Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

PROYECTO DE GRADO
INGENIERÍA EN
COMPUTACIÓN

Estudio de reputación a partir de comentarios extraídos de redes sociales

Tutor del proyecto:
Aiala Rosá, Universidad de la República

Martín Mori
Martín Tambucho
Diego Cardozo

Resumen

En los últimos años, el análisis de sentimiento (área de investigación del Procesamiento del Lenguaje Natural) y las técnicas de aprendizaje automático han cobrado una gran importancia a nivel académico y comercial. Esto ha sido fundamentalmente impulsado por la gran cantidad de información y datos que pueden ser extraídos desde las redes sociales.

En el presente proyecto se implementa un sistema de análisis de sentimiento para realizar un estudio de reputación a partir de comentarios extraídos de la red social Twitter. Para realizar esta tarea primero se construye un corpus anotado de tweets escritos en Uruguay; luego de que este corpus es confeccionado, un conjunto de usuarios procede a anotar manualmente los tweets, determinando si cada uno se trata o no de una opinión. En caso afirmativo, procede a clasificarla como positiva, negativa o neutral. Mediante un proceso de aprendizaje sobre los datos generados por los anotadores, un conjunto de algoritmos son entrenados para clasificar tweets automáticamente. Este conjunto se compone de tres algoritmos basados en reglas y seis clasificadores basados en los siguientes métodos de aprendizaje automático: Naive Bayes, árboles de decisión, SVM, KNN, regresión logística y redes neuronales. Naive Bayes y redes neuronales obtienen los mejores resultados con una medida F1 de 68% y 69% respectivamente.

Finalmente, se utiliza el clasificador de redes neuronales ya entrenado para realizar un estudio de la reputación de un conjunto de entidades o temas destacados. Para ello se descargan los tweets de dichas entidades y se procede a ejecutar el clasificador, graficando la evolución de la reputación de las mismas en el tiempo. Las entidades elegidas fueron: el deportista Emiliano Lasa en el contexto de las instancias finales de una competencia de salto largo; Maria Noel Riccetto, ganadora de un importante premio de Ballet; Verónica Alonso y Raúl Sendic en un período de tiempo en el cual fueron implicados en distintos escándalos políticos. Adicionalmente, se realizó el análisis de reputación de la Selección Uruguaya de Fútbol sub 20 cuando la misma fuera eliminada del mundial de fútbol. Todos los resultados obtenidos fueron coherentes con el contexto particular de cada caso.

Por otro lado, durante el desarrollo del proyecto se utilizaron dos lexicones: el primero de palabras con sentimientos marcados, el cual fue expandido con jerga uruguaya y el segundo con patrones de frases que asisten en el proceso para determinar si un tweet es una opinión. Los mencionados recursos, junto con el corpus anotado y los programas implementados podrán ser reutilizados en futuros proyectos.

Índice

Capítulo 1

Introducción

El análisis de sentimiento es un área de investigación del Procesamiento del Lenguaje Natural que cuenta con más de 15 años de desarrollo. En los últimos años está cobrando una gran importancia debido fundamentalmente a la gran cantidad de comentarios que se escriben en Internet por parte de millones de usuarios de todo el mundo a través de blogs, foros o redes sociales. Aunque existen ya varios trabajos relacionados con la temática, la gran mayoría de ellos utilizan únicamente textos en inglés.

Uno de los casos de interés de esta área es conocer lo que opinan los usuarios de las redes sociales sobre productos, servicios, marcas, personas e instituciones. El método tradicional para conocer la opinión de los usuarios son las encuestas y los estudios de mercado.

La diferencia más grande entre información recabada en encuestas y estudios de mercado a la obtenida en las redes sociales es su inmediatez, aunque esta información es espontánea y poco estructurada, y refleja en muchos casos lo que opinan los usuarios, no los expertos. En muchas ocasiones, la red se utiliza para difundir quejas o recomendaciones que llegan a afectar directamente a la imagen corporativa. Las compañías y agencias de comunicación entienden el valor de esta información para sus estrategias comerciales, la atención al cliente y la detección de tendencias. Además, las organizaciones que contratan empresas de comunicación quieren conocer el impacto social de sus noticias y campañas para valorar la estrategia comercial. [1]

Por otra parte la popularidad de las redes sociales en los últimos años se ha ido incrementando de forma excepcional. Sitios web como Facebook, Youtube, Twitter, LinkedIn e Instagram permiten a los usuarios comunicarse y compartir fotos, videos, comentarios e información personal con potencialmente miles de personas.

En particular Twitter permite enviar mensajes de texto plano de corta longitud, con un máximo de 140 caracteres, llamados tweets, que se muestran en la página principal del usuario. Los usuarios pueden suscribirse para visualizar los tweets de otros usuarios – a esto se le llama "seguir" y a los usuarios suscritos se les llama "seguidores". Por defecto, los mensajes son públicos. También existe la posibilidad de que los mensajes sean visibles únicamente por los seguidores del autor. Los usuarios pueden twittear desde la web del servicio o con aplicaciones móviles para teléfonos inteligentes o mediante mensajes de texto[2].

La clasificación, comprensión y evaluación de opiniones y comentarios publicados en Internet es una tarea difícil porque incluso los seres humanos a menudo están en desacuerdo sobre la polaridad de un texto dado. Y es una tarea más difícil aún cuando el texto sólo tiene 140 caracteres.

El campo del análisis de sentimiento, pese a que ha experimentado un importante crecimiento en los últimos años debido a sus variadas aplicaciones y el interés tanto académico como privado, se mantiene como un problema desafiante y aún en pleno desarrollo.

1.1 Motivación

Como se menciona en la introducción, el análisis de sentimiento y aprendizaje automático se encuentra en pleno crecimiento. Esto, combinado con la inmediatez de las redes sociales y la cantidad de información que brindan, crean un escenario muy prometedor para la investigación y creación de este tipo de herramientas aplicables en muchos aspectos de la vida humana. Sin embargo es evidente que existe una carencia de herramientas desarrolladas para el idioma español con este fin. Es por esto que una de nuestras motivaciones es experimentar con el análisis de sentimiento en un entorno local y contribuir a la comunidad con esta investigación.

La inmensa cantidad de comentarios, opiniones y otros tipos de información que pueden ser recuperados de las redes sociales resulta sumamente alentador para la experimentación con aplicaciones de aprendizaje automático; ya sea para realizar estudios de popularidad, o analizar el sentimiento del público general hacia una persona o tema. Este tipo de herramientas puede resultar clave para empresas publicitarias, candidatos a una elección, etc.

1.2 Objetivos

En este trabajo investigaremos e implementaremos un sistema de análisis de sentimiento para realizar un estudio de reputación a partir de comentarios extraídos de la red social Twitter.

Esta tarea se subdivide en 3 objetivos:

1. Construir un corpus anotado de tweets escritos en Uruguay. Los anotadores manualmente determinarán si un tweet es una opinión o no, y en caso afirmativo determinarán si la misma es positiva, negativa o neutral. De esta manera, las etiquetas posibles para los tweets del corpus son: “no es opinión”, “opinión positiva”, “opinión neutral” u “opinión negativa”.
2. Entrenar uno o varios clasificadores que permitan predecir la etiqueta de nuevos tweets no clasificados, basándose en la información proporcionada por los tweets existentes en el corpus. Se experimentará con clasificadores basados en reglas y clasificadores basados en aprendizaje automático.
3. Estudiar el análisis de reputación de una persona o tema en el tiempo. Para ello se descargan todos los tweets uruguayos disponibles que tratan sobre dicha persona o tema, y se utiliza el clasificador ya entrenado con mejores niveles de performance para predecir la clasificación de cada uno de ellos. Finalmente, se grafica cómo evoluciona la reputación de la persona o tema según el balance de tweets positivos y negativos de cada día.

1.3 Análisis del problema

Esta sección presenta una breve descripción del problema abordado en el marco de este proyecto. Consiste en aplicar los conocimientos del área del análisis de sentimiento a la creación de un programa que permita detectar automáticamente la polaridad de un conjunto de opiniones. Luego, este programa es utilizado en el estudio de reputación de un grupo de temas o personas durante un periodo de tiempo.

A continuación mencionaremos algunas observaciones obtenidas a partir del estudio del corpus ya que resultan relevantes para comprender aspectos claves sobre la dificultad de la resolución del problema. Más adelante, en la sección 3.1 (construcción del corpus) brindaremos más detalles sobre la construcción del corpus.

La naturaleza de red social de Twitter junto con la limitación de 140 caracteres por texto, ocasiona que la mayoría de los tweets sean escritos de forma informal. Es decir: con faltas de ortografía, errores gramaticales o abreviaciones como se muestra en la imagen 1.1. Algunas excepciones son los textos provenientes de periodistas o comunicadores como se puede ver en la imagen 1.2.

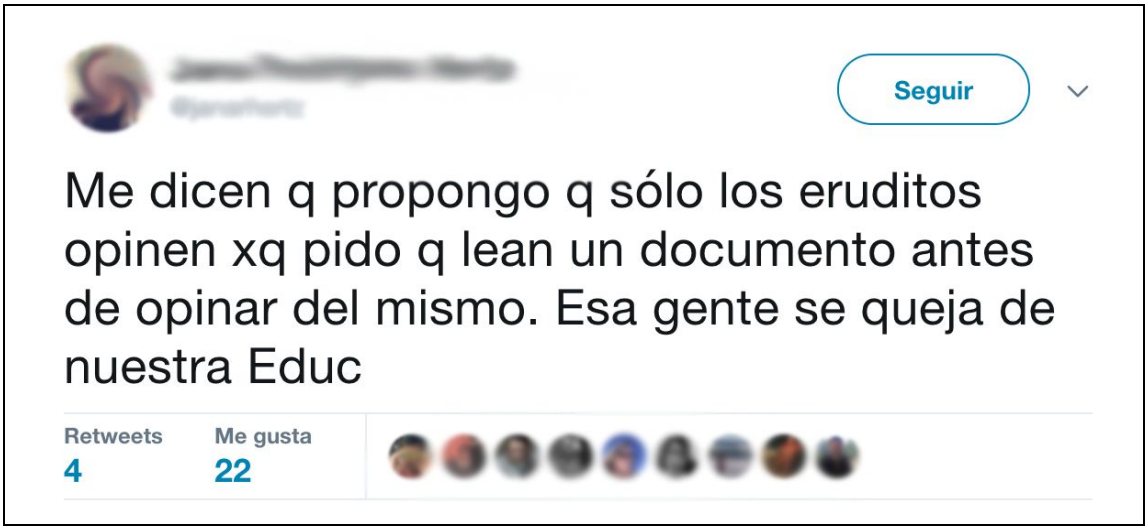


Imagen 1.1- Ejemplo de tweet que contiene abreviaciones

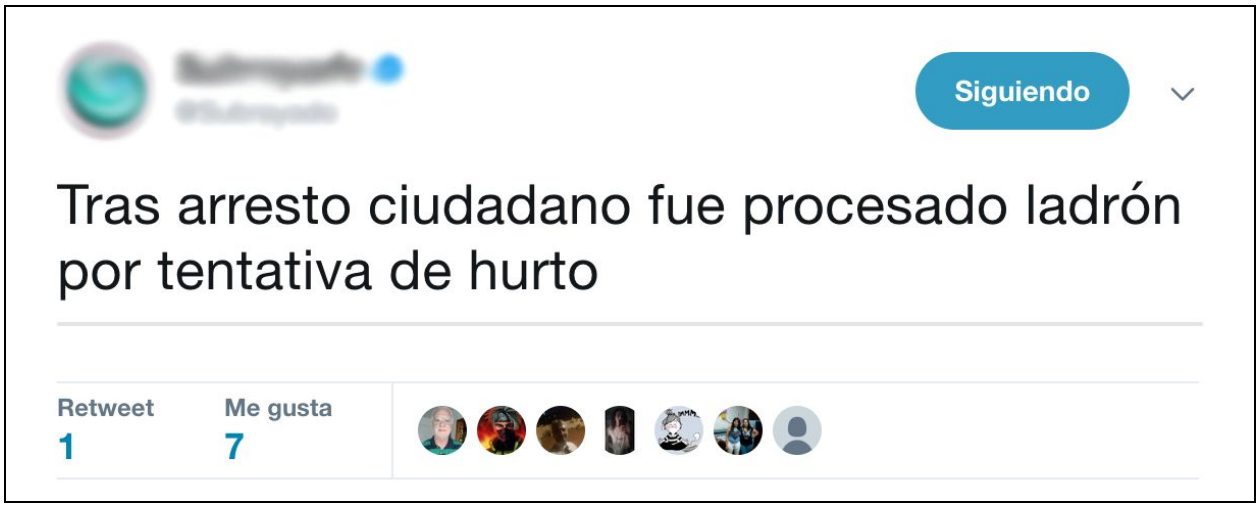


Imagen 1.2- Ejemplo de un tweet que no es opinión, escrito formalmente.

