id="w1401"> <text>que</text> </token>
   <token id="w1402"> <text>con</text> </token>
   <token id="w1403"> <text>el</text> </token>
   <token id="w1404"> <text>paso</text> </token>
   <token id="w1405"> <text>de</text> </token>
   <token id="w1406"> <text>los</text> </token>
   <token id="w1407"> <text>días</text> </token>
   <token id="w1408"> <text>los</text> </token>
   <token id="w1409"> <text>síntomas</text> </token>
   <token id="w1410"> <text>se</text> </token>
   <token id="w1411">
      <text>agudizaron</text>
```

```
      <lemma>agudizar</lemma>
      <depend head="w1400">obj</depend>
      <tags>
         <syntax>@MAIN</syntax>
         <morpho>V IND PRET PL P3</morpho>
         <subject>SUBJECT</subject>
      </tags>
   </token>
   <token id="w1412"> <text>y</text> </token>
   <token id="w1413"> <text>su</text> </token>
   <token id="w1414"> <text>estado</text> </token>
   <token id="w1415"> <text>de</text> </token>
   <token id="w1416"> <text>ánimo</text> </token>
   <token id="w1417"> <text>también</text> </token>
   <token id="w1418"> <text>.</text> </token>
</sentence>
```

2.4.3. Corin

El corpus Corin[4] está formado por textos literarios y de prensa de origen nacional, escritos durante el periodo 1996-2000. El mismo cuenta con 89 textos en total.

Preparación del corpus

La versión utilizada de este corpus no se encuentra anotada, por lo que el primer paso fue identificar las cláusulas de cada uno de los textos, así como su correspondiente verbo principal, para su posterior clasificación.

Se utilizaron las mismas clases con las que clasifica Elliphant: *SUBJECT* para sujeto explícito, *ZERO* para sujeto implícito, *IMPERSONAL* para oraciones impersonales y *ERROR* ya que en las cláusulas identificadas en ocasiones se marcan verbos de manera incorrecta.

A continuación se presentan ejemplos de las clases con las que se clasifica:

Ejemplo de sujeto explícito, marcado con SUBJECT.

```
<clausula>
    Zanocchi
    <verbo subject="SUBJECT">estaba</verbo>
    presente en la reunión
</clausula>
```

Ejemplo de sujeto omitido, marcado con ZERO.

```
<clausula>
    cómo
    <verbo subject="ZERO">van</verbo>
    a tratar el tema de el Comité Central Israelita
</clausula>
```

Ejemplo de oración impersonal, marcado con IMPERSONAL.

```
<clausula>
    <verbo subject="IMPERSONAL">Parecería</verbo>
    que los hombres no soportan los enigmas
</clausula>
```

Por último, se presenta un ejemplo de una cláusula de la clase ERROR, la misma corresponde a un error de segmentación durante el procesamiento de los textos con Clatex. En la figura 2.1 se incluye el contexto previo de la cláusula, mientras que en la figura 2.2 se presenta la salida retornada por Clatex. Este error se genera al considerar como cláusula el texto "insignificante en El Observador, [...] destaque", donde se toma como verbo principal *destaque*.

```
<clausula>
    insignificante en El Observador ,
    que dio cuenta de el discurso con
    <verbo subject="ERROR">destaque</verbo>
</clausula>
```

```
Con muy escaso tiempo, el ``operativo silencio'' obtuvo
diversos resultados: nulo en el diario El País, donde la
noticia fue tema de tapa; insignificante en El Observador,
que dio cuenta del discurso con destaque
```

Figura 2.1: Extracto de oración de Corin