Otro problema a tener en cuenta es que las personas hablan y escriben de distinta maneras dependiendo del país o región en que se encuentren, aún dentro del idioma español. Por esta razón decidimos adaptar un diccionario de palabras subjetivas ya existente, incorporando expresiones utilizadas por los uruguayos. Esto se explicará en detalle en la sección 3.2.1.

En el ejemplo que muestra la imagen 1.3, es claro ver que el sentimiento del texto es una opinión positiva, y que tanto la expresión “mucho huevo” como la palabra “bestia” tienen connotación positiva. Si este texto fuera presentado en otro país probablemente no tendría sentido.

Imagen 1.3- Ejemplo de un tweet con expresiones locales.

Como muestra la imagen 1.4, otra característica que tiene Twitter es que no solamente es posible compartir texto plano, sino que es posible compartir links, videos, noticias y otros tipos de contenido multimedia. Es necesario preprocesar y adecuar este contenido para posibilitar el proceso de entrenamiento.

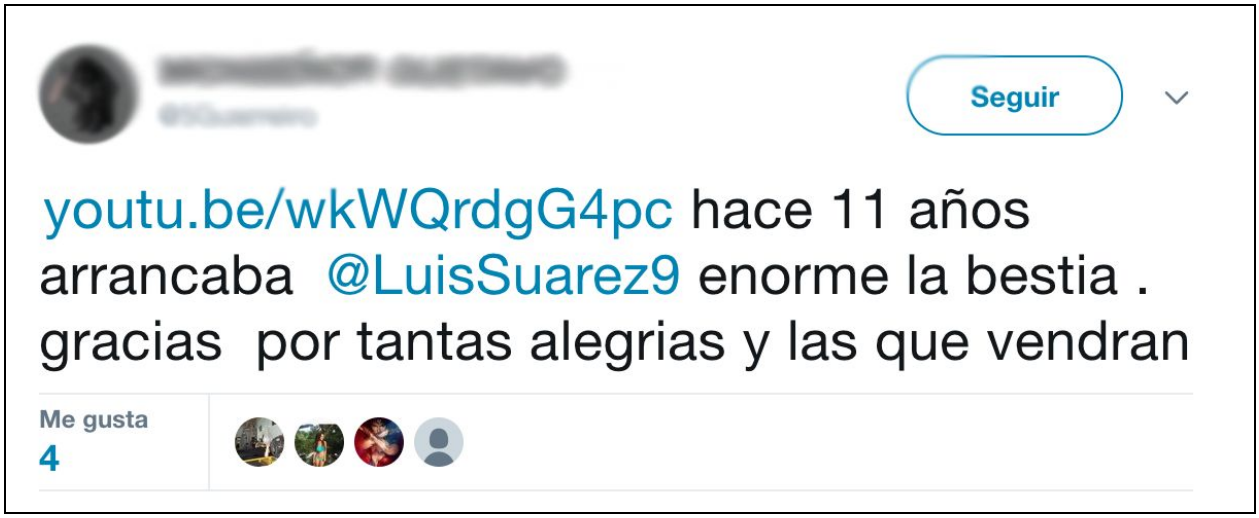


Imagen 1.4- Ejemplo de un tweet que contiene un link

Por otra parte, uno de los problemas más grandes en el contexto del procesamiento de lenguaje natural es detectar y manejar ironías. Incluso para las personas es difícil distinguir si un texto es irónico o no sin contar con suficiente contexto. En el ejemplo de la imagen 1.5, sin conocer el contexto en que fue escrito el tweet, diríamos que es claramente una opinión negativa (aún si conocemos de quién se está hablando). Sin embargo este tweet fue escrito en un partido donde Messi marcó tres goles. Este dato del contexto nos da suficiente información para detectar que el tweet es una ironía y que en realidad se trata de una opinión positiva.

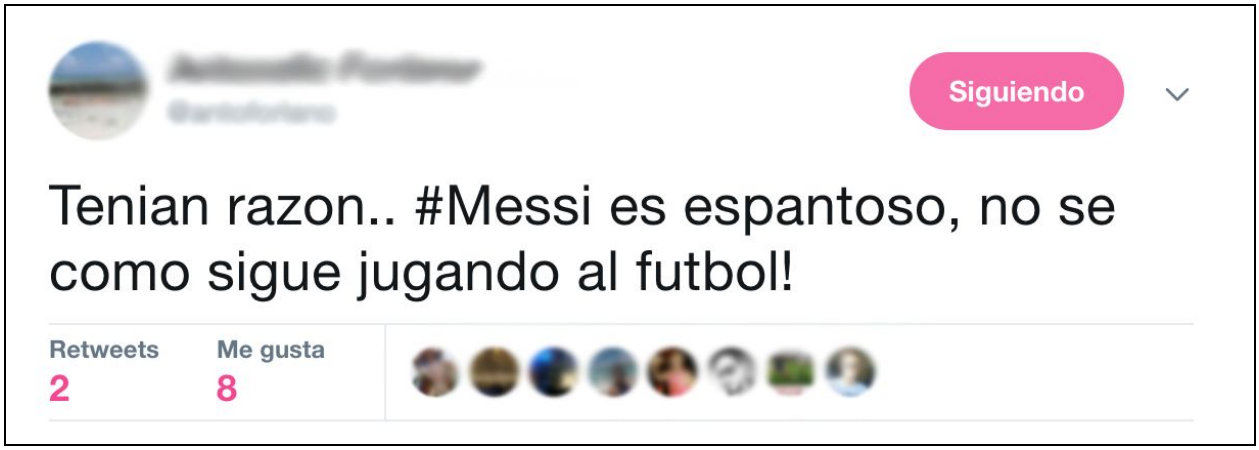


Imagen 1.5- Ejemplo de un tweet que contiene una ironía

Difícilmente un clasificador automático contará con suficiente información acerca del contexto en que se escribió un tweet como para detectar la ironía. Observando que los textos sólo contienen 140 caracteres, queda muy poco lugar para aportar información adicional sobre lo que se está escribiendo. Esto dificulta aún más la tarea de aprendizaje de un clasificador.

Para comprender el problema, como paso previo al desarrollo de una solución es necesario analizar las características de los tweets con distintas polaridades y estudiar en cada caso qué patrones tienen en común y qué los diferencia. Si bien muchas veces el sentimiento de una opinión se encuentra fuertemente marcado, nos encontramos con el problema que hay veces donde este sentimiento no es tan explícito. Parte de un texto puede expresar un sentimiento positivo mientras que otra parte puede expresar uno negativo. En estos casos es importante que el clasificador aprenda a diferenciar cuál de estos sentimientos acarrea mayor peso y cual es el sentimiento que el autor del tweet quiso transmitir.

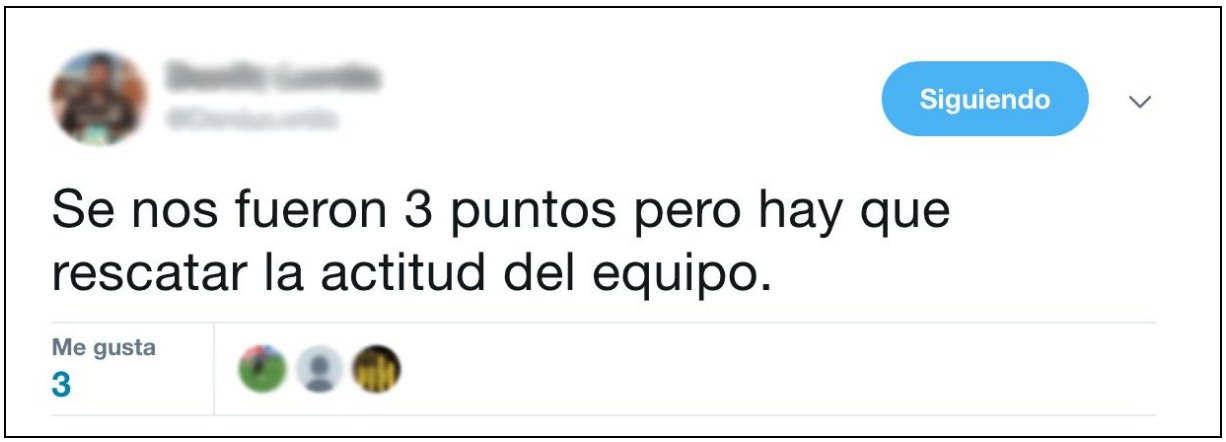


Imagen 1.6 - Ejemplo de una opinión positiva, con sentimientos positivos y negativos en el texto.

Finalmente, durante el análisis previo, notamos que no encontramos una gran cantidad de opiniones neutrales. Esto puede deberse a que generalmente cuando las personas invierten tiempo en dar una opinión es para expresar conformidad o disconformidad con alguna situación o persona. Rara vez una

opinión no incluye algún tipo de sentimiento. Además, es extremadamente complejo, incluso para una persona, diferenciar una opinión neutral de algo que no es opinión. Quizá por esta razón, como se verá en la sección Marco de trabajo, la mayoría de los trabajos relacionados a esta temática descartan esta categoría o tratan de incluirla dentro de otras. Más adelante explicaremos nuestro enfoque en el tratamiento de este tipo de opiniones en el desarrollo de nuestra solución.

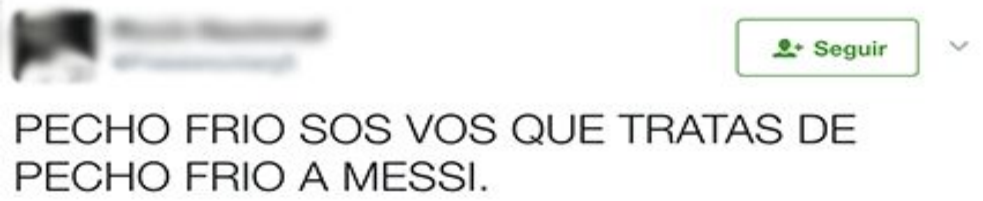
Tweet no opinión	
Tweet positivo	
Tweet neutral	
Tweet negativo	

Tabla 1: Ejemplos de clasificación de sentimiento

1.4 Estructura del documento

En el Capítulo 2 se brinda una definición formal del análisis de sentimiento, se detallan las característica de los tweets y se hace una descripción de los clasificadores basados reglas y basados aprendizaje automático. Adicionalmente, se muestra una breve reseña de los trabajos relacionados.

En el Capítulo 3 se resumen los recursos generados que pueden ser reutilizables en otros proyectos a futuro, la construcción del corpus y las dificultades que surgieron del proceso de construcción del mismo. Se enumeran los criterios de anotación y se realiza un estudio realizado sobre el corpus y las métricas de concordancia. Adicionalmente, se muestran los lexicones utilizados a lo largo del proyecto, mostrando las mejoras realizadas en los mismos.

Luego en el Capítulo 4 se describe como se realizo la solución del problema. Se detalla el preprocesamiento realizado sobre el corpus, el cual incluye la limpieza de etiquetas de Twitter, normalización de emoticones y gramática; se muestran los detalles de la línea base utilizada y los resultados obtenidos para la misma y se detalla el uso de Freeling. Adicionalmente, detallamos todos los clasificadores utilizados (basados en reglas gramaticales y basados en clasificadores de aprendizaje automático), se detallan los features utilizados y los clasificadores con sus respectivos resultados.

En el Capítulo 5 se muestra cómo se implementó el análisis de reputación, mostrando algunos ejemplos. Finalmente, en el Capítulo 6 se presentan las conclusiones del proyecto y se detallan las principales líneas de trabajo a futuro identificadas durante la elaboración del proyecto.

Capítulo 2

Marco de trabajo

En este capítulo se define y explica el análisis de sentimiento, analizando los conceptos sobre los que se construye y sobre los cuales trabajaremos a lo largo de esta investigación. Adicionalmente, se describen algunas características interesantes de los tweets, se realiza una descripción de los clasificadores utilizados y por último se describen los principales trabajos relacionados al desafío abordado en este proyecto.

2.1 Análisis de sentimientos

El análisis de sentimiento (también conocido como minería de opinión) se refiere al uso de procesamiento de lenguaje natural, análisis de texto y lingüística computacional para identificar y extraer información subjetiva de ciertos recursos. [3]

En términos generales, el análisis de sentimiento intenta determinar la actitud de un interlocutor o un escritor con respecto a algún tema o la polaridad contextual general de un documento. La actitud puede ser su juicio o evaluación, estado afectivo (o sea, el estado emocional del autor al momento de escribir), o la intención comunicativa emocional (o sea, el efecto emocional que el autor intenta causar en el lector).

En nuestro caso, en primera instancia nos centramos en el análisis de polaridad dentro de las opiniones. Dado un tweet, este será clasificado dentro de las categorías “opinión positiva”, “ opinión negativa”, “opinión neutral” y “no es opinión”.

2.2 Características de los tweets

El tamaño máximo de un mensaje en twitter es de 140 caracteres. A partir del conjunto de tweets con el que trabajamos en este proyecto se calcula que el tamaño promedio de un tweet es de 14 palabras o 78 caracteres. Esto difiere con otros trabajos donde los textos son más extensos.

Además de escribir texto plano, hay muchas otras acciones que se pueden realizar en Twitter. Es por esto que cada tweet puede contener algunas etiquetas, cada una de ellas indicando la acción que se está realizando.

Por ejemplo para referirse a otro usuario en el texto del tweet se utiliza una mención, la cual se forma a partir del nombre de usuario de la persona que se quiere mencionar precedido de un arroba (@). [4]

Un elemento característico y sumamente utilizado es el hashtag, es una palabra o serie de palabras o caracteres alfanuméricos precedidos por el numeral (#). Un hashtag permite organizar, clasificar o agrupar las publicaciones de acuerdo a su contenido. Facilita el intercambio de información o contenidos entre distintos usuarios sobre determinados temas o acontecimientos. [5]

Otra funcionalidad especial de Twitter es la capacidad de emitir respuestas a un tweet. Esto permite realizar referencias a los tweets de un usuario, agregando un comentario. Al realizar una respuesta, se agrega automáticamente una mención al autor del tweet original (o los autores en caso de tratarse de una respuesta a otra respuesta).

Por otro lado, para hacer referencia al tweet de otro usuario se puede realizar un retweet. Un retweet permite a los usuarios compartir tweets realizados por otros usuarios. Es algo así como mencionar lo que otra persona ha dicho, pudiendo agregar o no, un comentario al respecto. Dentro del tweet se representa como las letras “RT” seguido de una mención al usuario que realizó el tweet. [6]

El siguiente tweet de ejemplo es un retweet cuyo usuario original es “veroamorelli”, en el cual se hace mención al usuario de la radio Carve850 y hace referencia a un tema, “ExpoPrado” :

RT @veroamorelli: Al aire ahora en los estudios de @carve850 en la #ExpoPrado.

Adicionalmente, a mediados del 2014 Twitter añadió nuevas herramientas para el texto como: ‘los emojis’ (ver sección 4.1.2) , con lo que se ha creado un gran cambio en esta red social. [7]

Finalmente, es interesante observar que la frecuencia de errores de ortografía, lunfardos y abreviaciones de palabras es mucho mayor que en otros dominio. Esto se debe a que los usuarios de twitter redactan mensajes desde diferentes medios incluidos celulares, y las abreviaciones son una herramienta que permite evitar en parte la limitación de caracteres.

2.3 Trabajos relacionados

En esta sección se presenta una serie de conclusiones obtenidas de artículos que consideramos relevantes para comprender el contexto académico internacional que rodea al proyecto.

Los artículos se presentan en orden cronológico con el objetivo de apreciar la evolución significativa de la temática en la academia en la última década.

2.3.1 Clasificación de sentimientos de Twitter utilizando supervisión a distancia

Este paper de la universidad de Stanford es uno de los primeros papers publicados que trabajan sobre el análisis de sentimiento aplicado directamente a Twitter, para tweets en inglés. [8]

Los autores trabajan sobre clasificación de opiniones estrictamente positivas o negativas, excluyendo los tweets neutrales antes de comenzar el proceso de aprendizaje. En este sentido, el primer aporte de interés proviene de la aplicación de un criterio específico que determina si un tweet es neutral o no. Esto es: si el tweet puede aparecer como título de una noticia periodística o como una oración en Wikipedia, entonces se clasifica como “neutral”.

En este artículo se pueden apreciar los primeros resultados al utilizar los clasificadores clásicos: Naive Bayes, Maximum Entropy (MaxEnt), y Support Vector Machines (SVM). Se obtiene una precisión de alrededor de 80%, y una de las principales innovaciones de este artículo es el uso de los emoticones cómo features.

2.3.2 Análisis de sentimientos de datos de Twitter

En este artículo se experimentó con 2 clasificadores. Uno binario que clasifica tweets como positivos o negativos, y un clasificador multiclase que clasifica a un tweet como positivo, negativo o neutral. Con un corpus de 5127 tweets en inglés clasificados manualmente, cabe destacar que se realizó un pre procesamiento donde se etiquetaban tweets como “junk” (basura) y luego se descartaron si no podían ser clasificados por un humano (links, imágenes, etc). Además, se experimentó con 3 modelos: unigramas, un modelo basado en features y un clasificador arborescente (tree kernel). [9]

Como línea base se utiliza un clasificador en base a unigramas, el cual según el artículo ha demostrado funcionar bien para el análisis de sentimiento en tweets, obteniendo resultados un 20% mejores que un clasificador aleatorio para ambas tareas de clasificación.

El clasificador basado en features con 100 features obtuvo resultados equivalentes al de unigramas (el cual contaba con más de 10000 features). El clasificador basado en árboles supera los resultados de ambos clasificadores significativamente.

Adicionalmente, se experimentó con combinaciones de unigramas con el modelo de features, y del modelo de features con el modelo basado en árboles. Estas combinaciones superaron la línea base en poco más de 4%.

En el artículo se obtienen varios resultados sumamente interesantes para la selección de features en nuestro proyecto, ya que se mostró que los features que son específicos de Twitter (emoticones, hashtags, etc) agregan valor al clasificador pero solamente de forma marginal. Los features que combinan la polaridad de las palabras según un lexicón subjetivo con sus etiquetas POS son los más importantes para ambas tareas de clasificación.

Adicionalmente, se introducen 2 recursos sumamente valiosos. El primero es un diccionario manualmente anotado de emoticones con sus polaridades respectivas, y el segundo es un diccionario de acrónimos en inglés con sus traducciones (no puede utilizarse directamente en nuestro proyecto por la diferencia de idioma, pero el concepto de un diccionario de acrónimos puede resultar útil).

El mejor resultado obtenido para el clasificador binario fue obtenido con el clasificador basado en árboles, con una precisión de alrededor de un 73% al clasificar entre positivo y negativo, y 60% al clasificar entre positivo, negativo o neutral.

2.3.3 Técnicas de clasificación de opiniones aplicadas a un corpus en español

En este trabajo de la universidad de Jaén en España realizado en el año 2011 se presenta un estudio experimental sobre un corpus de opiniones sobre películas escrito en español compuesto de 3.878 críticas recogidas de la web "www.*muchocine.net*". Resulta de particular interés ya que, a diferencia de los artículos anteriores, se aplica directamente sobre el Español. [10]

Las críticas que componen el corpus no están escritas por profesionales, sino por usuarios de la web. Esto aumenta la dificultad de la tarea, ya que los textos pueden no ser gramaticalmente correctos, pudiendo aparecer faltas de ortografía o expresiones informales al igual que en Twitter.

Las críticas están puntuadas en un rango de 1 a 5, siendo el 1 una película muy mala, y el 5 una película muy buena. En caso de contar con una valoración inferior a 3 se considerarán como negativas, mientras que las valoradas con un 4 o un 5, se etiquetan como críticas positivas.

El número total de documentos sobre los que se han realizado los experimentos es de 2.625, de los cuales 1.274 se corresponden con críticas negativas, y 1.351 positivas. Estas cantidades fueron tomadas como base al momento de construir nuestro corpus (como se describe más adelante).

Se utilizan dos clasificadores: SVM y Naive Bayes, variando además distintos parámetros como el esquema de pesado o la utilización o no de stopper y stemmer. Los experimentos realizados muestran que SVM se comporta mejor que Naive Bayes y que el uso de stopper y stemmer también mejora los resultados.

Con los resultados obtenidos se puede concluir que la combinación de stopper, stemmer, TF-IDF y SVM es la que mejor rendimiento ofrece. En su mejor configuración, los autores presentan una precisión de alrededor de 87%, mejorando por más de 9 puntos porcentuales los resultados obtenidos por Cruz et al., 2008 [11].

Una primera observación es que la aplicación de los mismos clasificadores sobre textos de tamaño mayor parece aportar mayor precisión.

2.3.4 Determinación de la orientación semántica de las opiniones en textos de prensa

Este trabajo tiene como objetivos la investigación e implementación de un sistema de análisis de sentimiento, así como la integración con el ya existente sistema BuscOpiniones, y fue realizado en el marco de un Proyecto de grado de la FING del año 2015. [12]

Se busca crear un sistema que dada una opinión determine si es positiva, neutral o negativa, con un nivel de precisión cercano al logrado en el estado del arte del área. Finalmente se integra el algoritmo generado a BuscOpiniones. De esta forma, es posible visualizar en el programa el resultado del análisis y también incluir el sentimiento en las posibles claves de búsqueda.

Se creó un corpus anotado de opiniones de prensa uruguaya, se obtuvieron listas de palabras con su polaridad semántica y se las adaptó a la variante del español utilizado en Uruguay.

Se crearon tres clasificadores: uno con el enfoque de reglas que trabaja sobre la salida de un parser morfo-sintáctico del idioma español llamado MateParser, un clasificador exclusivamente basado en aprendizaje automático y uno con un enfoque híbrido que combina los anteriores. Este último enfoque generó los mejores resultados ya que se logró tomar lo mejor de cada uno de los métodos anteriores, alimentando al algoritmo de aprendizaje automático con la información obtenida mediante las reglas.

Los resultados obtenidos fueron de un 64% de precisión total y en promedio entre las tres clases.

2.3.5 Un sistema de Deep Learning para la clasificación de sentimiento en Twitter

Para cubrir el enfoque de redes neuronales decidimos estudiar el sistema Coooolll, premiado en segundo lugar en SemEval2014. Coooolll es un sistema de deep learning para clasificación de sentimiento en Twitter. [13]

La particularidad de este sistema es que utiliza una combinación de técnicas para definir los features utilizados en el entrenamiento. Los features se dividen en dos categorías: features elegidos manualmente teniendo en cuenta el estado del arte, y features SSWE (sentiment-specific word embedding) que incluyen información del sentimiento de los tweets en la representación continua de las palabras, generados a partir de una red neuronal SSWEu que captura el sentimiento de una oración además del contexto sintáctico de las palabras.

Los features SSWE son obtenidos a partir de 10 millones de tweets anotados automáticamente como positivos y negativos. Dado un n-grama y la polaridad de una oración, la red neuronal SSWEu predice un vector de escalares de 2 dimensiones: la primera representando el contexto sintáctico del n-grama y la segunda el sentimiento del n-grama.

Los features elegidos manualmente fueron: la cantidad de palabras escritas en mayúscula, la cantidad de emoticones negativos y positivos, y la cantidad de palabras que contienen más de 2 letras iguales (como por ejemplo "bieeen"). También se utilizaron varios lexicones para extraer features referidos a la cantidad de palabras que expresan sentimiento, la palabra con mayor polaridad, y la suma de las polaridades de las palabras de la oración. Además se utilizó la cantidad de negaciones y la presencia de ciertos n-gramas en el tweet.

Los mejores resultados del sistema se midieron utilizando la medida F1, llegando al 70% experimentando con 3 categorías de clasificación (positivo, negativo y neutral), y al 87% experimentando solamente con las categorías positivo y negativo.

2.3.6 Análisis de sentimientos en Twitter: TASS 2015

Esta sección se trata sobre uno de los artículos presentados para la tarea de clasificación global de cinco niveles de polaridad para un conjunto de tweets en español, el cual fue uno de los retos del TASS 2015. [14]

En este trabajo la representación de los tweets estuvo basada en características lingüísticas y de polaridad como lematizador de palabras, filtros de palabras, reglas de negación, entre otros. El preprocesamiento se enfocó en la corrección de errores, uso de tags especiales, POS tagging y procesamiento de negaciones.