```
[prop gf22/ insignificante en El Observador , [prop prl4/
que dio cuenta de el discurso con /prop prl4] destaque
/prop gf22]
```

Figura 2.2: Salida parcial de Clatex para texto de la figura 2.1

En total, se clasificaron 8 de los documentos de este corpus, obteniendo casi 600 cláusulas. En el cuadro 2.1 se muestra la distribución de clases en el mismo.

Tipo de Verbo	
SUBJECT	414 (69,81 %)
ZERO	119 (20,00 %)
IMPERSONAL	74 (12,48 %)
ERROR	13 (2,19 %)
Total	593

Cuadro 2.1: Distribución de tipos en corpus Corin

2.5. Clasificadores

Los clasificadores son implementaciones de algoritmos cuya tarea es la de asignar clases a partir de una entrada. Para esto, se utiliza un conjunto de features evaluadas sobre la entrada recibida por el clasificador. Durante un entrenamiento supervisado, estos vectores de features estarán acompañados del valor de clasificación deseado para la instancia que se está evaluando. A partir de estos pares de datos, los clasificadores buscarán generalizar lo aprendido a datos nuevos y no vistos. A continuación se describen las características principales la metodología general de evaluación, así como de las familias de clasificadores que se utilizan en este trabajo.

2.5.1. Vectorización

Generalmente, los clasificadores utilizados para aprendizaje automático requieren vectores numéricos como entrada. Sin embargo, no todos los datos representados por las features definidas se mapean naturalmente a una distribución numérica, particularmente las features con valores categóricos, como por

ejemplo las PoS tags de una palabra. Entonces, es necesario definir una estrategia de vectorización que ayude a la tarea del clasificador.

Label Encoding

La primera implementación de la vectorización se realizó de manera manual, adoptando la estrategia conocida como Label Encoding. En esta estrategia, simplemente se realiza un mapeo en el cual a cada valor de una feature se le asigna un valor numérico. Este mapeo debe ser consistente y reversible, donde es posible recuperar el valor categórico de una feature a partir de su codificación numérica. Se aplicó esta implementación a todas las features utilizadas en primera instancia, generando primero un listado de todos los valores posibles de cada una, según los valores encontrados en el corpus. De esta forma, para transformar una feature a un valor numérico basta con encontrar el índice de su ocurrencia en el listado correspondiente. Esta estrategia presenta un problema claro: se introduce una relación de orden a datos categóricos que no la tienen, pero que el clasificador utiliza para aprender. Por ejemplo, si se toma la feature *PERSON* que codifica la persona en la que está conjugado un verbo, se tienen tres opciones, *primera*, *segunda*, y *tercera*, y se le asignarían tres identificadores numéricos (por ejemplo 0, 1 y 2). El valor de esta feature no tiene relación interna, pero el clasificador entenderá que $primera < segunda < tercera$. Es por esto que esta no es la mejor estrategia para estos tipos de datos.

One Hot Encoding

Para resolver el problema planteado se utiliza una estrategia distinta, One Hot Encoding. La codificación One Hot consiste en transformar las todas features definidas en features binarias, es decir que los valores posibles de cada feature se transforman en una feature binaria cada uno. Esto tiene como ventaja la eliminación de la relación natural que ocurre al asignar a cada categoría un valor numérico de manera incremental. Label Encoding participa de este proceso, siendo el primer paso a realizar transformando los valores de las categorías de cada feature a valores numéricos (en esta etapa se utiliza la implementación del LabelEncoder incluido en la librería Scikit-learn). Luego, cada feature definida para el clasificador pasa a ser un vector de coordenadas binarias, cuyo largo se corresponde al total de las categorías observadas para esa feature, y donde este vector siempre tendrá no más de un bit encendido. En el caso del codificador One

Hot, también se utilizó la implementación disponible en la librería Scikit-learn.

Si consideramos nuevamente el ejemplo de la feature *PERSON*, el vector de la feature resultante tiene dimensión 3. Donde el vector `100` representa a la primera persona, `010` representa a la segunda persona, y `001` representa a la tercera persona.

El encoder se ajusta utilizando únicamente las features evaluadas del corpus de entrenamiento de AnCora, y se le indica que ignore los valores desconocidos al momento de transformar un vector. De este modo, si una feature evalúa a un valor desconocido al momento de codificar las cláusulas de evaluación, el encoder retornará ceros en todos los valores correspondientes a esa feature en lugar de generar un error.

Esta parece ser una mejor estrategia para la vectorización de los datos categóricos utilizados ya que elimina las dependencias artificiales en las features. La vectorización One Hot se utiliza entonces para la vectorización de todas las features categóricas implementadas en el modelo desarrollado.

Word Embeddings

Existe una desventaja en la estrategia One Hot, ya que con la misma se pierde la capacidad de reconocer relaciones en algunas features, como por ejemplo la feature de representanción del verbo principal de una cláusula, siendo que existen relaciones reales de similitud entre algunas palabras. En este caso sería deseable mantener aspectos de esta relación, como por ejemplo la cercanía semántica entre palabras, dado que los mismos podrían ayudar a la tarea de clasificación.

Word Embeddings es el término que engloba a un grupo de modelos vectoriales de representación de palabras. Con estos modelos es posible obtener un espacio vectorial de representación de palabras generado a partir de un corpus de texto. Bajo este modelo, existe una correlación entre la distancia de las palabras representadas y su significado o contexto. La inclusión de este tipo de técnicas suele traer mejorías en los resultados al evaluar, por esto, se utilizó esta herramienta para representar al verbo principal de las cláusulas, agregando sus coordenadas al vector de features generado con la codificación One Hot. La implementación de Word Embeddings utilizada en este trabajo es Word2vec[13].

2.5.2. Evaluación

Como se menciona en la descripción de la sección, los clasificadores buscan generalizar lo aprendido a partir de los datos de entrenamiento para lograr clasificar instancias nuevas, sobre las cuales el clasificador será evaluado. Es fundamental realizar la evaluación sobre instancias distintas a las de entrenamiento, sino no se lograrán buenos resultados en la evaluación, ya que solo necesita repetir la información con la que fue entrenado y probablemente tenga dificultades con datos nuevos y distintos, generando una visión falsa del desempeño del clasificador entrenado, esto se denomina sobreajuste. Existen varias estrategias para realizar la evaluación de un clasificador, durante el desarrollo de este proyecto se utilizan dos tipos de evaluación, dependiendo de la etapa del mismo, las cuales se describen a continuación.

Al comienzo, y durante la realización de experimentos exploratorios, así como durante los primeros ajustes realizados al modelo se utiliza el modelo de desarrollo Cross-validation de k iteraciones. Éste método consiste en dividir el corpus en k particiones, donde el clasificador se entrenará con $k - 1$ de las particiones, y se evaluará en la partición restante, calculando métricas sobre la misma. Este proceso se repite k veces, hasta que se hayan probado todas las permutaciones de entrenamiento/evaluación posibles. Tiene como ventaja el bajo nivel de pérdida de datos, dado que solo se reserva una k-ésima parte del corpus, permitiendo entrenar sobre el resto, además de reducir el sesgo del clasificador, dado que entrena sobre conjuntos distintos. Sin embargo, esta técnica aumenta la varianza, pues estos conjuntos pueden ser muy distintos entre sí. Aún así, este método permitió obtener un panorama general del desempeño de los clasificadores entrenados durante las primeras etapas de trabajo. En particular para este trabajo se utilizó $k = 5$.

Posteriormente, con un conjunto de features estables, y resultados aceptables, se modifica el modelo de evaluación, abandonando Cross-validation en favor de una división training-testing. En esta, se utiliza el 80 % del corpus para el entrenamiento de los clasificadores, y se evalúa en el restante 20 %. Con ésta metodología, se reduce la varianza, dado que se está utilizando un conjunto de datos mayor. De esta forma, se ofrece además una visión más clara de la actuación del modelo generado[6].

2.5.3. Medidas

Para cada una de las clases que se busca clasificar, se consideran los siguientes conceptos.

- *True positive (TP)* Cantidad de instancias de la clase A que se clasifican correctamente como clase A.
- *True negative (TN)* Cantidad de instancias de una clase distinta a A que se clasificaron correctamente como pertenecientes a otra clase.
- *False positive (FP)* Cantidad de instancias que se clasifican incorrectamente como pertenecientes a la clase A.
- *False negative (FN)* Cantidad de instancias pertenecientes a la clase A, que se clasifican incorrectamente como pertenecientes a otra clase.

Dadas las definiciones anteriores, se utilizaran las siguientes medidas a la hora de evaluar los clasificadores entrenados[6].

Accuracy Definida como $\frac{TP+TN}{TP+TN+FP+FN}$ representa la cantidad de instancias de la clase A que se clasificaron correctamente, sobre el total de las instancias.

Precision Definida como $\frac{TP}{TP+FP}$ identifica la cantidad de instancias de la clase A que se clasificaron correctamente como pertenecientes a la clase A, sobre el total de instancias que fueron clasificadas como pertenecientes a la clase A.

Recall Definida como $\frac{TP}{TP+FN}$ representa la cantidad de instancias de la clase A que se clasificaron correctamente como pertenecientes a la clase A, sobre el total de instancias pertenecientes a la clase A.

F-Measure Definida como $2\frac{precision*accuracy}{precision+accuracy}$ es el promedio armónico entre la precision y el recall.

Matriz de confusión Matriz cuadrada que ilustra los errores cometidos al momento de clasificar, marcando los errores entre clases cometidos. Cada fila de la matriz representa las instancias pertenecientes a una clase, mientras que las columnas representan las instancias que el clasificador asignó a cada clase. Entonces, dada la matriz C, cada entrada C_ij representa la cantidad de instancias pertenecientes a la clase i, pero que el clasificador determinó que pertenece a la clase j[3].

2.5.4. Baseline

Cuando se entrena un nuevo clasificador, muchas veces no se tiene otros resultados con los que compararlo, es en estos casos donde surge el cálculo del baseline. Se utiliza a modo de contextualizar los resultados obtenidos con los distintos clasificadores. En este proyecto se utilizó una herramienta de Scikit-learn llamada DummyClassifier. El mismo clasifica las cláusulas mediante una estrategia definida. Para este trabajo se utiliza la estrategia "most frequent", la cual evalúa a todas las cláusulas con el tag más frecuente encontrado en el corpus. Para todos los corpus utilizados, ese tag es "subject", para sujetos explícitos.

2.5.5. SVM (*Support Vector Machines*)

Esta familia de clasificadores genera un mapeo de los los datos de entrada en un plano n-dimensional, siendo n la cantidad de features definidas. Entonces, cada una de las entradas clasificadas que se ingresan al clasificador serán representadas por un punto en el plano. Los clasificadores SVM buscarán el hiperplano que mejor divida los puntos de una clase de los puntos de la otra. Idealmente, este plano debería además contar con la mayor distancia posible entre elementos de cualquiera de las clases, su *margen*, para minimizar los errores de clasificación.

Se utilizaron dos clasificadores de esta familia, SVC y LinearSVC, ambos implementados por la librería Scikit-learn, las cuales difieren principalmente en la función kernel que utilizan. Mientras que LinearSVC implementa siempre una funcion kernel lineal, SVC admite elegir la misma. Para este trabajo, la función

[3]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html - Último acceso 2018/11/25

utilizada es RBF (*Radial basis function kernel*).

Si bien los clasificadores SVM se definen para la clasificación entre dos clases, es posible extenderlos a un número mayor utilizando las estrategias *one-vs-one* o *one-vs-rest*. Dadas k clases posibles, en el caso *one-vs-one*, se generan $\frac{k*(k-1)}{2}$ clasificadores, entrenando únicamente con dos clases para cada uno. Para decidir la clase de una entrada a clasificar, se utilizan los resultados de cada uno de los clasificadores entrenados, implementando una función de decisión. Esta es la estrategia utilizada por el clasificador SVC. Por otro lado, LinearSVC utiliza *one-vs-rest*, donde se entrenan k clasificadores, donde cada clase se evalúa contra el conjunto de las restantes clases.

2.5.6. Naïve Bayes

Los clasificadores de la familia Naïve Bayes utilizan el teorema de Bayes, donde es posible calcular la probabilidad de un evento a partir de las probabilidades de eventos relacionados. Se los considera *naive* ya que se asume que existe independencia entre las features utilizadas. Existen varias implementaciones de estos clasificadores, que difieren en la distribución de probabilidad utilizada para calcular $P(x_i|y)$ (la probabilidad de que ocurra la feature x_i dado que la instancia es de clase y. El clasificador NaiveBayes implementado por la librería NLTK utiliza distribución Multinomial, mientras que los utilizados de la librería Scikit-learn utilizan distribución Gaussiana y de Bernoulli.

2.5.7. Lineales

Se utilizan dos clasificadores de esta familia, LogisticRegression (en Scikit-learn) o Maximum-Entropy (en NLTK) y *Stochastic Gradient Descent* (SGDClassifier). En esta familia de clasificadores, a diferencia de Naïve Bayes, no se asume ni se requiere que exista independencia entre las features. LogisticRegression genera un clasificador probabilístico, donde se genera la probabilidad de que una cierta entrada pertenezca a una clase.

Estos clasificadores son comparables a la familia SVM, dado que ambas buscan generar una división geométrica entre las distintas clases a clasificar. Mientras SVM maximiza el margen que separa a ambas clases, LogisticRegres-

sion optimiza la función de probabilidad.

La versión de SGDClassifier utilizada se aproxima a una implementación de SVM, comparable con el clasificador LinearSVC.

2.5.8. Ensamble

Los métodos *ensamble* combinan los resultados de varios estimadores para aumentar la robustez del la predicción realizada. Se utilizan dos clasificadores de este tipo: RandomForestClassifier y VotingClassifier.

En un clasificador RandomForest, el concepto principal es el de árbol de decisión. El mismo implica la división de el total de instancias de entrenamiento, buscando con cada división generar nodos más homogéneos internamente, aumentando la heterogeneidad entre nodos distintos. Estas decisiones se realizan en base a las features definidas, y continuarán hasta alcanzar una condición de parada, que puede ser la profundidad del árbol generado, o la homogeneidad de los nodos alcanzada, entre otras. Este clasificador genera varios de estos árboles de decisión, iniciándolos con subconjuntos de los datos de entrada elegidos de manera aleatoria así como un subconjunto de las features, elegido también aleatoriamente. La decisión del clasificador entrenado se genera promediando la predicción de cada uno de los árboles creados.

La versión de VotingClassifier utilizada en este trabajo implementa la estrategia *hard-vote*, donde todos los clasificadores utilizados generan su predicción, y se realiza una votación democrática, donde la clase con la mayor cantidad de votos gana. La idea de esta técnica se basa en balancear las debilidades de clasificadores con buenos resultados para obtener resultados aún mejores[4]

[4]https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier - Último acceso: 2018/11/13

Capítulo 3

Experimentos Previos

Con el fin de acercarse a una solución del problema, se decide replicar la metodología y los resultados expuestos por el proyecto Elliphant, mencionado en la sección 2.1.1, el cual tenía como objetivo la clasificación del sujeto en oraciones en español.

La razón de esto es principalmente aprovechar los avances de un trabajo serio sobre el mismo problema encarado en este proyecto. Para esto, es necesario reproducirlo lo más fielmente posible, para poder, en primer lugar, validar sus resultados. Es así que, logrando una aproximación a la metodología utilizada, y a los resultados obtenidos, es posible generar una linea base para medir la calidad de las mejoras introducidas.

Este trabajo es importante también para medir qué tan generales son sus resultados. Se mencionó en la sección 2.1.1 que se utilizan textos altamente especializados, por lo que es fundamental validar que el enfoque no está sobre-ajustado a estos, y que los mismos pueden extrapolarse a textos más generales. También se utiliza esta oportunidad para introducir estrategias nuevas, como la utilización de Word2vec, y observar su comportamiento en conjunto con las features base.

Finalmente, como se detalla en secciones siguientes, existen inconvenientes para realizar esta replicación, dado que no se cuenta con ciertas herramientas creadas específicamente para el trabajo. Se presenta una descripción detallada de las medidas tomadas para sortear esta dificultad, así como los resultados obtenidos, y su comparación con los resultados originales.

3.1. Replicación de Elliphant

Como primera tarea se procedió a la implementación de las features descritas en la sección 2.1.1, tomando como guía la documentación del proyecto.

Se analizaron una por una las features para poder determinar cuáles serían las herramientas necesarias para poder implementarlas. Como resultado de este análisis se determinó utilizar FreeLing, tomando para la resolución de algunas features su parser de dependencias y para otras su part-of-speech tagger.

En la tabla 3.1 se detalla cómo se realizó la implementación de todas las features y a partir de qué herramienta de FreeLing se obtiene la información para dicha implementación, con excepción de las features VERB y CLAUSE.

Feature	Entrada	Implementación
PARSER	Dep. Parser	Devuelve si encuentra o no la etiqueta “subj” en alguno de los nodos del árbol de dependencias
LEMMA	PoS Tagger	Lema del verbo principal
NUMBER	PoS Tagger	Se obtiene de la etiqueta gramatical
PERSON	PoS Tagger	Se obtiene de la etiqueta gramatical
INF	PoS Tagger	Cuanta la cantidad de verbos en infinitivo analizando la etiqueta gramatical
NHPREV	Dep. Parser	Cuenta la cantidad de etiquetas “sn” que hay en los nodos del árbol de dependencias que sean anteriores al verbo principal
NHTOT	Dep. Parser	Cuenta la cantidad de etiquetas “sn” que hay en todos los nodos del árbol de dependencias
SE	PoS Tagger	Devuelve si encuentra la palabra “se” al menos a dos palabras de distancia del verbo principal
A	PoS Tagger	Devuelve si se encuentra la palabra “a” con la etiqueta gramatical “SP”
POS_PRE	PoS Tagger	Devuelve las etiquetas gramaticales de las cuatro palabras anteriores al verbo principal
POS_POS	PoS Tagger	Devuelve las etiquetas gramaticales de las cuatro palabras posteriores al verbo principal

Cuadro 3.1: Features implementadas

En la figura 3.1 se muestra un ejemplo de cláusula, mientras que en el cuadro 3.2 se presentan los resultados de la evaluación de sus features.