Adicionalmente, con el fin de eliminar errores tipográficos se realizaron transformaciones pseudo fonéticas con reglas.

Finalmente, se utilizaron diferentes transformaciones como LDA, LSI y la matriz TF-IDF. Todas estas representaciones se combinaron con el clasificador SVM. Los resultados muestran que LSI y la matriz TF-IDF mejoran el rendimiento del clasificador SVM utilizado alcanzando un 40,4% de medida F1.

2.3.7 Resumen de TASS 2016

En este artículo se resume la quinta edición del taller de evaluación experimental TASS 2016, enmarcada dentro del Congreso Internacional SEPLN 2016. El principal objetivo de TASS es promover la investigación y el desarrollo de nuevos algoritmos, recursos y técnicas para el análisis de sentimientos en medios sociales (concretamente en Twitter), aplicado al idioma español. [15]

El corpus contiene 68.000 tweets escritos en español y el mismo se dividió en un diez por ciento para el conjunto de entrenamiento y noventa por ciento para el conjunto de testeo. La edición de 2016 se centró en el análisis y la comparación de los sistemas con las presentaciones de ediciones anteriores.

Utilizando 6 niveles de polaridad (Extremadamente Positivo, Positivo, Negativo, Fuertemente negativo y Ninguno), el mejor puntaje para estos 6 niveles de polaridad fue de 51,8% de medida F1.

2.3.8 Clasificación de sentimientos utilizando emoticones

Se propone crear una fuente novedosa de entrenamiento de datos basado en el lenguaje en conjunto con emoticones en Usenet newsgroups, para textos en inglés. [16]

Se utilizó una implementación propia del clasificador Naive Bayes y la implementación de SVM light de máquinas de vectores soporte que entre sus principales características destaca el permitir resolver no solo problemas de clasificación y regresión, sino también de ranking. Permite manejar cientos de miles de ejemplos de entrenamientos y muchos miles de vectores soporte.

El resultado medio utilizando el clasificador Naive Bayes fue de 61,5% mientras que el de SVM fue de 70,1%. Encontraron que mejora considerablemente cuando se clasifican textos que hablan sobre el mismo tema a cuando se usa datos mezclados.

2.3.9 Conclusiones

Si bien muchos de los artículos se basan en investigaciones sobre textos en inglés, los métodos y técnicas utilizadas nos permitieron analizar y anticiparnos a las dificultades que presenta evaluar texto escrito en una red social como Twitter. Estudiamos los obstáculos encontrados por cada uno de ellos y tomamos las soluciones que mejor se adecuan a nuestro caso de estudio.

Algunos de los problemas más comunes que se plantean en estos trabajos son la informalidad con la que se escribe y la limitación de caracteres. Los clasificadores más usados y que dan mejores resultados dentro de estos artículos fueron Naive Bayes y SVM. También analizamos distintos métodos de preprocesamiento, cuáles fueron los features más utilizados y distintos enfoques de categorización. La clasificación entre 2 categorías (positivo y negativo) es el método más utilizado.

Finalmente, estos trabajos también nos permitieron tener una referencia sobre los recursos utilizados, como por ejemplo el tamaño de corpus necesario, así como los resultados obtenidos.

2.4 Estudio de los métodos más usuales

En el estudio de trabajos relacionados se vieron algunos de los métodos más usuales para abordar este problema. En esta sección se describen los métodos basados en reglas y los métodos basados en aprendizaje automático.

2.4.1 Métodos basados en reglas

Los métodos basados en reglas se componen de algoritmos que se basan en exclusivamente en el uso de reglas deterministas, y no utilizan aprendizaje automático. En general resultan muy eficaces si todas las situaciones en las cuales se pueden tomar decisiones son conocidas de antemano. Por lo general se basan en el uso de lexicones de sentimiento y a partir de estos se construye un sistema que utiliza heurísticas lingüísticas, como por ejemplo el conteo de palabras positivas y negativas. También es posible implementar métodos más complejos mediante el uso reglas gramaticales, haciendo uso de la información semántica y sintáctica del texto y/o agregando elementos como pueden ser negadores o intensificadores que influyen en las polaridades de la opinión. El resultado de estos algoritmos es limitado y depende de la profundidad y amplitud de recursos empleados.

2.4.2 Métodos basados en aprendizaje automático

El aprendizaje automático es el subcampo de las ciencias de la computación (y una rama de la inteligencia artificial) cuyo objetivo es desarrollar técnicas que permiten que las computadoras aprendan. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de un información suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. El aprendizaje automático puede ser visto como un intento de automatizar algunas partes del método científico mediante métodos matemáticos. [17]

En esta sección se realiza una breve descripción de los clasificadores basados en aprendizaje automático utilizados.

Naïve Bayes

En teoría de la probabilidad y minería de datos, un clasificador Bayesiano ingenuo (Naïve Bayes) es un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales. Es a causa de estas simplificaciones (que se suelen resumir en la hipótesis de independencia entre las variables predictoras) que recibe el apelativo de ingenuo.

La clase $c*$ se le asigna a un tweet d, donde:

$$c* = argmac_c P_{NB}(c|d)$$

$$P_{NB}(c|d) := \frac{(P(c)\,\Sigma_{i=1}^{m} P(f|c)^{n_i(d)})}{P(d)}$$

En esta fórmula, f representa un feature y $n_i(d)$ representa la cantidad de veces que el feature f_i es encontrada en tweet d. Hay un total de m features. Los parámetros $P(c)$ y $P(f|c)$ se obtienen a través de estimaciones de máxima verosimilitud, y el suavizado de Laplace se utiliza para contemplar features no vistos.

Árboles de decisión

El aprendizaje basado en árboles de decisión es un método comúnmente utilizado en la minería de datos. [18] El objetivo es crear un modelo que predice el valor de una variable de destino en función de diversas variables de entrada. Los modelos de árbol donde la variable de destino puede tomar un conjunto finito de valores se denominan "árboles de clasificación". En estas estructuras de árbol, las hojas representan etiquetas de clase y las ramas representan las conjunciones de características que conducen a esas etiquetas de clase. Los árboles de decisión donde la variable de destino puede tomar valores continuos (por lo general números reales) se llaman "árboles de regresión".

SVM

Las Máquinas Vectoriales de Soporte (SVMs) son un conjunto de algoritmos de aprendizaje supervisado que están propiamente relacionados con problemas de clasificación y regresión. La teoría de la SVM está basada en la idea de minimización de riesgo estructural. En muchas aplicaciones, las SVM han mostrado tener gran desempeño y han sido introducidas como herramientas poderosas para resolver problemas de clasificación. [19]

Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Una SVM primero mapea los puntos de entrada a un espacio de features y encuentra un hiperplano que los separe y maximice el margen entre las clases en este espacio. Se obtiene una buena separación mediante la construcción del hiperplano que maximiza la distancia al dato más cercano de cualquier clase. Nuevos datos se clasifican asignando el punto correspondiente a una sección u otra del hiperplano.

KNN

k vecinos más cercanos (k Nearest Neighbors) es un método de clasificación supervisada donde se clasifica un nuevo elemento mediante el análisis de la clasificación de los elementos conocidos más cercanos en el espacio. Se ubican a las instancias en un espacio vectorial, de forma análoga a SVM. Podemos entonces considerar la similitud de dos puntos como la distancia entre ellos en este espacio bajo alguna métrica apropiada. Para predecir una nueva instancia, se clasifica directamente según el valor de ejemplos vistos considerados parecidos: se elige entre los k puntos de datos más cercanos a la nueva instancia, y se toma como clasificación la más común entre ellos. Es por eso que se llama el algoritmo de vecinos más cercanos. [20]
k-nn es un algoritmo de tipo tipo "lazy learning", donde la función se aproxima solo localmente y todo el cómputo es diferido a la clasificación.

Regresión logística

Al igual que Naive Bayes, la regresión logística funciona extrayendo un conjunto de features ponderados y realizando combinaciones lineales de ellos. En nuestro caso utilizamos regresión logística multinomial, ya que debemos clasificar en múltiples clases. [21]

La diferencia más importante con Naive Bayes es que este último es un clasificador generativo, mientras que regresión logística es un algoritmo de clasificación discriminativo. Esto quiere decir que teniendo en

cuenta que el trabajo de un clasificador probabilístico es elegir una etiqueta “y” dada una entrada “x”, eligiendo el “y” que maximice $P(y|x)$. Naive Bayes utiliza la regla de Bayes para estimar el mejor “y” indirectamente a través de la función de verosimilitud $P(x|y)$.

$$\widehat{y} = argmax_y\ P(y|x)\ = argmax_y\ P(x|y)P(y)$$

Esta indirección es lo que convierte a Naive Bayes en un modelo generativo, entrenado para generar la data x a partir de la clase y. La función de verosimilitud $P(x|y)$ expresa que dada la clase y, se intenta predecir qué features esperamos en la entrada x. Luego utilizando la regla de Bayes se calcula la probabilidad deseada $P(y|x)$

En cambio un modelo discriminativo calcula directamente $P(y|x)$ al discriminar entre los diferentes posible valores de la clase y. La regresión logística estima $P(y|x)$ extrayendo y combinando linealmente algunos features de la entrada. Sin embargo no es posible calcular $P(y|x)$ directamente con los features y pesos de la siguiente forma: $p(c|x)\ ? = \sum_{i=1}^{N} w_i f_i$

El problema es que la fórmula anterior genera valores entre $-\infty$ y $+\infty$, mientras que una probabilidad real debe encontrarse entre los valores 0 y 1. Por lo tanto primero se calcula el exponente de $\sum_i w_i f_i$, con el objetivo de siempre obtener valores positivos. Luego se normaliza la fórmula para obtener una probabilidad entre 0 y 1. Obteniendo como resultado:

$$p(c|x)\ =\ \tfrac{1}{Z} exp\left(\sum_i w_i f_i\right)$$

Redes Neuronales

Las redes de neuronas artificiales (RNA) son un paradigma de aprendizaje automático inspirado en las neuronas de los sistemas nerviosos de los animales [22]. Se trata de un sistema de enlaces de neuronas que colaboran entre sí para producir un estímulo de salida. Las conexiones tienen pesos numéricos que se adaptan según la experiencia. De esta manera, las redes neurales se adaptan a un impulso y son capaces de aprender. La importancia de las redes neurales cayó durante un tiempo con el desarrollo de los vectores de soporte y clasificadores lineales, pero volvió a surgir a finales de la década de 2000 con la llegada del aprendizaje profundo.

Tal vez la mayor ventaja de las RNA es su capacidad de ser utilizado como un mecanismo de función de aproximación arbitraria que "aprende" a partir de datos observados. Sin embargo, su uso no es tan sencillo, y una relativamente buena comprensión de la teoría subyacente es esencial.

- Elección de modelo: Esto dependerá de la representación de datos y la aplicación. Excesivamente complejos modelos tienden a conducir a problemas en el aprendizaje.
- Algoritmo de aprendizaje: Existen numerosas soluciones de compromiso entre los algoritmos de aprendizaje. La selección y el ajuste de un algoritmo para la formación en los datos invisibles requieren una cantidad significativa de tiempo y una inmensa cantidad de datos para el

entrenamiento, sin embargo, la estructura de una RNA es paralela, por lo cual una vez entrenada con el hardware adecuado, se pueden obtener respuestas en tiempo real.

- Robustez: Si se seleccionan el modelo, la función de costo y algoritmo de aprendizaje adecuado, la RNA resultante puede ser extremadamente robusto, pueden aprender a reconocer patrones con ruido, distorsionados o incompletos.

Con la aplicación correcta, las RNA pueden ser utilizadas de forma natural en el aprendizaje en línea y aplicaciones de grandes conjuntos de datos. Su aplicación sencilla y la existencia de dependencias locales en su mayoría expuestos en la estructura permiten implementaciones rápidas y paralelas en el hardware.

Capítulo 3

Recursos Generados

A lo largo del proyecto se generó una serie de recursos que fueron reutilizados por los distintos algoritmos y clasificadores, y que a su vez pueden resultar reutilizables en otros proyectos. En esta sección se describe en detalle el proceso de creación, la composición y las conclusiones obtenidas a partir de la generación de estos recursos, dentro de los que se se encuentran el corpus de tweets etiquetados y los distintos lexicones utilizados.

3.1 Construcción del corpus

En esta sección se detalla el proceso de construcción del corpus que contiene el conjunto de tweets etiquetados. Los tweets fueron descargados mediante la API de Twitter, desde múltiples cuentas uruguayas que se detallarán más adelante. Como se mencionó en los objetivos, una vez que los tweets fueron descargados múltiples usuarios participaron de un proceso de votación para clasificarlos entre "opinión positiva", "opinión negativa", "opinión neutral" o "no es opinión". Dentro de esta sección explicará el proceso de construcción del corpus, los criterios de anotación, un estudio del corpus conformado y métricas de concordancia entre anotadores. Estas métricas surgen como conclusión de la composición y el proceso de construcción del mismo.

3.1.1 Descarga de tweets

En la siguiente sección especificaremos el procedimiento seguido para descargar y poblar el corpus de tweets, el cual fue etiquetado manualmente y utilizado posteriormente como entrada para los clasificadores.

Web y aplicación Android para el registro de votos

En primer lugar se construyó una aplicación web con el objetivo de facilitar la extracción, almacenamiento y etiquetado manual de los tweets que conforman el corpus. En paralelo se construyó una aplicación nativa de Android con el mismo propósito.

Mediante ambas interfaces, múltiples usuarios ingresan con el propósito de etiquetar tweets que han sido previamente almacenados por el grupo en una base de datos. Cuando un usuario se autentica en la aplicación o web, se le asigna un ID único persistente entre sesiones que permite identificarlos unívocamente.

Luego de que un usuario ingresa a la aplicación, se muestra un tweet aleatorio desde la base de datos. Los usuarios votan para clasificar los tweets según las siguientes opciones: “Opinión positiva”, “Opinión negativa”, “Opinión neutral”, “No es opinión” o “No estoy seguro”.

Si se elige la opción “No estoy seguro”, no se registra ninguna entrada del usuario, y simplemente se muestra un nuevo tweet. De esta manera se evita forzar a que los usuarios voten cuando no están seguros, generando datos de alto valor.

Con el objetivo de llegar a una buena cobertura sobre el total de los tweets, una vez que una de las clasificaciones posibles del mismo cuenta con al menos 3 votos más que el resto, se deja de mostrar ese tweet.

La web puede encontrarse en http://proygrado.herokuapp.com (protegida por contraseña), y los usuarios pueden ingresar a clasificar tweets previamente almacenados. Al final de este documento se encuentra un anexo donde se detalla los aspectos técnicos de los servicios implementados en la creación de esta solución.

Imagen 3.1- Captura de la aplicación web

Imagen 3.2- Captura de la aplicación Android

Selección de cuentas, tags y almacenamiento de tweets y voto

En esta sección se describen decisiones relevantes al mecanismo de selección del origen de los tweets y a la extracción de los mismos.

Para confeccionar el corpus se eligieron cuentas y temas diversos, intentando crear un balance entre 3 áreas donde típicamente tiene sentido el análisis de reputación: política, deporte y temas polémicos de los cuales se habló mucho durante la confección del corpus (#Uber, #NiUnaMenos, etc).

Habiendo inicialmente descargado una amplia mayoría de tweets de cuentas asociadas a la política, luego de la primera ronda de votación descubrimos que había un fuerte desbalance entre las clases del corpus. Al haberse registrado que más de un 80% de las opiniones eran negativas, se agregaron temas marcadamente positivos para contrarrestar, dentro de los que destacan “@LuisSuarez9”, “#Messi” y “#Uber”.

- Lista de fuentes de tweets:
    - Cuentas
        - turkabdala
        - igalvar71
        - janarhertz
        - edgardonovick
        - frascafrasca
        - LuisLacallePou
        - PedroBordaberry
        - FlacoLamolle
        - MiguelCapito
        - jorgewlarranaga
        - Pablo_Mieres
        - gabrielhpereyra
    - Hashtags
        - #Uber
        - #NiUnaMenos
        - #Messi
        - #Brexit
        - #Suarez
    - Menciones a usuarios
        - @SoyCelestee
        - @LuisSuarez9

Una vez que se seleccionaron las cuentas, se extraen y almacenan los siguientes datos de cada tweet: ID, texto, idioma, fecha de creación, cantidad de usuarios que retweetearon ese tweet, ID del autor, nombre de usuario del autor, nombre del autor e imagen del autor. La extracción de datos de twitter se realiza mediante un conjunto de servicios implementados por el grupo, y detallados en el Anexo 1.

Se decidió filtrar manualmente los tweets que no contienen texto (solo imagen, etc) y los tweets cuyo idioma no es el español. Más adelante se presentan detalles de la cantidad de tweets filtrados en la sección 3.1.4 (composición del corpus).

Nótese que algunos de esos datos se almacenan con el único propósito de mostrarle información pertinente al usuario en la aplicación y en la web (imagen y nombre del autor, etc).
Adicionalmente, se cuentan los votos de cada una de las categorías.

Dificultades

Es importante destacar que nos encontramos con las siguientes limitantes en la API de twitter durante la construcción del corpus.

- La API de twitter permite solamente obtener los últimos 200 tweets de un usuario dado.
- Las búsquedas a la API de twitter retornan un máximo de 100 resultados, y solamente se retornan los tweets de la última semana.
- Para poder obtener todos los resultados de la última semana de una búsqueda es necesario realizar varias llamadas sucesivas a la API, donde cada una trae un máximo de 100 tweets.
- Filtrar geográficamente por un país resulta sumamente complejo. Optamos por uno de los filtros disponibles donde se especifica latitud, longitud y un radio, y solo se devuelven los tweets emitidos dentro de ese radio.
- Si bien twitter cuenta con un clasificador que indica el idioma de cada tweet, encontramos casos donde se equivoca como muestra la imagen 3.3. También es importante notar que hay muchos usuarios que o bien escriben o retweetean en inglés cada tanto.

Imagen 3.3- Ejemplo de tweet con clasificación de idioma errónea (devuelve "Inglés")

3.1.2 Criterios de anotación

Según la RAE, una opinión es un "Juicio o valoración que se forma una persona respecto de algo o de alguien." [23]. Esto supone que la opinión admite la posibilidad de error ya que no hay evidencia plena. En este sentido, la opinión se considera como una afirmación con menor evidencia de la verdad que una certeza. Utilizamos esta definición como criterio para etiquetar opiniones. Por lo tanto, tomamos como etiquetamos como "no es opinión" a aquellos tweets que no entran dentro de esta definición. Es decir, aquellos que se pueden calificar como certezas o declaraciones inequívocas de un hecho y por lo tanto no son subjetivos ni expresan ningún tipo de sentimiento.

Dentro de las opiniones clasificamos como opiniones positivas a aquellas en las cuales creemos que la intención del autor del tweet fue transmitir un sentimiento positivo. A su vez clasificamos como opiniones negativas aquellas en las cuales consideramos que la intención del autor del tweet fue transmitir un sentimiento negativo.

Como opinión neutral definimos aquellas que entran dentro de la definición de opinión, pero que a nuestro entender no expresan ningún tipo de sentimiento. Es decir que son declaraciones subjetivas, que pueden estar equivocadas pero no transmiten sentimiento. Por ejemplo la declaración *"A mi entender la selección debería jugar con una formación 4 3 3"* claramente es una opinión, que no expresa sentimiento por la tanto es una opinión neutral.

Durante esta etapa nos encontramos con algunas dificultades. Por ejemplo, el hecho de que la anotación de opiniones en muchos casos puede ser muy subjetiva o tener más de una interpretación (ambigua) y/o carecer de contexto (lo que es sumamente común cuando consideramos que solo se pueden utilizar 140 caracteres para transmitir una idea). Para esto definimos un criterio único de anotación para cada problema encontrado con el fin de minimizar las diferencias entre los anotadores.

La falta de contexto es uno de los problemas más comunes que tienen los tweets y más aún en los casos de las opiniones que son irónicas. Para poder manejar adecuadamente la ironía es necesario contar con mucha información sobre el contexto de quién o qué se está hablando. A veces también es necesario saber el momento y el lugar donde se emite el mensaje, así como eventos o situaciones pertinentes. Para estos casos el criterio de anotación que creímos más conveniente fue el de intentar abstraernos lo más posible del contexto ya sea del tema o persona que se está hablando y de la persona que escribió el tweet y analizarlo de forma literal.

Otro de los casos encontrados fueron opiniones que tienen un sentimiento positivo y negativo a la vez. En estos casos el criterio que decidimos adoptar fue tratar de encontrar cuál es el sentimiento más fuerte dentro del texto o que quiere transmitir el autor del tweet. Un ejemplo de esto es "*Uruguay sin mucho fútbol pero con abundante marca y actitud*". En este caso acordamos que el sentimiento con más peso que quiere expresar el autor, es el que se encuentra a continuación de "pero", y por ende se anotó como opinión positiva.

3.1.3 Limpieza de clasificaciones ambiguas

Todo tweet del corpus debe contar con una clasificación (etiqueta) clara y única. Sin embargo, el sistema de votación implementado dá lugar a que existan tweets que no cumplan con esta condición, a los cuales llamamos ambiguos. Supongamos que tenemos un tweet con 4 votos: 2 votos para “opinión positiva” y 2 votos para “opinión negativa” (y ningún otro voto para el resto de las categorías posibles). Siguiendo este ejemplo, no podemos afirmar con seguridad si este tweet es una opinión positiva o negativa, y por ende no cuenta con una categoría definida. Optamos por descartar todos los tweets ambiguos.

3.1.4 Composición del corpus

Luego de que tanto la aplicación web y Android estuvieron disponibles por un período de aproximadamente 6 meses, se dió por finalizado el proceso de votación y se procedió a obtener datos estadísticos sobre cantidad de tweets descargados, votos emitidos por categoría y cantidad de votantes. Estos datos son presentados a continuación.

- Cantidad de tweets descargados inicialmente: 3600
- Cantidad de tweets luego de filtrar manualmente por idioma y contenido: 2738
- Cantidad de votos: 4845
- Cantidad de votantes: 17
- Cantidad de votos positivos: 1260 (26% de 4845)
- Cantidad de votos negativos: 1512 (31% 4845)
- Cantidad de votos neutrales: 330 (7% 4845)
- Cantidad de votos “no es opinión”: 1743 (36% de 4845)
- Cantidad de tweets luego de filtrar clasificaciones ambiguas: 2629
- Cantidad de tweets con mayoría de votos positivos: 701 (27% de 2629)
- Cantidad de tweets con mayoría de votos negativos: 671 (26% de 2629)
- Cantidad de tweets con mayoría de votos neutrales: 159 (5% de 2629)
- Cantidad de tweets con mayoría de votos “no es opinión”: 1098 (42% de 2629)

3.1.5 Métricas de concordancia

Tradicionalmente, la concordancia en usuarios se considera como un buen estimador del tope de la performance a la que puede llegar el clasificador. Ya que se trata de una medida sumamente importante, decidimos calcularla de 2 formas distintas: internamente y considerando los votos emitidos por todos los votantes. Adicionalmente, calculamos también la concordancia si consideramos solamente 2 clasificaciones posibles: es opinión o no es opinión. Llamamos a esto “concordancia simple”, y en otras palabras se obtiene al unir las clasificaciones de “es opinión positiva”, “es opinión negativa” y “es opinión neutral” en “es opinión”.

Es fácil ver que cuando un tweet tiene 1 solo voto, no tiene sentido calcular la concordancia sobre el mismo (es siempre 100%). De la misma manera, cuando un tweet tiene solamente 2 votos la confianza que tenemos sobre la concordancia en los votos del mismo es muy baja dado que los valores posibles son 0%, 50% o 100%. Por este motivo, utilizamos solamente los tweets con al menos 3 votos para calcular la concordancia sobre todos los votantes en el corpus.

Concordancia entre los votantes

En primer lugar medimos la concordancia entre todos los votantes. Calculamos esta medida como la probabilidad de que 2 personas distintas emitan el mismo voto para un tweet dado, y promediamos la probabilidad entre todos los tweets del corpus. La misma se realizó con 1223 tweets del corpus, que son los que cuentan con al menos 3 votos.