```
El sarcasmo hiriente del autor enlaza subversivo con la
novela de formación alemana , con el novelar en torno a
un personaje principal que se hace sujeto de sí mismo ,
paralelamente a los hechos que lo determinan
```

Figura 3.1: Cláusula de ejemplo

Feature	PARSER	LEMMA	NUMBER	PERSON	INF	NHPREV	NHTOT	SE	A	POS_PRE	POS_POS
Resultado	True	enlazar	SG	P3	1	2	7	False	True	AQ0CS00 SP DA0MS0 NCMS000	AQ0MS00 SP DA0FS0 NCFS000

Cuadro 3.2: Vector de features para la cláusula de ejemplo presentada en la figura 3.1

La feature VERB describe cuál es el tipo del verbo principal, determinando si el verbo era o no del tipo copulativo, pronominal, transitivo o con uso impersonal. Lamentablemente esta feature no pudo ser implementada, ya que según se describe en la documentación, utiliza listas de verbos de cada tipo, y busca si el verbo de la cláusula que está siendo evaluada existe en ellas. Dado que no se logró obtener una fuente de datos similar, se optó por descartar la feature.

La siguiente feature conflictiva es CLAUSE, la cual evalúa el tipo de la cláusula que está siendo clasificada. Hasta ahora se venían implementando las features sobre cada oración del corpus, pero a partir de analizar esta última feature, se desprende que la evaluación de resultados no se hace directamente sobre cada oración del corpus, sino que se evalúa sobre las cláusulas extraídas de las mismas.

Se aplicaron todas las features excepto las mencionadas y se procedió a entrenar clasificadores sobre las oraciones enteras, utilizando 5-fold Cross Validation de Scikit-learn para entrenar clasificadores de NLTK y los resultados obtenidos fueron: para Naive Bayes el accuracy promedio fue **0,755** y para Maximum Entropy **0,732**.

3.2. Extracción de cláusulas

Como se menciona en la sección 2.2, la resolución del sujeto se debe realizar sobre las cláusulas de una oración y no sobre la oración misma, por lo que es

necesario definir una metodología para poder extraerlas.

3.2.1. Extracción de cláusulas Custom

En el proyecto Elliphant, se menciona que el parser que utiliza no le brinda la información suficiente como para poder delimitar las cláusulas de una oración, por lo que se opta por construir un extractor de cláusulas. Dicho extractor construye cada cláusula identificando los verbos finitos de la oración y luego buscando palabras o caracteres que indiquen el principio o fin de la misma, como pueden ser pronombres relativos, conjunciones, puntos, entre otros.

A simple vista se apreció que este método puede descartar información valiosa de la oración, por ejemplo en la oración *Su libro, por sobre todas las cosas, apunta a ese objetivo.* extrae sólo la cláusula *apunta a ese objetivo*, descartando el sujeto que en la oración está explícito.

Con el fin de no perder esta información, se opta por construir el extractor de cláusulas a partir del árbol resultado de procesar cada oración con el parser de dependencias de FreeLing. Se determina como cláusula a todos los árboles y subárboles cuya raíz sea un verbo finito. Al aplicar esto se observó que no todos los verbos finitos eran raíz de un subárbol e incluso muchas veces dichos verbos eran nodos sin hijos, por lo que esta implementación se descartó, ya que no se lograban identificar todas las cláusulas correctamente.

A continuación se optó por seguir los pasos del proyecto y obtener las cláusulas recorriendo la oración en busca de los límites establecidos. El método que se implementó recorre la oración palabra por palabra generando una lista hasta encontrar uno de los límites establecidos, el siguiente paso es verificar que la lista de palabras contenga un verbo finito, en el caso de que lo contenga se almacena la cláusula, en caso contrario se descarta. Una vez desarrollado el extractor, se utiliza para obtener las cláusulas del corpus, y se observa que el número de cláusulas obtenidas (6497) se asemeja a las obtenidas por el proyecto Elliphant (7086).

Se aplican las features implementadas, esta vez sobre las cláusulas extraídas, y se obtuvieron los siguientes resultados de accuracy promedio.

- Naive Bayes: 0,739

- Maximum Entropy: 0,802
- SVC: 0,752

3.2.2. Extracción de cláusulas con Clatex

La extracción de cláusulas realizada en el proyecto Elliphant descarta mucha información y los resultados obtenidos, a pesar de la mejora con algunos clasificadores, siguen sin aproximarse a los esperados. Es por eso que se buscó otras alternativas para la extracción de cláusulas.

Se utilizó la herramienta Clatex para extraer las cláusulas de la oraciones del corpus, la cual segmenta oraciones en proposiciones, que se tomaron como las cláusulas.

Como primer paso, fue necesario realizar un conjunto de transformaciones sobre los datos para la correcta utilización de la herramienta. Luego, se generaron nuevas fuentes donde las palabras se listan una por línea, junto a su lema, etiqueta gramatical, y probabilidad estimada de la etiqueta seleccionada. Para esto se utilizó el tagger de FreeLing, al cual se le modificó una de sus opciones para no separar los clíticos, ya que esto genera problemas en los siguientes pasos del proceso.

En el siguiente ejemplo, se puede ver la transformación de datos realizada, mostrando la oración original como aparece en el corpus, y el formato necesario para ser procesada por Clatex.

Oración original: *De todas las funciones, destaco aquellas que son relevantes para este estudio de caso:*

Oración adaptada para procesamiento con Clatex:

```
De de SP 0.9999614754213187
todas todo DI0FP0 0.9340731070496084
las el DA0FP0 0.9881835086975967
funciones función NCFP000 0.994186046511628
, , Fc 1.0
destaco destacar VMIP1S0 1.0
aquellas aquel PD0FP00 0.8197674418604651
```

```
que que PR0CN00 0.5501392534733338
son ser VSIP3P0 0.9951967063129004
relevantes relevante AQ0CP00 1.0
para para SP 0.9998335884976369
este este DD0MS0 0.9492011470708727
estudio estudio NCMS000 0.9704301075268816
de de SP 0.9999614754213187
caso caso NCMS000 0.9994450610432852
: : Fd 1.0
. . Fp 1.0
```

Para evitar que Clatex combinara oraciones separadas del corpus se agrega un punto al final de cada una de ellas.

Salida de Clatex:

```
[enonce en2/ [prop gf21/ De todas las funciones ,
destaco aquellas [prop prl1/ que son relevantes para
este estudio de caso /prop prl1] /prop gf21] :
.
/enonce en2]
```

A partir de esta salida se creó otro módulo Python para la extracción de cada una de las cláusulas identificadas por la herramienta. Una diferencia con las cláusulas extraídas con el proceso aplicado en Elliphant es el anidamiento de las mismas, donde la oración original se considera siempre como una de las cláusulas. Por ejemplo en la oración anterior se extraen las cláusulas: *"De todas las funciones, destaco aquellas que son relevantes para este caso"* y *"que son relevantes para este caso"*.

Un problema que se detectó, es que dada la necesidad de transformar la entrada, se perdió la relación con la oración original en el corpus. Esta última contiene la clasificación del sujeto de cada verbo que es necesaria para el entrenamiento y evaluación de los clasificadores, por lo que se programó un nuevo módulo que busca regenerar dicha relación. Este proceso no fue sencillo y tampoco llegó a resultados perfectos, a lo que fue necesario realizar la corrección manual de algunas de las fuentes. Debido a las transformaciones realizadas al texto, muchas cláusulas se perdían ya que no se encontraba en la fuente exactamente la oración que la contenía. Por lo tanto, el método consumía todo el documento sin éxito, perdiendo así todas las cláusulas posteriores de esa fuente en particular. La solución entonces no debe buscar la inclusión del 100 % de la cláusula en la oración, dado que en ocasiones, al reducir todas las oraciones a una única secuencia de tokens para Clatex, algunas fueron combinadas, o el corpus en

ocasiones considera como tokens secuencias de palabras, lo que generaba gran pérdida de cláusulas.

Algunos de los ejemplos observados en el corpus incluyen los siguientes casos:

1. **Verbos compuestos:** Se encuentran en el corpus varios ejemplos de verbos compuestos, los cuales la herramienta FreeLing concatena utilizando el caracter "_". Los mismos son problemáticos ya que se dificulta recrear la relación con la oración original del corpus, así como la extracción de la información del verbo. El corpus en sí no reconoce estas construcciones, tratando cada parte como un token distinto, e incluyendo la información del verbo en una de las partes.

   - *Oración original:* "Puede sugerir que no se tomen decisiones sobre la relación hasta que se den cuenta de cómo llegaron a la situación actual."
   - *Clausula:* 'que se den cuenta de cómo llegaron a la situación actual"
   - *Verbo principal:* "den cuenta" (registrado como "den_cuenta")

2. **Sintagmas nominales compuestos:** Análogo al caso anterior, la herramienta une estas construcciones, transformándolas en un solo token.

   - *Oración original:* "Antes de llegar a acuerdos y decidir como y cuando empezar, esperamos a recabar la información definitiva de l Departamento de Psiquiatría Infantil tras la valoración realizada."
   - *Clausula:* "esperamos a recabar la información definitiva del **Departamento_de_Psiquiatría_Infantil** tras la valoración realizada"

La ocurrencia de estos fenómenos impedía una comparación completa entre las cláusulas y las oraciones originales del corpus, por esto se toma la decisión de utilizar una aproximación entre los tokens de una y otra, considerando poco probable que exista una coincidencia de tal cantidad de tokens de la cláusula y que la misma no pertenezca a la oración.

Para solucionar este problema, se determina si una cláusula pertenece o no a una oración comparando el conjunto de tokens de ambos y calculando la diferencia. Si la cantidad de tokens que pertenecen a la cláusula pero no a la oración es menor a $1/4$ del total de tokens en la cláusula, entonces se asume que la misma está incluida. De esta forma, se recuperan las cláusulas perdidas anteriormente.

3.3. Evaluación

Finalmente se realizaron dos evaluaciones con estas cláusulas. La primera con los resultados obtenidos sin mayores consideraciones o correcciones (por ej. separando clíticos y no considerando oraciones combinadas) y la segunda con todas las correcciones mencionadas.

Total de cláusulas en el corpus (sin correcciones): 3777

- Textos de salud: 2034
- Textos legales: 1734

Accuracy promedio evaluando sobre el total de cláusulas extraídas con ClaTex (sin correcciones):

- Naive Bayes: 0,797
- Maximum Entropy: 0,884
- SVC: 0,883

Total de cláusulas en el corpus (con correcciones): 5597

- Textos de salud: 3468
- Textos legales: 2129

Accuracy promedio evaluando sobre el total de cláusulas extraídas con ClaTex (con correcciones):

- Naive Bayes: 0,783
- Maximum Entropy: 0,866
- SVC: 0,843

En el cuadro 3.3 se detallan las cantidades de cláusulas obtenidas con ambos métodos, siendo *Cláusulas custom* la cantidad de cláusulas obtenidas por el primer método aplicado,

Clatex (corregidas) la cantidad de cláusulas obtenidas utilizando dicha herramienta y *Elliphant* las indicadas en el proyecto.

	Salud	Legales	Total
Cláusulas custom	3187	3310	6497
Clatex (corregidas)	3468	2129	5597
Elliphant	3398	3429	6827

Cuadro 3.3: Total de cláusulas de corpus Elliphant según procedimiento para obtenerlas

	Elliphant	Replicación
Mejor Accuracy	0,867	0,866

Cuadro 3.4: Mejor accuracy obtenido en Elliphant y en su replicación

Como se puede observar en los resultados obtenidos y en el cuadro 3.4, las cláusulas perdidas generaron un aumento significativo en el accuracy del clasificador, que no se mantiene al incorporar las correcciones y las cláusulas recuperadas. Esto probablemente indica que las cláusulas perdidas agregan dificultad para clasificar. Pero si bien los resultados finales tuvieron menor accuracy, igualmente mejoraron sobre los resultados obtenidos con las cláusulas extraídas con el método del proyecto Elliphant. Estos nuevos resultados se aproximan a los resultados de Elliphant al entrenar con el 20 % del corpus, donde el clasificador con mayor accuracy era K*, con 0.850. De todos modos, los resultados se asemejan mucho a los obtenidos con el clasificador K*, el cual entrenando sobre el 100 % del corpus llega a una accuracy de 0.867 (utilizando 10-fold cross validation). Es notorio que los resultados obtenidos con Naive Bayes se encuentran muy por debajo de los obtenidos por la autora, donde en todas sus variantes logran una accuracy por encima de 0.84.

Es necesario notar que los resultados mostrados se obtuvieron sobre un corpus significativamente menor al utilizado por la autora, ya que particularmente en los textos legales, se continúa perdiendo una cantidad considerable de cláusulas. Mientras que con las cláusulas custom, que se aproximan más a las cantidades originales, la accuracy de los clasificadores es peor.

Capítulo 4

Desarrollo

Utilizando lo aprendido con Elliphant se procede a trabajar con el corpus AnCora. El aumento significativo de archivos para procesar llevó a una reestructuración del código existente para aumentar el grado de automatización al momento de procesar los datos. Además, fue necesario procesar estos documentos para llevarlos a un formato más apropiado para la extracción y clasificación de las oraciones. Durante todo este proceso se mantuvo la misma estructura de documentos original, a modo de no perder el fácil acceso al contexto de las oraciones, en caso de ser necesario.

4.1. Primera evaluación

4.1.1. Cálculo de Baseline para clasificadores

Luego de haber procesado el corpus para alcanzar el mismo formato con el que se evaluó Elliphant se procedió a evaluar el mismo. Como primer paso se elabora un baseline descrito en la sección 2.5.4), el cual predice SUBJECT para todas las intancias. El mismo se aplica a ambos corpus, de manera de tener resultados base como referencia. El resultado obtenido para Elliphant es de **0,669** y para AnCora de **0,733**.

También se calcula la distribución de tipos en cada corpus y se detalla en el cuadro 4.1. Siendo "Subject" cuando el sujeto se encuentra explícito en la cláusula, "Zero" cuando se encuentra omitido, "Impersonal" cuando la cláusula es impersonal y "Error" cuando se detecta algún error en la cláusula, ya sea porque no tiene verbo finito o porque éste se detecto erroneamente.

	Corpus	
Tipo de sujeto	Elliphant	AnCora
Subject	3984 (66.96 %)	28067 (73.32 %)
Zero	1755 (29.50 %)	9802 (25.60 %)
Impersonal	189 (3.18 %)	414 (1.08 %)
Error	22 (0,37 %)	0 (0 %)
Total	**5950**	**38283**

Cuadro 4.1: Distribución de tipos en corpus

4.1.2. Evaluación

En los cuadros 4.2 y 4.3 se muestran los resultados obtenidos para los clasificadores *MaxEnt*, *NaiveBayes* y *SVC* utilizando las features PARSER, LEMMA, NUMBER, PERSON, INF, NH_PREV, NH_TOT, SE, A, POS_PRE y POS_POS. Los resultados se obtienen para cada clase definida y para los corpus Elliphant y AnCora, respectivamente.

		SUBJECT	ZERO	IMPERSONAL	ERROR
MaxEnt	Precision	0,881	0,816	0,825	0,800
	Recall	0,928	0,742	0,648	0,600
	F-Measure	0,904	0,777	0,725	0,667
	Avg. Accuracy				**0,863**
NaiveBayes	Precision	0,875	0,638	0,549	0,333
	Recall	0,815	0,753	0,463	0,167
	F-Measure	0,844	0,691	0,502	0,214
	Avg. Accuracy				**0,783**
SVC	Precision	0,807	0,988	0,965	0,800
	Recall	0,996	0,503	0,651	0,633
	F-Measure	0,892	0,666	0,755	0,693
	Avg. Accuracy				**0,838**

Cuadro 4.2: Resultados por clasificador para corpus Elliphant en primera evaluación

El primer resultado que salta a la vista es la diferencia de los clasificadores entre ambos corpus. En el corpus Elliphant los tres clasificadores evaluados son bastante buenos evaluando las tres categorías, siendo MaxEnt una excepción ya que tiene resultados bastante pobres en las categorías Zero e Impersonal en comparación a los otros dos.

Por otra parte, si bien con AnCora los clasificadores tienen un rendimiento aceptable para clasificar cláusulas del tipo Subject, sus resultados son pésimos en las otras dos categorías.

		SUBJECT	ZERO	IMPERSONAL	ERROR
MaxEnt	Precision	0,379	0,864	0,557	0,0
	Recall	0,571	0,757	0,192	0,0
	F-Measure	0,687	0,505	0,285	0,0
	Avg. Accuracy				**0,614**
NaiveBayes	Precision	0,799	0,556	0,164	0,0
	Recall	0,881	0,371	0,326	0,0
	F-Measure	0,838	0,455	0,214	0,0
	Avg. Accuracy				**0,744**
SVC	Precision	0,746	0,806	0,750	0,0
	Recall	0,994	0,073	0,031	0,0
	F-Measure	0,852	0,133	0,059	0,0
	Avg. Accuracy				**0,747**

Cuadro 4.3: Resultados por clasificador para corpus AnCora en primera evaluación

Para AnCora, vemos que el clasificador SVC tiene una buena posibilidad de estar en lo correcto al clasificar una cláusula como Zero o Impersonal. El problema es que clasifica correctamente a una ínfima fracción de las ocurrencias de estos tipos. MaxEnt, sin embargo, se muestra bastante capaz de evaluar cláusulas del tipo Zero, pero sus resultados para el tipo Subject son muy pobres.

Una de las conclusiones más valiosas que se desprenden de este análisis, es que el accuracy general es muy engañoso, ya que resulta principalmente del rendimiento sobre el tipo Subject, sin revelar los problemas con las demás categorías.

Esto, sumado a que si comparamos el accuracy con el resultado del baseline para AnCora, se puede notar que los mismos son extremadamente similares, es decir, las features implementadas no parecen tener ningún efecto sobre la clasificación.

A partir de esto se comienza a analizar algunas características del proceso de clasificación, buscando mejorar su rendimiento.

4.2. Ajustes al modelo

4.2.1. Extracción del parser de dependencias

Surge la duda de cuánto aporta el parser de dependencias a los resultados obtenidos, teniendo en cuenta que utilizar dicho parser se puede ver como tomar un atajo para llegar a la solución final ya que nos brinda la información de si encuentra o no un sujeto en el texto analizado. Por otra parte, dado que parser de dependencias de FreeLing es lento, se genera un aumento en el costo de la evaluación de cada cláusula.

Para saber si realmente aporta a los resultados es necesario evaluar el corpus modificando las features utilizadas para que no usen dicho parser. Lamentablemente no todas las features se pueden modificar, ya que algunas solo se pueden calcular con los datos obtenidos por el parser que se quiere extraer. Por lo que se toma la decisión de evaluar los corpus pero sin estas features que sí o sí dependen del parser de dependencias. Dichas features son: SUBJECT, NH_TOT y NH_PREV.

Al realizar las nuevas evaluaciones los resultados llaman la atención, pues el accuracy promedio para los clasificadores Naive Bayes y SVC crece alrededor de un uno por ciento en los dos corpus. Sin embargo para MaxEnt el accuracy promedio baja alrededor de un uno por ciento. Igualmente esto no deja de ser novedoso, ya que antes de calcular los resultados se pensaba que las features que se sacaron, sobretodo la de PARSER, tenía más peso en el resultado final.

En el cuadro 4.4 se listan los accuracy promedios obtenidos de esta evaluación para los corpus Elliphant y AnCora.

	Corpus	
Tipo de clasificador	Elliphant	AnCora
Naive Bayes	0,791	0,749
Maximum Entropy	0,858	0,608
SVC	0,841	0,755

Cuadro 4.4: Accuracy obtenida durante evaluación sin parser de dependencias

4.2.2. POS-tagging

En el caso de las features que indican los part-of-speech tags de las cuatro palabras anteriores y posteriores del verbo principal de las clases, se eliminaron elementos del

tag, mejorando ligeramente los resultados obtenidos. Específicamente, se eliminaron las partes de los tags que indican el tipo y modo de los verbos. Reducir las tags de otros tipos de elemento no tuvo efectos sobre los rendimientos de los clasificadores.

4.2.3. Nuevas features

Como recurso adicional se implementaron dos features adicionales: NH_TOT_AGREE y NH_PREV_AGREE. Similarmente a NH_TOT y NH_PREV cuentan las noun phrases que son candidatas a ser el sujeto de la cláusula, pero que en este caso deben coincidir en persona y número con el verbo principal de la cláusula. En la figura 4.1 se muestra nuevamente una cláusula de ejemplo del corpus, así como la evaluación de las nuevas features implementadas (ver cuadro 4.5). Las mismas se suman al resto del vector de features que se muestra en el cuadro 3.2.