La probabilidad de que 2 votantes independientes emitan el mismo voto es la suma de la probabilidades de que ambos voten positivo, negativo, neutral o "no opinión". Estas probabilidades serán estimadas mediante conteos. Por ejemplo, si un tweet tiene solamente 3 votos positivos y 1 negativo, estimamos que la probabilidad de que 2 votantes independientes voten que es una opinión positiva está dado por (¾).(¾).

Tomando esto en cuenta, el algoritmo puede resumirse de la siguiente forma:

$$concordancia_{4\ clases}(Corpus) = P(voto_{votante\ 1} = voto_{votante\ 2}) =$$
$$= (P_{voto=positivo}(t))^2 + (P_{voto=negativo}(t))^2 + (P_{voto=neutral}(t))^2 + (P_{voto=no\ opinión}(t))^2$$
$$\simeq \sum_{t \in tweets(Corpus)} \left(\frac{\#votos\ positivos(t)}{\#total\ votos\ (t)}\right)^2 + \left(\frac{\#votos\ negativos(t)}{\#total\ votos\ (t)}\right)^2 + \left(\frac{\#votos\ neutrales(t)}{\#total\ votos\ (t)}\right)^2 + \left(\frac{\#votos\ no\ es\ opinión(t)}{\#total\ votos\ (t)}\right)^2$$

Siguiendo el ejemplo anterior donde un tweet tiene 3 votos positivos y 1 negativo, la concordancia quedaría:

$$concordancia_{4\ clases} = (3/4)^2 + (1/4)^2 + (0/4)^2 + (0/4)^2 = 62.5\%$$

Luego se implementó el siguiente algoritmo para medir la concordancia simple.

$$concordancia_{simple}(Corpus) = P(voto_{votante\ 1} = voto_{votante\ 2} \mid clasificaciones = \{es\ opinión,\ no\ es\ opinión\}) =$$
$$= (P_{voto=opinión}(t))^2 + (P_{voto=no\ opinión}(t))^2$$
$$\simeq \sum_{t \in tweets(Corpus)} \left(\frac{\#votos\ positivos(t) + \#votos\ negativos(t) + votos\ neutrales(t)}{\#total\ votos\ (t)}\right)^2 + \left(\frac{\#votos\ no\ es\ opinión(t)}{\#total\ votos\ (t)}\right)^2$$

Concordancia interna al equipo

Adicionalmente se midió la concordancia interna de los integrantes del grupo del proyecto. Se extrajo aleatoriamente 100 tweets del corpus, sobre los que cada integrante votó de forma independiente.

La concordancia interna se calculó promediando la concordancia de cada pareja de 2 miembros del equipo. Análogamente a lo implementado para medir la concordancia de todo el corpus, se calculó la concordancia al tomar las 4 clases y una concordancia simple.

$$concordancia_{interna} = \frac{concordancia(I_1, I_2) + concordancia(I_2, I_3) + concordancia(I_1, I_3)}{3}$$ donde:

$I_1 = integrante_1,\ I_2 = integrante_1,\ \ I_3 = integrante_3$

3.1.6 Resultados y conclusiones

En la tabla 3.1 se muestran los resultados de las métricas de concordancia, tanto la concordancia entre todos los votantes como la concordancia interna del equipo.

Tipo de concordancia	**Resultado**
Concordancia sobre todos los votantes con 4 clases	85%
Concordancia simple sobre todos los votantes (es opinión o no es opinión)	89%
Concordancia interna con 4 clases	77%
Concordancia interna con 3 clases (positivo, negativo, no es opinión)	80%
Concordancia interna simple (es opinión o no es opinión)	83%

Tabla 3.1: Concordancia entre votantes

A primera vista en la Tabla 3.1 resulta contra-intuitivo el hecho de que la concordancia medida sobre todos los votantes sea ligeramente mayor que la concordancia interna del equipo. Sin embargo, luego de un detenido análisis observamos que en general los votantes evitaban los tweets más conflictivos (donde es difícil o ambiguo determinar la polaridad). Esto se debe a que contaban con la posibilidad de saltear tweets mediante la acción de “No estoy seguro”. Un gran número de estos tweets fueron descartados al filtrar por la cantidad de votos (al menos 3).

Adicionalmente, es importante notar que al medir la concordancia interna no contamos con ningún tipo de contexto, mientras que los votantes en la aplicación o la web veían el nombre e imagen del autor.

En la tabla 3.2 se muestra la concordancia entre todos los votantes discriminada por clases.

Tipo de concordancia	**Resultado**
Solamente considerando tweets positivos	84%
Solamente considerando tweets negativos	89%
Solamente considerando tweets neutrales	68%
Solamente considerando tweets no son opiniones	85%

Tabla 3.2: Concordancia entre votantes por clases

Una conclusión sumamente importante que surgió durante la etapa de análisis del problema y estudio del corpus fue el hecho de que la categoría "neutral" no aporta valor al problema debido a los siguientes motivos:

- Existen muy pocas opiniones realmente neutrales. Aún luego de intentar balancear el corpus, sólo fue posible conseguir un 7% de tweets que representan opiniones neutrales.
- Como votantes notamos una gran dificultad para distinguir entre opiniones neutrales y opiniones positivas o negativas, donde la diferenciación casi siempre es semántica o asociada al contexto.
- Como puede visualizarse en la sección anterior, la concordancia entre opiniones neutrales es marcadamente menor a la concordancia entre otras clases.
- Las opiniones neutrales no aportan valor en cuanto al análisis de reputación

Las conclusiones presentadas en esta sección nos llevaron a decidir que tendría más sentido que los clasificadores (a presentar en las secciones siguientes) trabajen sobre solamente 3 categorías: positivo, negativo y no es opinión. De todas maneras, queremos destacar que fueron implementados de manera tal que mediante configuración puedan ejecutarse para distinguir entre 2, 3 o 4 categorías,

De la misma forma, los resultados presentados a lo largo del documento se basan en 3 categorías.

3.2 Lexicones

A lo largo de este trabajo se utilizan tres tipos de lexicones. Dos de ellos se utilizan para evaluar la polaridad de ciertas palabras, siendo uno de ellos el recurso principal utilizado durante el transcurso de este proyecto y el segundo como apoyo en la extracción de features para los clasificadores. El tercer lexicón fue creado en este proyecto y diseñado para evaluar patrones que pueden indicar que un texto no es una opinión.

Cabe destacar que la incorporación de lexicones y mejoras realizadas sobre los mismos son resultado del proceso de investigación y experimentación realizados. A lo largo del informe se detalla el proceso de razonamiento detrás de cada mejora por lo que en esta sección nos limitaremos a describir y detallar el contenido de los mismos.

3.2.1 Lexicón ML-SentiCon

Se utiliza el lexicón ML-SentiCon al cual nos referiremos como "Lexicón Original", elaborado en la Universidad de Sevilla[24], disponible de forma pública para su uso.

Se trata de un conjunto de lexicones de polaridades semánticas a nivel de lemas para inglés, español, catalán, gallego y euskera. En nuestro trabajo solamente utilizamos el lexicón correspondiente al idioma español. Los lexicones se han generado automaticamente a partir de una mejora del método utilizado para generar SentiWordNet, un recurso ampliamente utilizado que recoge estimaciones de positividad y negatividad a nivel de synsets.

El lexicón cuenta con 11.557 palabras, 5.570 positivas y 5.987 negativas. Se encuentra estructurado en capas, lo que permite seleccionar distintos compromisos entre la cantidad de estimaciones de positividad y negatividad y la precisión de dichas estimaciones. Adicionalmente, se encuentra dividido en 8 niveles de confianza, donde el nivel 1 contiene las palabras con mayor confianza y el nivel 8 las palabras con confianza menor.