```
El sarcasmo hiriente del autor enlaza subversivo con la
novela de formación alemana , con el novelar en torno a
un personaje principal que se hace sujeto de sí mismo ,
paralelamente a los hechos que lo determinan
```

Figura 4.1: Cláusula de ejemplo

Feature	NH_TOT_AGREE	NH_PREV_AGREE
Resultado	6	2

Cuadro 4.5: Valores de features nuevas para ejemplo de cláusula de figura 4.1

4.2.4. Nueva evaluación

En los cuadros 4.6 y 4.7 se muestran los resultados obtenidos en la nueva evaluación sobre el corpus Elliphant y AnCora, respectivamente, utilizando las features antes usadas PARSER, LEMMA, NUMBER, PERSON, INF, NH_PREV, NH_TOT, SE, A, POS_PRE y POS_POS, y agregando las nuevas features NH_TOT_AGREE y NH_PREV_AGREE.

En AnCora se nota una clara mejoría en los resultados obtenidos. Todos los clasificadores obtuvieron mejores resultados luego de las correcciones mencionadas. Quien más se beneficio fue MaxEnt, el cual aumentó su accuracy de 0,61 a 0,85 además de mejorar notoriamente su precision tanto para los tipos Subject e Impersonal (perdiendo un poco para la categoría Zero), y aumentando el recall en todos los casos.

		SUBJECT	ZERO	IMPERSONAL	ERROR
MaxEnt	Precision	0,883	0,820	0,804	0,800
	Recall	0,930	0,746	0,622	0,600
	F-Measure	0,905	0,781	0,700	0,667
	Avg. Accuracy				**0,864**
NaiveBayes	Precision	0,873	0,606	0,514	0,468
	Recall	0,781	0,759	0,465	0,300
	F-Measure	0,824	0,674	0,483	0,356
	Avg. Accuracy				**0,762**
SVC	Precision	0,812	0,984	0,965	0,800
	Recall	0,995	0,518	0,651	0,633
	F-Measure	0,894	0,679	0,775	0,693
	Avg. Accuracy				**0,842**

Cuadro 4.6: Resultados por clasificador para corpus Elliphant en segunda evaluación

		SUBJECT	ZERO	IMPERSONAL	ERROR
MaxEnt	Precision	0,889	0,760	0,701	0,0
	Recall	0,927	0,660	0,398	0,0
	F-Measure	0,903	0,706	0,507	0,0
	Avg. Accuracy				**0,853**
NaiveBayes	Precision	0,905	0,525	0,219	0,0
	Recall	0,738	0,766	0,576	0,0
	F-Measure	0,813	0,623	0,316	0,0
	Avg. Accuracy				**0,743**
SVC	Precision	0,770	0,873	0,900	0,0
	Recall	0,990	0,194	0,078	0,0
	F-Measure	0,866	0,317	0,141	0,0
	Avg. Accuracy				**0,776**

Cuadro 4.7: Resultados por clasificador para corpus AnCora en segunda evaluación

Si se observa en detalle se pueden notar grandes cambios. Por ejemplo, Naive Bayes aumentó el recall de cláusulas del tipo Zero e Impersonal de 0,37 y 0,32 a 0,76 y 0,57 respectivamente, aunque también es cierto que el mismo valor pasó de 0,88 a 0,73 para el tipo Subject.

En el corpus Elliphant los cambios no fueron particularmente significativos, pero sí muestran una tendencia positiva.

4.3. Mejoras en el modelo

En este punto se llegó a un límite con la librería utilizada previamente, NLTK, que si bien fue de mucha utilidad al comienzo, carece de algunos recursos más avanzados que se pretendía usar. Para esto, se decidió migrar completamente a la librería Scikit-learn, de la cual ya se utilizaban algunos recursos, como el clasificador SVC y sus cálculos de métricas.

4.3.1. Vectorización

Los resultados posteriores a esta sección implementan una versión más sofisticada de la vectorización de las features del modelo. Esta tarea se realiza mediante la codificación One Hot, definida en la sección 2.5.1, que transforma cada feature en un vector de coordenadas binarias, y largo n, siendo n la cantidad de valores distintos observados para la feature en el corpus de entrenamiento de AnCora, sobre el cual se ajustó el clasificador. El vector correspondiente a cada feature tiene como máximo un solo bit encendido, el correspondiente al valor de la feature evaluada, y todos los demás en cero.

4.3.2. Feature VERB TYPE

Al comienzo se había descartado esta feature ya que no se habían encontrado las listas de verbos utilizados por el proyecto Elliphant. Pero con el fin de mejorar la clasificación de los sujetos impersonales se decide implementar esta feature utilizando otras listas de verbos obtenidas mediante scrapping de Wikcionario[1]. La nueva feature, a la que llamaremos VERB_TYPE, retorna una tupla de tres valores booleanos, que indican si el verbo principal de la cláusula es impersonal, si es exclusivamente pronominal, o si tiene uso pronominal. Estas categorías tienen 64, 705, y 2111 verbos respectivamente. En el cuadro 4.8 se muestra el resultado obtenido al evaluar estar nuevas features en la cláusula de la figura 4.1.

[1]https://es.wiktionary.org/ - Último acceso 2018/11/13

Feature	IMPERSONAL	PRONOMINAL_ONLY	PRONOMINAL_USE
Resultado	False	False	True

Cuadro 4.8: Valores de features VERB_TYPE para ejemplo de cláusula de figura 4.1

4.3.3. Word2Vec

El sujeto de una cláusula está fuertemente ligado con el verbo principal de la misma, por lo que se considera pertinente tener features para toda la información que este brinda. Ya se tienen features con la información morfológica del verbo, así como también una con su lema, pero no se está utilizando el verbo tal como aparece en la cláusula. Pero como el conjunto de verbos que pueden aparecer es muy grande, se opta por utilizar una matriz de Word2vec[1], agregando una nueva feature, llamada MAIN_VERB_VECTOR, con el vector que representa dicho verbo en la matriz. En pruebas posteriores se reemplaza además la feature LEMA utilizada anteriormente por el vector correspondiente del mismo.

Cabe destacar que al momento de codificar el vector de features correspondientes a una cláusula mediante el One Hot encoder mencionado previamente, los elementos correspondientes al vector del verbo principal de la cláusula, así como el vector correspondiente a su lema (en el caso en el que el mismo se representa de ese modo) son excluidos de los vectores a evaluar. Esto se debe a que en realidad estos valores no referencian categorías, sino un espacio vectorial, por lo que el codificador restaría información en este caso. Posteriormente, se unen en el mismo vector los valores correspondientes a las features categóricas con los valores de los vectores, y el valor conjunto es el que se envía al clasificador.

4.3.4. Procesamiento de cláusulas

Analizando los errores durante el desarrollo se notó que en algunos casos se estaba calculando mal el verbo principal de la cláusula y además se guardaban varias cláusulas con el mismo verbo principal. Para solucionar esto se modificó la forma de obtener las cláusulas. De aquí en adelante, las cláusulas se guardan con su verbo principal ya identificado a partir de la salida de Clatex, se ignoran cláusulas si una cláusula anterior proveniente de la misma oración tiene el mismo verbo principal, y al momento de calcular las features que dependan del verbo principal, se utiliza el almacenado junto a la cláusula. Esto resulta en la eliminación de cláusulas redundates donde se repite el mismo verbo principal.

4.4. Evaluación general del corpus

Se implementó un nuevo módulo de entrenamiento y evaluación utilizando el nuevo esquema mencionado, abandonando también el método de Cross-Validation. En las pruebas mencionadas a continuación se utilizó un 80 % del corpus total para entrenamiento, y se utilizó el 20 % restante para la evaluación del clasificador, como se describió en la seccion 2.5.2.

Este proceso se realizó utilizando ocho clasificadores: LogisticRegression, SGDClassifier, GaussianNB, BernoulliNB, SVC, LinearSVC, RandomForestClassifier y VotingClassifier. El último es un clasificador conjunto que utiliza los primeros siete clasificadores y determina un resultado mediante una votación democrática entre ellos. Particularmente, se utiliza la estrategia *hard-vote*, donde el voto de cada clasificador tiene el mismo valor, como se menciono en la sección 2.5.8.

4.4.1. Evaluación sobre AnCora

La distribución por clases del corpus AnCora se ilustra en el cuadro 4.9.

	Training	Testing
SUBJECT	68,79 %	69,39 %
ZERO	29,99 %	29,28 %
IMPERSONAL	1,22 %	1,32 %

Cuadro 4.9: Composición de corpus AnCora

	SUBJECT			ZERO			IMPERSONAL			
	P	R	F	P	R	F	P	R	F	Acc
LogisticRegression	0,886	0,929	0,907	0,766	0,677	0,719	0,857	0,366	0,513	0,859
SGDClassifier	0,890	0,917	0,903	0,738	0,694	0,715	0,893	0,305	0,455	0,854
GaussianNB	0,888	0,106	0,190	0,238	0,702	0,356	0,045	0,695	0,085	0,264
BernoulliNB	0,886	0,753	0,814	0,510	0,705	0,592	0,261	0,585	0,361	0,739
SVC	0,777	0,988	0,870	0,847	0,219	0,349	0,0	0,0	0,0	0,782
LinearSVC	0,890	0,928	0,909	0,766	0,686	0,724	0,796	0,476	0,595	0,862
RandomForestClassifier	0,850	0,939	0,892	0,753	0,538	0,628	0,884	0,463	0,608	0,832
VotingClassifier	0,885	0,935	0,909	0,777	0,667	0,718	0,892	0,402	0,555	0,861

Cuadro 4.10: Precision, Recall, F-Measure y Accuracy según clasificador y clase para corpus AnCora

En el cuadro 4.10 se presenta en detalle los resultados obtenidos al evaluar cada clasificador sobre el corpus AnCora, como también la *accuracy* obtenida para cada uno, utili-

zando las features PARSER, LEMMA, NUMBER, PERSON, INF, NH_PREV, NH_TOT, SE, A, POS_PRE, POS_POS, NH_TOT_AGREE, NH_PREV_AGREE, MAIN_VERB_VECTOR y VERB_TYPE. Se observa que los clasificadores **LogisticRegression**, **LinearSVC** y **VotingClassifier** son los que obtuvieron mejores resultados en general.

Si bien la *precision* obtenida sobre las tres clases es muy buena para estos tres clasificadores, se observa que la mayor deficiencia se encuentra en el *recall* tanto para la clase ZERO como IMPERSONAL. Esto no es raro en el caso de la clase IMPERSONAL, debido a que existen muy pocos ejemplos en el corpus. En el caso de la clase ZERO, si bien los resultados no son terribles, tampoco llegan a compararse con los obtenidos para la clase SUBJECT. De cualquier modo, son resultados alentadores, y es muy posible que los mismos mejoren al aumentar la cantidad de instancias en el corpus de entrenamiento.

4.4.2. Evaluación sobre Elliphant

Para comprobar que los resultados obtenidos son comparables a los del trabajo base, se vuelve a realizar una evaluación sobre el corpus Elliphant. En este caso, se realizan dos pruebas: En la primera, utilizando los clasificadores entrenados sobre AnCora, se evalúa su actuación al clasificar el corpus Elliphant en su totalidad (cuyos resultados se muestran en el cuadro 4.11.) La segunda, realizando una prueba análoga a la descrita en la sección 4.4, donde se entrenan los clasificadores sobre el 80 % del corpus, y se evalúa en el 20 % restante (ver cuadro 4.12).

	SUBJECT			ZERO			IMPERSONAL			
	P	R	F	P	R	F	P	R	F	Acc
LogisticRegression	0,873	0,852	0,862	0,604	0,677	0,638	0,387	0,223	0,283	0,790
SGDClassifier	0,929	0,664	0,775	0,492	0,840	0,620	0,151	0,346	0,210	0,699
GaussianNB	0,857	0,167	0,280	0,272	0,831	0,409	0,096	0,362	0,151	0,338
BernoulliNB	0,891	0,691	0,778	0,466	0,751	0,575	0,192	0,285	0,229	0,694
SVC	0,882	0,713	0,788	0,464	0,769	0,579	0,0	0,0	0,0	0,708
LinearSVC	0,871	0,843	0,857	0,601	0,674	0,636	0,259	0,215	0,235	0,783
RandomForestClassifier	0,817	0,871	0,843	0,555	0,494	0,523	0,541	0,154	0,240	0,756
VotingClassifier	0,891	0,812	0,850	0,569	0,740	0,643	0,373	0,238	0,291	0,778

Cuadro 4.11: Resultados de evaluación sobre corpus Elliphant con clasificadores entrenados en corpus AnCora

En el primer caso, se observa que los resultados obtenidos al utilizar los clasificadores entrenados en AnCora son similares a los obtenidos al evaluar sobre AnCora (ver cuadro 4.10). Esto es un buen indicio de la calidad del corpus utilizado, ya que se logran buenos resultados a pesar de ser tipos de texto muy distintos. Aún así, se observa que la actuación baja en mayor medida en el caso de las cláusulas impersonales. Es posible que

las mismas sean muy distintas en un corpus y en otro, lo que aumenta la dificultad para poder detectarlas.

	SUBJECT			ZERO			IMPERSONAL			
	P	R	F	P	R	F	P	R	F	Acc
LogisticRegression	0,858	0,922	0,889	0,735	0,623	0,675	0,846	0,379	0,524	0,832
SGDClassifier	0,896	0,743	0,812	0,506	0,775	0,612	0,875	0,483	0,622	0,744
SVC	0,720	1,0	0,837	0,0	0,0	0,0	0,0	0,0	0,0	0,720
LinearSVC	0,862	0,884	0,873	0,659	0,630	0,644	0,727	0,552	0,627	0,811
RandomForestClassifier	0,809	0,930	0,865	0,673	0,409	0,509	0,737	0,483	0,583	0,787
VotingClassifier	0,854	0,931	0,891	0,749	0,594	0,663	0,875	0,483	0,622	0,833

Cuadro 4.12: Resultados de evaluación sobre 20 % de corpus Elliphant con clasificadores entrenados en 80 % de corpus Elliphant

Los resultados del segundo caso son esperables, pero sirven para validar la calidad del modelo. En el mismo los clasificadores obtienen mejor rendimiento en algunos casos y mejora ampliamente el reconocimiento de cláusulas de tipo ZERO e IMPERSONAL. Como se mencionó previamente, se trata de un corpus de contenido muy particular, por lo que naturalmente los clasificadores tienen mejores posibilidades de aprender a clasificar este tipo de textos cuando se entrena con ellos.

Clase	Precision	Recall	F-Measure
SUBJECT	90,1 %	92,3 %	91,2 %
ZERO	77,2 %	74,0 %	75,5 %
IMPERSONAL	85,6 %	63,1 %	72,7 %

Cuadro 4.13: Resultados originales de Elliphant (Clasificador K*) [17]

Comparando el total de los resultados obtenidos con los expuestos en el trabajo Elliphant, como se muestra en el el cuadro 4.13, se puede ver que son razonablemente parecidos a los obtenidos utilizando el corpus AnCora en su totalidad. De todas formas existen diferencias en los resultados obtenidos para la clase IMPERSONAL, donde se reportan valores de *precision* y *recall* superiores a los obtenidos en este trabajo. Un punto importante a tener en cuenta al momento de realizar esta comparación, es que no fue posible averiguar cómo fue hecha la división del corpus de entrenamiento y evaluación, por lo que es posible que una diferencia en este aspecto afecte los resultados finales. En este trabajo tampoco fueron calculados los valores de *accuracy* para cada clase en particular, sino que solo se obtuvo el *accuracy* de los clasificadores en general.

4.4.3. Evaluación sobre Corin

Como forma de realizar una evaluación limpia del trabajo realizado, se utilizó un corpus completamente nuevo, descrito en la sección 2.4.3.

	SUBJECT			ZERO			IMPERSONAL			
	P	R	F	P	R	F	P	R	F	Acc
LogisticRegression	0,806	0,841	0,823	0,425	0,647	0,513	0,900	0,122	0,214	0.697
LinearSVC	0,806	0,833	0,819	0,428	0,647	0,515	0,867	0,176	0,292	0,698
VotingClassifier	0,804	0,843	0,823	0,433	0,647	0,519	0,909	0,135	0,235	0,700

Cuadro 4.14: Resultados de evaluación de clasificadores con corpus Corin

La evaluación se realizó utilizando los clasificadores entrenados con el corpus AnCora. Nuevamente se seleccionan los mismos clasificadores que mostraron el mejor desempeño en las pruebas anteriores, cuyos resultados se exhiben en la tabla 4.14.

Varias cosas se destacan al observar los datos obtenidos. En primer lugar, los resultados son marcadamente peores que a los obtenidos al evaluar sobre AnCora presentados en el cuadro 4.10. Particularmente, las clases que más sufren en este nuevo corpus son las de tipo ZERO e IMPERSONAL. Si bien la *precision* para la clase IMPERSONAL es muy buena, rondando el 90 %, el *recall* es muy bajo, promediando el 15 %. Por otra parte, en la clase ZERO, tanto la precisión como el *recall* se reducen a casi la mitad de las medidas observadas en el corpus AnCora de evaluación.

Algo interesante de notar, es que en el corpus Corin, la distribución de tipos en las cláusulas de tipo, como se muestra en el cuadro 2.1, difiere de los corpus utilizados anteriormente en la mayor cantidad de cláusulas impersonales que contiene. Además, se observaron construcciones impersonales no vistas en el corpus de AnCora, donde la totalidad de las cláusulas así etiquetadas utilizan la forma “se”. Esto lleva a pensar que o bien el corpus AnCora solo marca este tipo de construcciones impersonales, o que no se lograron obtener todas las cláusulas de este tipo del mismo. Cabe recordar que se utilizó un atributo especial del corpus que señala los verbos como impersonales, pero no cuenta con tanta información como en los casos donde el sujeto esta omitido. Una conclusión razonable en base a esta información es que faltan features destinadas a identificar otros tipos de construcciones impersonales.

Otra diferencia importante de este corpus sobre los anteriores es el origen, ya que si bien se trata también de textos de prensa, los mismos son de origen nacional, por lo que existen diferencias considerables dado que los clasificadores fueron entrenados en base a cláusulas formuladas en español de España. Estas diferencias pueden jugar un papel importante en la disminución de performance de los clasificadores.

Capítulo 5

Análisis de errores

En esta sección se analizarán los resultados obtenidos sobre el corpus de AnCora, a modo de obtener información sobre los errores cometidos por los clasificadores utilizados. Para esto se utilizarán las matrices de confusión de los tres clasificadores que mejor accuracy obtuvieron en las evaluaciones mencionadas en la sección 4.4, así como un análisis sobre las features implementadas con respecto a los resultados obtenidos. Vale aclarar que los valores presentados a continuación son sobre la partición de testing del corpus de AnCora.

5.1. Matriz de confusión

			Valor predicho		
			SUBJECT	ZERO	IMPERSONAL
LinearSVC	Valor real	SUBJECT	3434	238	2
		ZERO	366	1176	9
		IMPERSONAL	23	15	32
LogisticRegression	Valor real	SUBJECT	3457	215	2
		ZERO	361	1185	5
		IMPERSONAL	27	15	28
VotingClassifier	Valor real	SUBJECT	3461	210	2
		ZERO	371	1174	5
		IMPERSONAL	27	15	28

Cuadro 5.1: Matrices de Confusión sobre la partición de testing del corpus de AnCora

Observando el cuadro 5.1 se nota, como se esperaba, que la mayor confusión para todos los clasificadores sucede en las clases ZERO e IMPERSONAL, las cuales son clasificadas casi exclusivamente como SUBJECT, que es la clase más común en el corpus. Por otra parte, en las ocasiones en las que clasifica erróneamente cláusulas del tipo SUBJECT, muestra una preferencia marcada por la categoría ZERO, que es la segunda más común del corpus.

En la categoría IMPERSONAL se observa un aspecto interesante, donde un tercio de las ocasiones sus cláusulas se clasifican como SUBJECT, pero rara vez se marca como IMPERSONAL una cláusula que no lo es, como se observa en la medida de precisión para los mismos tres clasificadores del cuadro 4.10.

5.2. Análisis por feature

Antes de comenzar efectivamente con el análisis de errores, primero se debe preparar los datos a evaluar. Para esto se selecciona el clasificador LinearSCV, mencionado anteriormente, sobre el que se realizará el análisis en profundidad. Dado que los mayores problemas se presentan en las clases ZERO e IMPERSONAL, el foco será puesto en ellas.

La preparación de datos consiste en analizar todas las cláusulas de la partición de testing, y para cada una de ellas almacenar en un objeto la cláusula, su respectivo vector de features (el cual no incluye los valores correspondientes al vector de Word2vec del verbo), su valor predicho y su valor real.

Con los datos almacenados se obtienen las matrices de confusión para las distintas features que se consideran de mayor importancia a la hora de clasificar una cláusula como ZERO o IMPERSONAL, en particular features con poca dispersión, como lo son aquellas features binarias y numéricas. Este análisis se realizó conjuntamente con la evaluación general del corpus, descrita en la sección 4.4.

5.2.1. Cláusulas Impersonales

Los resultados obtenidos con la feature VERB_TYPE (IMPERSONAL), ilustrados en el cuadro 5.2 fueron inesperados, dado que esta feature se incluyó específicamente para ayudar con la detección de construcciones impersonales. Si bien el listado de verbos de uso impersonal es pequeño (el mismo se encuentra adjunto en el apéndice B), contando con solo 64 instancias e incluyendo verbos que indican fenómenos de la naturaleza que siempre generan construcciones impersonales, y dado que no es necesario un verbo

exclusivamente impersonal para construir una oración de este tipo, lo esperable es que la evaluación de esta feature indicara generalmente que el verbo no está en la lista, esto se puede verificar en el cuadro 5.2. Sin embargo, no se esperaba que el clasificador diera mejores resultados para detectar una cláusula impersonal cuando la evaluación de la feature señalaba que el verbo no pertenece a la lista, ya que si la feature marcaba que el verbo pertenece a la lista de verbos impersonales se esperaba fuera porque se estaba ante una cláusula impersonal, pero esos resultados no fueron los obtenidos.

Los siguientes son ejemplos de cláusulas impersonales donde la feature VERB_TYPE (IMPERSONAL) indica que el verbo principal está incluido en la lista de verbos impersonales. El ejemplo de la figura 5.1 el clasificador marca correctamente que se trata de una cláusula impersonal, sin embargo la cláusula de la figura 5.2 el clasificador indica incorrectamente que se trata de una cláusula con sujeto.

si se le **hubiese** enyesado la pierna

Figura 5.1: Cláusula impersonal de AnCora original

aunque desde muchos rincones de la sociedad se **ha** avisado en los últimos meses de los problemas que puede acarrear esta medida revolucionaria

Figura 5.2: Cláusula impersonal de AnCora original

Verbo incluido en lista de verbos impersonales	Tipo de sujeto anotado	Tipo de sujeto predicho		
		SUBJECT	ZERO	IMPERSONAL
SI	SUBJECT	539	35	1
SI	ZERO	53	168	2
SI	IMPERSONAL	6	3	6
NO	SUBJECT	2895	202	1
NO	ZERO	313	1007	7
NO	IMPERSONAL	17	12	26

Cuadro 5.2: Matrices de Confusión para los distintos resultados de la feature "Verb Type - Impersonal Verb"

Otra feature utilizada principalmente por su relación con las cláusulas impersonales es la feature "Se". Para la misma, habiendo observado varios ejemplos de este tipo de cláusulas en el corpus, se esperaba que la evaluación de esta feature como verdadera se aproximara aún más al 100 %. Por esta razón se volvió a evaluar la implementación de la misma, notando que no se había realizado correctamente debido a que no se consideraban los casos donde el pronombre "se" se encuentra separado del verbo principal por uno o más tokens. Por lo mismo, en la siguiente cláusula impersonal, la

feature evaluaba a falso:
"A la principal , Maishima , **se** *la* **denominará** la Isla Olímpica ."
Solucionando esto, los resultados para esta feature pasan a ser verdadera en todos los casos para cláusulas impersonales, tanto cuando estas se clasifican correctamente como cuando no, esto se puede ver en el cuadro 5.3 y en los ejemplos de las figuras 5.3 y 5.4 de cláusulas impersonales donde siempre esta presente el pronombre "se" pero no siempre se clasifican correctamente, al ejemplo de la figura 5.3 clasificada como IMPERSONAL correctamente y el de la figura 5.4 clasificada como SUBJECT incorrectamente.

Indocumentado y a esas horas, **se** trataría de un drogadicto

Figura 5.3: Cláusula impersonal de AnCora original

Por el momento no **se** sabe cuándo se marchará el jefe de la delegación siria, Faruk Al Chará, ministro de Asuntos Exteriores

Figura 5.4: Cláusula impersonal de AnCora original

Ocurre "se"	Tipo de sujeto anotado	Tipo de sujeto predicho		
		SUBJECT	ZERO	IMPERSONAL
SI	SUBJECT	428	17	2
SI	ZERO	38	83	8
SI	IMPERSONAL	23	15	32
NO	SUBJECT	3006	220	0
NO	ZERO	328	1092	1
NO	IMPERSONAL	0	0	0

Cuadro 5.3: Matrices de Confusión para los distintos resultados de la feature "Se"

Otro fenómeno notado analizando la feature LEMA, es que la dispersión de la misma es menor dentro de las cláusulas impersonales clasificadas correctamente. Un caso destacable, es que el verbo "tratar" es el verbo principal en el 57 % de las cláusulas clasificadas correctamente, pero éste aparece raramente entre las cláusulas erróneas, donde el verbo dominante es "haber". Cabe mencionar, que dentro de las cláusulas impersonales utilizadas para entrenar los clasificadores, el verbo "tratar" es el más común, siendo el verbo principal en casi el 26 % de las cláusulas, siendo el verbo "haber" el segundo más común, con un 11 %. En términos generales, la dispersión de esta feature en las cláusulas clasificadas incorrectamente es aproximadamente el doble que en las cláusulas clasificadas correctamente.

La feature "A", cuyo comportamiento se muestra en la tabla 5.4, también muestra una diferencia pronunciada entre las cláusulas según si su clasificación fue correcta o no. Particularmente, esta feature toma un valor falso más frecuentemente entre las cláusulas clasificadas correctamente.

Ocurre "a"	Tipo de sujeto anotado	Tipo de sujeto predicho		
		SUBJECT	ZERO	IMPERSONAL
SI	SUBJECT	1124	78	0
SI	ZERO	100	321	6
SI	IMPERSONAL	10	5	11
NO	SUBJECT	2310	159	3
NO	ZERO	266	854	3
NO	IMPERSONAL	13	10	11

Cuadro 5.4: Matrices de Confusión para los distintos resultados de la feature "A"

Observando el comportamiento de features referentes a las tags POS que rodean al verbo principal, se observan algunas diferencias. Por ejemplo, la cantidad de valores distintos que toma la feature POS_{pre} en la segunda posición anterior al verbo principal, es la mitad que en los casos en los que se clasifica mal. Sin embargo, los valores más comunes en ambos casos son muy similares, siendo el valor "CS", correspondiente a una conjunción subordinada, en más del 30 % de las cláusulas que son clasificadas correctamente, y en aproximadamente el 20 % cuando no.

Finalmente, en features que realizan un conteo de situaciones particulares, se observan también diferencias. Se puede ver en le cuadro 5.5 que en el caso de la cantidad de sintagmas nominales previos al verbo, cuando se clasifica correctamente como impersonal, el valor de esta feature es 0 en casi el 90 % de los casos, mientras que cuando el clasificador comete un error, este valor sólo aparece en aproximadamente un 60 % de los casos. De esto se puede interpretar que es una feature que colabora en la identificación de esta clase.

5.2.2. Cláusulas con sujeto implícito

En el caso de la features correspondiente a los POS tag anteriores al verbo principal se observa que la proporción de cláusulas donde los valores de esta feature son vacíos, es decir, cuando no hay token anterior al verbo en esa posición, aumenta casi de manera

Sintagmas nominales previos al verbo	Tipo de sujeto anotado	Tipo de sujeto predicho		
		SUBJECT	ZERO	IMPERSONAL
0	SUBJECT	708	200	2
0	ZERO	234	1085	8
0	IMPERSONAL	13	15	28
1	SUBJECT	1453	22	0
1	ZERO	75	67	1
1	IMPERSONAL	4	0	2
>=2	SUBJECT	1273	15	0
>=2	ZERO	57	23	0
>=2	IMPERSONAL	6	0	2

Cuadro 5.5: Matrices de Confusión para los distintos resultados de la feature "NH_PREV"

lineal cuanto más se aleja del verbo. Además, en aquellas cláusulas clasificadas como ZERO, este valor es aproximadamente 20 % mayor que en aquellas clasificadas como SUBJECT. Esto no sucede con las features POS tag posteriores al verbo, donde no hay mayor diferencia entre las que se clasifican bien y las que se clasifican mal, y donde tampoco hay gran ausencia de tokens.

Las features "Parser" y "NH_PREV", ilustradas en los cuadros 5.6 y 5.5 respectivamente son las que se suponía iban a jugar un rol importante en la clasificación de las clausulas con sujeto implícito. Particularmente, la feature "Parser" evalúa precisamente si el verbo principal cuenta con un sujeto asociado. Claramente esta prueba no es totalmente certera, ya que sino este trabajo no tendría sentido, pero sí aporta información importante. Por razones similares el comportamiento de la feature "NH_PREV" se ajusta a las asunciones previas, particularmente, analizándolo en conjunto con la feature "NH_TOT", la cual toma el valor cero con mucha menos frecuencia. Una razón para esto puede ser que si bien no se incluye el sujeto del verbo principal, el mismo puede tener un sintagma nominal como objeto asociado. Además, es poco probable que un verbo se marque como teniendo sujeto explícito ante la total ausencia de candidatos posibles.

Existe otra explicación para la diferencia entre los valores de la feature "NH_PREV" y "NH_TOT", y es el caso de las cláusulas con sujetos pos-verbales. El clasificador desarrollado parece particularmente incapaz de reconocer estas construcciones, donde se observa que en estos casos, la clase asignada normalmente es la clase ZERO. Utilizando el clasificador construido, se realiza la clasificación de oraciones obtenidas de un documento educativo[1] de la Administración Nacional de Educación Pública (ANEP)

[1]http://www.anep.edu.uy/prolee/phocadownload/materiales/docentes/glosario/el-

que incluye ejemplos de oraciones con sujetos pos-verbales. Clasificando las oraciones "Eso *dijimos* **nosotros**", "*Llegaron* **los invitados**", y "*Quedó* **un libro** sobre la mesa", el clasificación retorna una predicción de la clase ZERO para las tres, a pesar de contar con un sujeto explícito luego del verbo principal.