Dentro del lexicón, la polaridad de una palabra es un número real entre -1 y 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<senticon lang="es">
    <layer level="1">
        <positive>
            <lemma pos="a" pol="0.708" std="0.149"> acertado </lemma>
            <lemma pos="a" pol="0.906" std="0.125"> admirable </lemma>
            <lemma pos="n" pol="0.45" std="0.331"> admiración </lemma>
            <lemma pos="v" pol="0.75" std="0.177"> admirar </lemma>
            <lemma pos="a" pol="0.375" std="0.0"> afectivo </lemma>
            <lemma pos="n" pol="0.321" std="0.112"> afecto </lemma>
            <lemma pos="a" pol="0.563" std="0.131"> afectuoso </lemma>
            <lemma pos="n" pol="0.5" std="0.425"> afición </lemma>
            <lemma pos="a" pol="0.813" std="0.063"> afortunado </lemma>
            <lemma pos="a" pol="0.75" std="0.181"> agradable </lemma>
            <lemma pos="a" pol="0.5" std="0.125"> agradecido </lemma>
            <lemma pos="v" pol="0.458" std="0.315"> alegrar </lemma>
            <lemma pos="a" pol="0.661" std="0.185"> alegre </lemma>
            <lemma pos="a" pol="0.594" std="0.0"> alentador </lemma>
```

Imagen 3.4- Extracto del lexicón

Ampliación de vocabulario del lexicón

Como se comenta en la sección 4.3.2, una de las conclusiones obtenidas al analizar los casos incorrectamente clasificados por la línea base fue la necesidad de agregar palabras al lexicón, ya que por defecto contiene muchas palabras en español neutro pero no contempla los gerundios y expresiones uruguayas.

Para realizar esta tarea implementamos un algoritmo que elabora una lista de las palabras que se repiten por lo menos 3 veces en tweets de una misma categoría en el corpus de entrenamiento y que además no se encuentran en el lexicón. Analizamos esta lista de palabras manualmente una por una, inspeccionando si realmente es positiva o negativa según corresponda. Luego de este filtro, las palabras son agregadas con una polaridad igual a un sinónimo o a la palabra más parecida que ya se encontraba en el lexicón. En total se agregaron 434 palabras al lexicón: 146 positivas y 288 negativas, que sumado a las palabras originales suman 11.991.

Incorporación de emoticones al lexicón

Otra iniciativa para expandir el lexicón fue incorporar emoticones al mismo. Inicialmente la existencia de emoticones se agregó como un nuevo feature, pero no obtuvimos buenos resultados debido a que no contamos con una gran cantidad de emoticones en el lexicón. Notamos que el peso del feature se veía fuertemente disminuido frente otros y no aporta cambios significativos.

Optamos por eliminar este feature y agregar los emoticones directamente al lexicón como si fueran una palabra más. De esta forma se aumenta el valor de todos los otros features existentes.

Nos contactamos con los autores de uno de los artículos [25] de la Universidad de Columbia presentados en la sección de trabajos relacionados el cual menciona un lexicón de emoticones. Luego de que nos proporcionaron su lexicón de emoticones, notamos que la lista de emoticones existentes desde el 2011 ha crecido exponencialmente.

Para actualizar este lexicón, mapeamos manualmente la polaridad que tenía cada uno a la polaridad de nuestro lexicón. Fue necesario implementar métodos de preprocesamiento para poder trabajar con los emoticones presentes en los tweets. Esto se detalla en la sección de preprocesamiento.

Luego de la ampliación del vocabulario del Lexicón Original y la incorporación de emoticones, llamaremos a este nuevo lexicón mejorado “Lexicón Extendido” o “senticon”.

3.2.2 Segundo Lexicón

Con el objetivo de no depender solo de un lexicón y de los criterios de los creadores del mismo, así como para poder cubrir mayor cantidad de sentimientos de palabras, decidimos agregar un segundo lexicón. “ElhPolar_esV1.lex” es el lexicón utilizado en la competencia TASS2014 [26] y en este documento lo llamaremos “Lexicón ELH”. Cuenta con 5210 palabras, 1897 positivas y 3302 negativas. A diferencia del lexicón anterior, “ElhPolar” no cuenta con una escala para establecer la polaridad. Es decir que solo las

clasifica como positivas o negativas, ni con niveles de confianza. Además, claramente es de menor tamaño. Es por esto que solamente lo utilizamos como recurso para calcular alguno de los features que se verán mas adelante, tales como “Cantidad de palabras positivas dentro de ELH” o “Cantidad de palabras negativas dentro de ELH”.

3.2.3 Antisenticon

Dado que el senticon (ML-SentiCon) funcionó muy bien para clasificar opiniones, nos pareció intuitivo intentar aplicar la misma idea para tweets que no son opiniones. De esta forma creamos un nuevo lexicón que denominamos “Antisenticon” , el cual contiene secuencias de términos o patrones que generalmente aparecen en tweets que no son opinión.

El Antisenticon se creó analizando un nuevo conjunto de tweets pertenecientes a cuentas especialmente seleccionadas que contienen en su gran mayoría tweets informativos. Algunas de las cuentas de las cuales se extrajeron los tweets fueron "@Subrayado", "@elpaisuy", "@Ovaciondigital", "@ObservadorUY", "@Oceano939". Se realizó un análisis de tweets de estas cuentas buscando patrones que nos provean la información de que lo que se está diciendo no es una opinión. Es fácil ver que hay ciertos patrones que, en caso de cumplirse, indican casi inequívocamente que un tweet no es una opinión: avisos de programación, anuncios, publicidad, etc.

Imagen 4.6- Ejemplo de tweet que no es opinión

Una diferencia clave con el senticon surge de la siguiente observación: palabras aisladas no son suficientes para determinar con alto grado de confianza que una oración no es una opinión. Sin embargo, ciertos patrones o secuencias de palabras pueden determinar si un texto es o no una opinión con un mayor nivel de confianza. Como consecuencia de esta observación cada entrada del Antisenticon está formada por una secuencia de palabras y un largo de “ventana” X. Para considerar que un patrón del Antisenticon se cumple en una oración, las palabras del patrón deben aparecer en la oración con una separación máxima de X palabras entre la primera y la última.

Por ejemplo el patrón (“hoy”,”cumplen”, “años”) con una ventana de largo 6 se cumple para el siguiente tweet “Hoy se cumplen 60 años que mi bisabuelo Atilio Arrillaga Safons falleció en el Directorio del Partido Nacional”.

El Antisenticon cuenta con 27 patrones o entradas. A continuación se presentan algunos ejemplos de patrones tomados del Antisenticon a modo ilustrativo:

- (“a”, “las”, <HORARIO>)
- (“en”, “instantes”)
- (“mañana”,”en)
- (“canal”, <NUMERO>)
- (“la”, “entrevista”, “con”)

Como puede observarse, se trata de patrones de expresiones donde se busca transmitir información o realizar anuncios, lo cual es sumamente común en Twitter. Adicionalmente, en la sección 4.1 (preprocesamiento) se detalla el procedimiento realizado para detectar los patrones tales como <HORARIO>, <NÚMERO>, etc.

Cabe destacar que a lo largo del trabajo no pretendemos crear un Antisenticon extenso o comprensivo. Simplemente se intentó experimentar con la idea y medir el valor que puede aportar. Más adelante, se presenta como trabajo a futuro extender y enriquecer los patrones del Antisenticon.

Capítulo 4

Realización de la solución

En este capítulo se describe en detalle la solución llevada a cabo para el problema dado. Se describe el pre-procesamiento que se le realizó a los tweets, las métricas utilizadas para evaluar el desempeño de los clasificadores, la construcción de la línea base, herramientas utilizadas, y los clasificadores construidos.

Es necesario destacar que toda la solución que se detalla en este capítulo fue implementada utilizando Jupyter notebooks con Python 2.7.

En la imagen 4.1 se presenta un esquema simplificado de la solución implementada que se analizará a lo largo de este capítulo. Representa el proceso por el cual se cargan tweets, se entrena a los clasificadores y se realiza el análisis de reputación.

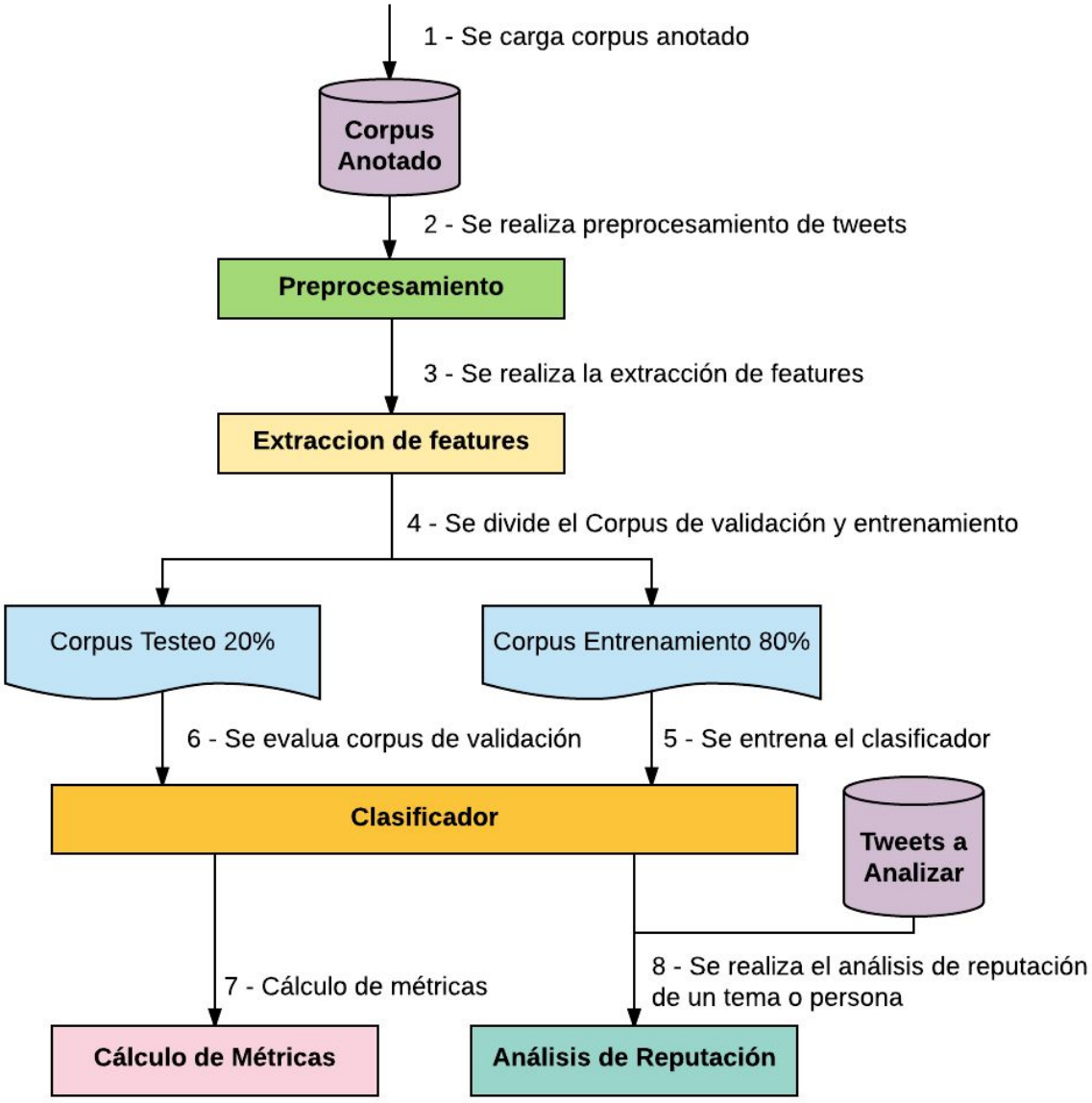


Imagen 4.1: Esquema solución implementada

4.1 Preprocesamiento de los tweets

"El propósito fundamental del preprocesamiento de datos es manipular y transformar datos en estado puro, de forma que la información contenida en estos datos pueda ser mostrada o accedida más fácilmente." [27]

Un tweet es un texto escrito de forma informal que contiene como máximo 140 caracteres, por lo que más allá de errores ortográficos o de sintaxis, los usuarios al expresar una opinión muchas veces utilizan

contracciones, abreviaciones, siglas o hasta se eliminan palabras para intentar expresar una idea. La mayoría de los tweets ni siquiera formando oraciones completas, y utilizan caracteres especiales para los hashtags o menciones. Es por esto que sin preprocesamiento previo sería casi imposible utilizar algún analizador sintáctico, analizador morfológico o siquiera buscar las palabras del tweet en un lexicón.

Uno de los obstáculos más grandes para que los textos pudieran ser aptos para el aprendizaje fue adaptar la estructura de los tweets, sustituir siglas y corregir palabras con errores o incompletas,.

Se creó un conjunto de funciones específicas para resolver distintos aspectos del preprocesamiento. Cada una de estas funciones es aplicada al texto del tweet luego de que estos son cargados y previo al entrenamiento o creación de features.

Antes de mencionar las funciones utilizadas y el propósito que cumple cada una de ellas, vale la pena destacar que cada función fue implementada con 4 modos de sustitución:

“Off” - En este modo el tweet no se vera afectado por la función, retornando el texto original del tweet.
“Sustitución Nombre” - En este modo la función elimina caracteres especiales del contexto de un nombre o mención, pero mantiene el nombre original en el texto.
"Sustitución Genérica" - En este modo la función elimina caracteres especiales del contexto de un elemento, nombre o mención, y además sustituye el elemento encontrado por un TAG genérico, por ejemplo <LINK>, <MENCION>, <HASHTAG>.
"Sustitución Eliminar” - En este modo la función elimina el elemento, nombre o mención.

4.1.1 Limpieza de etiquetas de Twitter

Durante el preprocesamiento, las funciones se utilizan por defecto en modo “Sustitución nombre”. Este modo elimina los caracteres especiales que conforman las menciones a otros usuarios y hashtags, sustituyendo la mención por el nombre de usuario al que se hace mención o las palabras que integran el hashtag como parte del texto. Por ejemplo, luego de ser preprocesado el siguiente tweet “*Que linda juntada con @Carlitos #DomingosEnFamilia*” se transforma en “*Que linda juntada con Carlitos Domingos en familia*”. Es posible configurar el preprocesamiento para que, en lugar de lo mencionado anteriormente, se sustituya la mención y el hashtag por los tags <MENCION> y <HASHTAG>.

Durante esta etapa también se sustituyen los links por el tag <LINK>, y se eliminan distintos tipos de caracteres especiales del texto.

4.1.2 Emoticones y emojis

Otra forma de comunicación y de expresar emociones que ha tomado mucha popularidad es el uso de emoticones y emojis en chats, mensajes de texto, y redes sociales.

Emoticón es un neologismo que proviene de emoción e icono. Un emoticón es una secuencia de caracteres ASCII que, en un principio, representaba una cara humana y expresaba una emoción. Posteriormente, se crearon otros con significados muy diversos. [28] Emoji es una palabra japonesa que se utiliza para designar las imágenes o pictogramas que son usados para expresar una idea, emoción o sentimiento en medios de comunicación digital. El término es una palabra compuesta que significa, imagen (e) + letra (moji). [29]

La diferencia con los emoticones es que los emojis son representados con imágenes mientras que los emoticones están formados por símbolos del teclado, como la cara sonriente que se representa con dos puntos seguidos de un guión y un paréntesis: “:-)”. Actualmente la mayoría de apps y servicios de comunicación digital convierten los emoticones en emoji cuando se tipean los símbolos.

Es claro ver que estos símbolos brindan mucha información sobre el sentimiento de un tweet. Sin embargo para asociar cada emoticón o emoji con un sentimiento fue necesario normalizar los distintos tipos de emojis. Con normalizar un emoji nos referimos a que consideramos que un emoji de un color expresa el mismo sentimiento y con la misma intensidad que el mismo emoji pero de otro color. Es decir que la información con respecto al sentimiento que transmite este emoji:

es igual a este:

Esto último fue necesario ya que twitter soporta 5 colores (tonalidades de piel) distintos para cada emoji y el objetivo es agrupar la mayor cantidad de emojis en la misma categoría para que resulte más fácil generar patrones en el entrenamiento. La solución implementada consistió en eliminar los códigos de los colores de los emojis, ya que consideramos que el color no aporta información con respecto al sentimiento que transmite el emoji.

4.1.3 Gramática

Por último, como se menciona en la introducción de esta sección notamos que muchas veces los usuarios escriben con abreviaciones o siglas. Por ejemplo “*te quiero mucho*” como “*tqm*” o “*porque*” como “*xq*”. Para poder capturar este tipo de frases que expresan sentimiento y a su vez para ayudar a freeling a parsear los textos, se creó una gramática muy simple que mediante expresiones regulares sustituye las ocurrencias más comunes de estas abreviaciones por las palabras correctamente escritas.

Finalmente, en afán de buscar patrones que pudiesen ser utilizados en los features y facilitar la tarea de los parsers se sustituyen distintos formatos de horarios por el tag <HORARIO>, así como ocurrencias de números enteros por el tag <NÚMERO>.

4.2 Decisiones sobre la medición del desempeño de los clasificadores

Antes de construir los clasificadores que trabajan sobre los tweets preprocesados, es importante definir un proceso y un conjunto de métricas que permitan evaluar el desempeño de los clasificadores de forma confiable y comparable.
En esta sección se detallan la partición del conjunto de tweets en un corpus de entrenamiento y otro de testeo o prueba. Adicionalmente, se especifica la forma de calcular las métricas que permitan evaluar y comparar el desempeño de los clasificadores.

4.2.1 Partición del corpus en entrenamiento y testeo

Particionar el corpus completo de datos disponibles (en este caso tweets) en un corpus de entrenamiento y otro de testeo es una práctica estándar en la resolución de problemas de aprendizaje automático. El corpus de entrenamiento se utiliza para formar la hipótesis de aprendizaje, y el corpus de testeo se utiliza para evaluar cómo se desempeñará dicha hipótesis sobre datos subsiguientes que no forman parte de los datos conocidos [30].

Siguiendo esta práctica, tomamos el conjunto de los tweets con clasificación no ambigua (cuentan con una clasificación con mayoría de votos) y conformamos el corpus de testeo con un 20% de los tweets de dicho conjunto, tomados aleatoriamente. El 80% de los tweets restantes se utiliza para conformar el corpus de entrenamiento. Se comprobó que la composición del corpus de testeo es representativa de la composición del corpus completo. El mismo consta de los siguientes datos:

- Total de tweets en el corpus de testeo: 526
- Porcentaje de tweets positivos en el corpus de testeo: 26%
- Porcentaje de tweets negativos en el corpus de testeo: 26%
- Porcentaje de tweets que no son opiniones en el corpus de testeo: 48%

Por otro lado, para los clasificadores basados en reglas (el cual incluye la línea base), no tiene sentido realizar dicha separación ya que en nuestro caso se trata de algoritmos estáticos que no realizan un proceso de aprendizaje y por ende no tienen riesgo de sobreajuste. Para estos clasificadores, evaluar con un 20% del corpus solamente limita la confianza que tenemos sobre la generalización del clasificador a datos desconocidos.

4.2.2 Métricas de desempeño

Las métricas utilizadas para medir el desempeño de los clasificadores fueron Accuracy, Precision, Recall y Medida F1. Para poder definir cómo se calculan estas métricas, primero es necesario definir algunos conceptos. En un contexto de clasificación binaria (entre solamente 2 clases):

- Verdaderos Positivos (VP): Las muestras (opiniones en el caso de este proyecto) que son de la categoría que se está analizando y fueron correctamente clasificadas.
- Verdaderos Negativos (VN): Las muestras que no pertenecen a la categoría que se analiza y fueron correctamente clasificadas como no pertenecientes a ella.
- Falsos Positivos (FP): Las muestras que pertenecen a otra categoría pero fueron incorrectamente clasificadas como de la que se está analizando.
- Falsos Negativos (FN): Las muestras que no pertenecen a la categoría que fue analizada pero se clasificaron incorrectamente como pertenecientes a ella.

La medida de accuracy (acierto) es la más básica de las existentes: es el porcentaje de muestras correctamente clasificadas. Formalmente, se define como:

$$\frac{VP + VN}{VP + VN + FP + FN}$$

Precisión es la métrica que presenta la relación entre la cantidad de muestras correctamente clasificadas como de una clase con todas las que fueron clasificadas como pertenecientes a la misma. Formalmente se define como:

$$\frac{VP}{VP + FP}$$

Cuando se minimizan las FP, es decir cuanto más cercano al 1 sea el resultado, más certero y confiable es el clasificador.

Por lo general, esta métrica se acompaña del Recall, que presenta la relación entre la cantidad de muestras clasificadas correctamente como de una clase con la cantidad total de muestras de esa clase presente en el conjunto de test. Formalmente:

$$\frac{VP}{VP + FN}$$

Cuanto más cercano a 1 sea el recall, menor cantidad de ejemplos de una determinada clase serán "perdidos" al ser clasificados como de otra.

Precision y Recall son medidas complementarias. Para generar una medida única que permita obtener la información de ambas, se crea la medida F1 que se define como la media armónica de las anteriores:

$$F_{\beta} = (1 + \beta^2) * \frac{precision * recall}{\beta^2 * precision + recall}$$

El parámetro beta se utiliza para ponderar la importancia de una de las dos medidas sobre la otra.

En nuestro proyecto se trabaja con más de 2 clasificaciones posibles. El enfoque seguido por el grupo fue calcular precision, recall y medida F para cada clasificación posible (en formato uno contra todos) y luego para obtener una medida general se realiza un promedio ponderado por la cantidad de tweets en cada categoría.

Por ejemplo, el promedio ponderado de la precisión entre 3 clases (positivo, negativo y no opinión) se calcularía de la siguiente manera:

$$precision_{promedio\ ponderado} =$$
$$precision_{positivos} * \%_{positivos} + precision_{negativos} * \%_{negativos} + precision_{no\ opinión} * \%_{no\ opinión}$$
$$donde:$$
$$\%_{positivos} = \frac{cantidad\ de\ tweets\ positivos}{cantidad\ de\ tweets\ total}$$
$$\%_{negativos} = \frac{cantidad\ de\ tweets\ negativos}{cantidad\ de\ tweets\ total}$$
$$\%_{no\ opinión} = \frac{cantidad\ de\ tweets\ no\ opinión}{cantidad\ de\ tweets\ total}$$

Es importante destacar que, cuando trabajamos con clasificadores basados en aprendizaje automático, estas métricas siempre se calculan exclusivamente sobre corpus de testeo (incluyendo las cantidades de tweets de cada clase).

4.3 Construcción de la línea base

La línea base de nuestro clasificador se basa en un algoritmo sencillo, el cual suma la polaridad de todas las palabras de un tweet dado, apoyándose en un lexicón de polaridades semánticas a nivel de lemas.

4.3.1 Algoritmo

El algoritmo utilizado en la línea base suma la polaridad de cada palabra del tweet que pertenece al Lexicón Original. Existen cotas para determinar si el tweet es una opinión positiva, negativa o neutral según la suma de las polaridades de las palabras del mismo.

En caso de no encontrar ninguna palabra en el tweet cuyo lema pertenezca al lexicón, clasifica ese tweet como “no es opinión”.

En resumen, el algoritmo se puede expresar como:

$$opinión(tweet) =$$
$$No\ es\ opinión,\ si\ \neg \exists (palabra \in tweet) : \{palabra \in Lexicon\ Original\}$$
$$Positivo,\ si \sum_{palabra \in tweet} polaridad(palabra) \geq LIMITE_{positivo}$$

$$Negativo,\ si \sum_{palabra \in tweet} polaridad(palabra) \leq LIMITE_{negativo}$$

$$Neutral,\ si\ LIMITE_{negativo} < \sum_{palabra \in tweet} polaridad(palabra) < LIMITE_{positivo}$$

Para calcular la suma de las polaridades, solo consideramos las palabras que se encuentran en el Lexicón Original con un valor de confianza que se encuentre por encima de una cota determinada.

Una de las grandes ventajas de nuestra línea base es que es altamente parametrizable. A continuación se listan los parámetros utilizados:

- Límite positivo de polaridad = 0.05
- Límite negativo de polaridad = -0.05
- Límite de valor de confianza para la polaridad de un lema en el lexicón = 7
- Modo de clasificación = 3 categorías sin neutrales

Los resultados completos se muestran en la tabla 4.1 y su matriz de confusión en la tabla 4.2.

Clasificación	Precision	Recall	Medida F1
Positivo	52%	32%	40%
Negativo	60%	18%	28%
No es opinión	54%	87%	67%
Total (ponderado)	**55%**	**54%**	**49%**

Tabla 4.1: Resultados de la línea base

	Positivo	Negativo	No es opinión
Positivo	**216**	28	427
Negativo	85	**125**	491
No es opinón	112	54	**1091**

Tabla 4.2: Matriz de confusión de la línea base

Podemos observar que la línea base se desempeña por encima de un clasificador aleatorio, teniendo en cuenta que se debe clasificar entre 3 clases no balanceadas. Quizás la conclusión más importante de la línea base es la fuerte tendencia a clasificar los tweets como “no es opinión”. Luego de analizar los casos clasificados incorrectamente concluimos que la mayoría de los errores pueden atribuirse a al menos uno de los siguientes factores:

1. Falta de palabras locales (de la jerga propiamente uruguaya) en el Lexicón Original
2. Falta de una solución de lematización para encontrar el lema correcto de una palabra en el lexicón
3. No existe manejo de negación. Ejemplo: “no me gusta” se toma como positivo.

Todos estos problemas serán atacados más adelante en las secciones siguientes.

4.4 Incorporación de herramientas de análisis lingüístico

Con el objetivo de abordar los problemas encontrados en la línea base, se incorpora un conjunto de herramientas para utilizar tanto en clasificadores basados en reglas como en clasificadores de aprendizaje automático.

En esta sección detallaremos las herramientas y procesos de análisis morfológico y análisis sintáctico que fueron incorporadas con el objetivo de atacar los problemas de lematización y expansión del lexicón existente.

4.4.1 Uso de Freeling

En esta sección se detalla el uso de la herramienta Freeling, su ejecución y dificultades de la misma.

Ejecución de la herramienta

Se decidió utilizar la herramienta Freeling como base para obtener la estructura sintáctica de cada tweet. Esta herramienta permite obtener POS tags, lemas, árbol gramatical y constituyentes gramaticales de cada oración. Para ello se ejecuta Freeling en modo Shallow Parsing, desactivando la opción de Multiword Detection.

Inicialmente se ejecutó Freeling en batch sobre el texto crudo (sin preprocesamiento) de todo el corpus. Luego se ejecutó Freeling en batch sobre el texto de los tweets preprocesados, siendo este resultado ampliamente superior en calidad al anterior.

En las imágenes 4.3 y 4.4 se presenta un ejemplo de la salida obtenida de ejecutar Freeling con y sin preprocesamiento del tweet de la imagen 4.2.

Imagen 4.2- Tweet original