```
Este colectivo verá como la tributación que soportan en el
impuesto sobre la renta desciende una media de un punto,
del 27% al 26%.
```

Figura 5.5: Oración de AnCora original

```
[enonce en2/ [prop ge22/ Este colectivo verá como la
tributación [prop prl4/ que soportan en el impuesto sobre
la renta /prop prl4] /prop ge22] [prop gf21/ desciende una
media de un punto , del 27% al 26% /prop gf21] . /enonce
en2]
```

Figura 5.6: Oración de la figura 5.5, procesada por Clatex

Es necesario mencionar también que algunos de los errores observados son introducidos durante el proceso de extracción de cláusulas. En la figura 5.5 se muestra una oración del corpus AnCora y en la figura 5.6 la salida correspondiente de Clatex, donde se resalta una de las cláusulas obtenidas. La misma se encuentra marcada en el corpus con la clase SUBJECT, mientras que el clasificador la marca con la clase ZERO, por lo que se computa como un error. Sin embargo, observando la cláusula aisladamente, es cierto que la misma no tiene sujeto. Su sujeto, *"la tributación"*, se encuentra presente pero se pierde al extraer la cláusula.

Presencia de sujeto según el parser	Tipo de sujeto anotado	Tipo de sujeto predicho		
		SUBJECT	ZERO	IMPERSONAL
SI	SUBJECT	2214	58	0
SI	ZERO	96	303	2
SI	IMPERSONAL	6	7	8
NO	SUBJECT	1220	179	2
NO	ZERO	270	872	7
NO	IMPERSONAL	17	8	24

Cuadro 5.6: Matrices de Confusión para los distintos resultados de la feature "Parser"

sujeto.pdf - Último acceso 2018/11/25

Ocurre "que"	Tipo de sujeto anotado	Tipo de sujeto predicho		
		SUBJECT	ZERO	IMPERSONAL
SI	SUBJECT	2335	195	2
SI	ZERO	259	658	6
SI	IMPERSONAL	13	10	16
NO	SUBJECT	1099	42	0
NO	ZERO	107	517	3
NO	IMPERSONAL	10	5	16

Cuadro 5.7: Matrices de Confusión para la ocurrencia del "que"

Por otra parte, observando las cláusulas de tipo implícitas clasificadas erróneamente se observó que una gran proporción de estas contienen la palabra "que". Este fenómeno es interesante, ya que la misma puede funcionar como sujeto en ciertas ocasiones, mientras que en muchas otras funciona como conjunción. Esto lleva a pensar que una clasificación más detallada de este caso puede llegar a mejorar el rendimiento de los clasificadores para las cláusulas con sujeto omitido.

Como se puede observar en el cuadro 5.7, existe una gran proporción de cláusulas que contienen la palabra "que" cuando las mismas se clasifican de manera incorrecta, especialmente cuando las mismas pertenecen a la clase ZERO y se clasifican como SUBJECT. Cabe notar que la cantidad de cláusulas que contienen la palabra "que" en el corpus de evaluación completo son el 66 %, lo que indica que realmente parece ser un factor que influye la clasificación.

Capítulo 6

Conclusiones y Trabajo Futuro

Al término de este proyecto, se concluyó con la construcción de una herramienta capaz de recibir oraciones en español y retornar las distintas cláusulas identificadas en la misma junto a la clasificación de su verbo principal, según el tipo de sujeto que presenta.

6.1. Conclusiones

Se cumplió con el objetivo de generar un clasificador capaz de identificar distintos tipos de sujetos en oraciones en español. El mismo obtiene buenos resultados a la hora de identificar cláusulas con sujeto explícito, así como aquellas con sujeto omitido. Se logró además alcanzar los resultados del estado del arte identificado, así como superar sus resultados con adiciones realizadas al modelo. Entre estas mejoras se encuentra la utilización de una herramienta interna para obtener cláusulas a partir de oraciones complejas, y la inclusión de features de nuevas áreas de interés, como Word2vec.

Un hito a marcar es la utilización de uno de los mayores corpus anotados sintácticamente para el español, AnCora, como base para el desarrollo del clasificador. Éste es considerablemente mayor al utilizado en la primer etapa del trabajo, y su utilización era uno de los objetivos planteados.

El clasificador construido utilizando la implementación VotingClassifier obtiene 0,833 de accuracy. Las medidas F para las clases SUBJECT, ZERO e IMPERSONAL son 0,891, 0,663 y 0,622 respectivamente.

Finalmente, se consolidaron los resultados obtenidos durante el desarrollo de este trabajo en una herramienta simple de utilizar, la cual recibe textos de largo arbitrario, y retorna como salida un listado de sus cláusulas, junto al verbo principal identificado y la clasificación de sujeto del mismo. La misma requiere algunas dependencias externas para realizar el procesamiento del texto a clasificar, pero es relativamente auto-contenida, ya que toda la información de los clasificadores se encuentra almacenada junto al módulo interactivo. En el apéndice A se detalla el proceso de instalación y uso de la herramienta construida.

6.2. Trabajo Futuro

Como primer objetivo, sería interesante utilizar los resultados de este trabajo para atacar el problema de resolución de co-referencias. Para esto se necesita agregar un paso sobre el comienzo del pipeline realizado, ya que es ineludible trabajar sobre un contexto mayor para lograr realizar esta tarea. Sobre este contexto extendido, ya sea sobre un documento entero, o uno o varios párrafos, se deben identificar los candidatos a ocupar el lugar del sujeto que fue omitido, por lo que el mismo debe coincidir tanto en número como en género. Una extensión deseable sobre este problema, y que va más allá de la detección de sujetos omitidos, es la resolución de pronombres, los cuales operan como sujetos para los propósitos de la herramienta generada, pero no permiten la identificación total del sujeto cuando se observan de manera aislada.

De las clases de cláusula sobre las que trabaja la herramienta realizada, los resultados más pobres se obtienen para las cláusulas impersonales. Una de las razones más claras para esta situación es la falta de suficientes datos en el corpus utilizado. Esto se refiere tanto a la falta de cantidad de cláusulas de este tipo, como la falta de variedad en la construcción de las mismas. Según se pudo identificar durante la etapa de análisis de errores, todas las cláusulas impersonales con las que se cuenta se construyen utilizando el participio “se”. Sería de gran utilidad enriquecer el corpus utilizado con otro tipo de construcciones impersonales, como por ejemplo oraciones sobre fenómenos climáticos, los cuales son inherentemente impersonales.

Es necesario validar de manera más exhaustiva el uso de la herramienta generada con otros tipos de texto. Como se observó en la sección 4.4.3, los resultados obtenidos sufrieron pérdidas de accuracy importantes al trabajar sobre un texto con una versión distinta del español. Es razonable pensar que también se generarán problemas al trabajar sobre textos de orígenes distintos, debido a que este desarrollo se realizó exclusivamente con textos de prensa. Particularmente serían problemáticos textos más informales, donde se utilizan contracciones, acrónimos, u otros tipos de jerga, sin mencionar errores ortográficos, que dificultarían el trabajo del POS-tagger y el parser de dependencias.

Otras mejoras posibles incluyen la incorporación de características de índole lingüística, que permitan explotar aspectos formales del idioma para mejorar los resultados de esta herramienta. Como se vio en la sección 5.2.2, la palabra "que" demostró ser un gran obstáculo a la hora de clasificar cláusulas de tipos SUBJECT y ZERO. Una de las razones para esto es la variedad de funciones que la misma puede cumplir, por lo que probablemente un análisis más formal sea de gran ayuda.

Finalmente, existen infinidad de mejoras que pueden realizarse a la herramienta generada, dado que la misma no es particularmente agradable para el usuario. Además, sería muy útil ofrecer la misma como herramienta online, al estilo de la ofrecida por FreeLing.[1]

[1] http://nlp.lsi.upc.edu/freeling/demo/demo.php - Último acceso 2018/11/25

Bibliografía

[1] A. Azzinnari y A. Martínez. «Representación de Palabras en Espacios de Vectores». Proyecto de grado. Universidad de la República, Uruguay. 2016.

[2] Steven Bird y Edward Loper. «NLTK: the natural language toolkit». En: *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics. 2004, pág. 31.

[3] Xavier Carreras, Isaac Chao, Lluís Padró y Muntsa Padró. «FreeLing: An Open-Source Suite of Language Analyzers». En: *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*. 2004.

[4] Mariela Grassi, Marisa Malcuori, Javier Couto, Juan José Prada y Dina Wonsever. «Corpus informatizado: textos del español del Uruguay (CORIN)». En: *SLPLT-2-Second International Workshop on Spanish Language Processing and Language Technologies-Jaén, España*. 2001.

[5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann y Ian H Witten. «The WEKA data mining software: an update». En: *ACM SIGKDD explorations newsletter* 11.1 (2009), págs. 10-18.

[6] Dan Jurafsky y James H Martin. *Speech and language processing*. Vol. 3. Pearson London, 2014.

[7] Mateusz Kopeć. «Zero subject detection for Polish». En: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. 2014, págs. 221-225.

[8] Kurosawa Hideto Kudo Taku Ichikawa Susumu. «Language independent null subject prediction for statistical machine translation». En: *Proceedings of the The 21st Annual Convention of the Society of Language Processing*. 2015, págs. 812-815.

[9] Marina Lloberes, Irene Castellón y Lluís Padró. «Spanish FreeLing Dependency Grammar». En: *Proceedings of 7th Language Resources and Evaluation Conference (LREC'10)*. La Valletta, Malta, mayo de 2010.

[10] Marta Luján. «Expresión y omisión del pronombre personal». En: *Gramática descriptiva de la lengua española*. Espasa Calpe. 1999, págs. 1275-1316.

[11] Pedro Pablo Devís Marquez. «El Parámetro del Sujeto Nulo y la enseñanza del español como lengua extranjera. Reflexión gramatical». En: *Didáctica. Lengua y Literatura* 23 (2011), págs. 59-86.

[12] Wes McKinney y col. «Data structures for statistical computing in python». En: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, págs. 51-56.

[13] Tomas Mikolov, Kai Chen, Greg Corrado y Jeffrey Dean. «Efficient Estimation of Word Representations in Vector Space». En: *CoRR* abs/1301.3781 (2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[14] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[15] Lluís Padró y Evgeny Stanilovsky. «FreeLing 3.0: Towards Wider Multilinguality». En: *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. ELRA. Istanbul, Turkey, mayo de 2012.

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg y col. «Scikit-learn: Machine learning in Python». En: *Journal of machine learning research* 12.Oct (2011), págs. 2825-2830.

[17] Luz Rello, Ricardo Baeza-Yates y Ruslan Mitkov. «Elliphant: Improved automatic detection of zero subjects and impersonal constructions in spanish». En: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2012, págs. 706-715.

[18] Olga M Fernández Soriano. «El pronombre personal. Formas y distribuciones. Pronombres átonos y tónicos». En: *Gramática descriptiva de la lengua española*. Espasa Calpe. 1999, págs. 1209-1274.

[19] Mariona Taulé, Maria Antònia Martı y Marta Recasens. «AnCora: Multilevel Annotated Corpora for Catalan and Spanish.» En: *Lrec*. 2008.

[20] Dina Wonsever, Serrana Caviglia, Javier Couto y Aiala Rosá. «Un sistema para la segmentación en proposiciones de textos en español». En: *Letras de Hoje. Porto Alegre.* 41.2 (2006), 89-101. URL: http://halshs.archives-ouvertes.fr/halshs-00158851.

[21] Alina Wróblewska y Marcin Woliński. «Preliminary experiments in Polish dependency parsing». En: *Security and Intelligent Information Systems*. Springer, 2012, págs. 279-292.

Glosario

Clítico Elemento gramatical, típicamente un pronombre, que aparece ligado a un verbo. Puede representar al objeto como en el siguiente ejemplo:

- María **lo** fue a ver
- María fue a ver**lo**

O puede ser un clítico indirecto:

- María no **se lo** quiso decir
- María no quiso decir**selo**

.

Feature Una feature es una característica medible de un objeto. En el contexto del aprendizaje automático se utilizan como entrada para los clasificadores utilizados.

Grupo verbal (sintagma verbal) Grupo de palabras cuyo núcleo es un verbo. Estos pueden ser verbos junto a sus complementos, como por ejemplo: *Mucha gente [vio la película]*$_{SV}$.

Lengua pro-drop Del inglés pronoun-dropping, es decir, que omite un pronombre, son aquellos lenguajes que permiten omitir un pronombre siempre y cuando él mismo sea inferible.

Lenguas balto-eslavas Lenguas habladas en las regiones bálticas y eslavas de Europa.

Lenguas romances Grupo de lenguas derivadas del latín común hablado en territorios del antiguo Império Romano. Incluye el español (castellano), portugues, italiano, etc.

Noun Phrase Ver *Sintagma Nominal*.

NSL (Null Subject Languages) Idiomas, como el español, en las cuales es posible construir oraciones donde el sujeto de las mismas esta implícito.

Parser Conocido también como analizador sintáctico, toma un texto de entrada, y según reglas gramatícales dadas, lo analiza y genera un árbol sintáctico como salida.

Parser de dependencias Analizador, el cual toma una oración o frase y utilizando gramáticas de dependencias, genera una relación entre cada elemento de la entrada.

PoS Tagger Tiene como fin el asignar a cada elemento de la entrada, una categoría gramatical, teniendo en cuenta la definición del elemento o su contexto en la entrada.

Sintagma adjetival Grupo de palabras cuyo núcleo es un adjetivo, como por ejemplo: *El equipo se sentía [muy decepcionado con el resultado]*$_{SA}$.

Sintagma Nominal Palabra o grupo de palabras que cuyo núcleo es un sustantivo y su rol en la oración puede ser de sujeto, complemento directo o indirecto, etc.

Apéndice A

Manual de uso de la herramienta

A.1. Instalación

Antes de comenzar se debe asegurar de tener todas las dependencias necesarias para una instalación limpia de la herramienta en un ambiente Unix.

El siguiente comando instala todas las dependencias de sistema:

```
sudo apt install python-pip python3 python3-dev libboost-dev foma-bin
foma libfoma-dev foma-bin build-essen tial automake autoconf libtool
git libboost-regex-dev libboost-program-options libboost-program-options-dev
libboost-system-dev libboost-thread-dev git
```

A.1.1. Instalación de FreeLing

La instalación se realiza a partir del código fuente[1] de la herramienta FreeLing.

```
git clone https://github.com/TALP-UPC/FreeLing.git
```

Se utilizó una versión no etiquetada del código, la más actual al momento de su instalación, para la cual se probó el correcto funcionamiento de la herramienta. Para utilizar la misma versión, es necesario moverse a un punto específico del código.

[1] https://github.com/TALP-UPC/FreeLing/ - Último acceso 2018/11/25

```
git checkout ff81c77efea3130d2b0a4dda603398da0c65dd96
```

Una vez que se cuenta con el código, es necesario compilarlo. para esto se siguen las instrucciones indicadas en la descripción de FreeLing[2].

A.1.2. Entorno de desarrollo

Para simplificar la instalación y manejo de dependencias se utilizó un entorno virtual de Python[3]. La herramienta requiere la utilización de Python3, por lo que el entorno debe ser creado con una versión igual o superior a la misma.

```
mkvirtualenv pgrado --python=/usr/bin/python3.5
```

Una vez creado el entorno virtual, se instalan las dependecias las dependencias locales, incluidas en el archivo `requirements.txt`

```
pip install -r requirements.txt
```

Finalmente, es necesario declarar la ubicación de la API Python del modulo FreeLing. El mismo se encuentra en la ruta `APIs/python` relativa al directorio donde se clonó el repositorio de la herramienta. Para esto se utilizan la variable de ambiente `FREELING_API`, declarada de la siguiente manera:

```
export FREELING_API=/home/usuario/FreeLing/APIs/python
```

A.2. Utilización del modulo

El primer paso para utilizar la herramienta es activar el entorno virtual creado, luego simplemente se ejecuta el módulo principal de la herramienta.

```
workon pgrado
python pgrado
```

[2]https://talp-upc.gitbook.io/freeling-4-0-user-manual/installation#install-from-github-repositories - Último acceso 2018/11/25

[3]https://virtualenvwrapper.readthedocs.io/en/latest/install.html#basic-installation - Último acceso 2018/11/25

Una vez dentro del programa se presenta un texto de ayuda y se habilita el input del usuario. Los comandos disponibles que posee la herramienta son: `clasificador`, `clasificar`, `help` y `exit`.

El comando principal es `clasificar`, el cual como dice su nombre es el que permite clasificar un fragmento de texto. Una vez ingresado ese comando, sin argumentos, se habilita el input para ingresar texto a clasificar. Es conveniente copiar el mismo y pegarlo directamente, ya que la interfaz no permite navegar con el cursor hacia atrás. La herramienta es capaz de clasificar multiples oraciones, es por esta razón que es necesario indicarle cuando se ha terminado de ingresar texto y se desea comenzar con la clasificación, para esto se utiliza el caracter `#` seguido de un salto de línea. Luego, la herramienta imprime el listado de cláusulas extraídas con el verbo principal identificado y la predicción del clasificador seleccionado.

Es posible cambiar el clasificador a utilizar con el comando
`clasificador <nombre_clasificador>`
donde `<nombre_clasificador>` es uno de los ofrecidos en la lista que se muestra al iniciar el programa o al utilizar el comando `help`.

Apéndice B

Listado de verbos impersonales

A continuación se presenta el listado de verbos utilizados para la implementación de la feature *VERB_TYPE (impersonal)*. La misma se obtuvo de Wiktionary[1]

- abocanar
- acaecer
- acantalear
- acontecer
- acuntir
- alborecer
- algaracear
- amanecer
- anochecer
- anochecerse
- antojarse
- apedrear
- asolarse
- atardecer
- atañer
- bastar
- brisar
- cascarrinar
- cellisquear
- cercear
- chaparrear
- chispear
- clarecer
- concernir
- diluviar
- empecer
- encelajarse
- esclarecer
- garuar
- granizar
- haber
- hacer
- harinear
- helar
- hojecer
- llover
- lloviznar
- mollinear
- molliznar

[1]https://es.wiktionary.org/wiki/Categor%C3%ADa:ES:Verbos_impersonales - Último acceso 2018/11/21

- molliznear
- neblinear
- nevar
- neviscar
- obstar
- ocurrirse
- orbayar
- orvallar
- oscurecer
- paramar
- paramear
- parecer
- pintear
- poder
- podsolizar
- podsolizarse
- podzolizar
- podzolizarse
- querer
- rocear
- rosar
- tardecer
- ventiscar
- ventisquear
- zaracear