```
<sentences><sentence id="1">
  <token id="t1.1" form="Era_el_de_Chile" lemma="era_el_de_chile" tag="NP00000"
      ctag="NP" pos="noun" type="proper">
  </token>
  <token id="t1.2" form="!" lemma="!" tag="Fat" ctag="Fat" pos="punctuation"
      type="exclamationmark" punctenclose="close">
  </token>
  <token id="t1.3" form="!" lemma="!" tag="Fat" ctag="Fat" pos="punctuation"
      type="exclamationmark" punctenclose="close">
  </token>
  <token id="t1.4" form="!" lemma="!" tag="Fat" ctag="Fat" pos="punctuation"
      type="exclamationmark" punctenclose="close">
  </token>
  <constituents>
    <node label="S">
      <node label="sn">
        <node head="1" label="grup-nom-ms">
          <node head="1" label="w-ms">
            <node leaf="1" head="1" token="t1.1" word="Era_el_de_Chile"/>
          </node>
        </node>
      </node>
      <node label="F-term">
        <node leaf="1" head="1" token="t1.2" word="!"/>
      </node>
      <node label="F-term">
        <node leaf="1" head="1" token="t1.3" word="!"/>
      </node>
      <node label="F-term">
        <node leaf="1" head="1" token="t1.4" word="!"/>
      </node>
    </node>
  </constituents>
</sentence>
</sentences>
```

Imagen 4.3- Salida de freeling para el tweet presentado en la imagen anterior, sin preprocesamiento

```
<sentences><sentence id="1">
  <token id="t1.1" form="era" lemma="ser" tag="VSII3S0" ctag="VSI" pos="verb"
    type="semiauxiliary" mood="indicative" tense="imperfect" person="3"
    num="singular">
  </token>
  <token id="t1.2" form="el" lemma="el" tag="DA0MS0" ctag="DA" pos="determiner"
    type="article" gen="masculine" num="singular">
  </token>
  <token id="t1.3" form="de" lemma="de" tag="SP" ctag="SP" pos="adposition"
    type="preposition">
  </token>
  <token id="t1.4" form="chile" lemma="chile" tag="NCMS000" ctag="NC"
    pos="noun" type="common" gen="masculine" num="singular">
  </token>
  <constituents>
    <node label="S">
      <node label="grup-verb">
        <node head="1" label="verb">
          <node leaf="1" head="1" token="t1.1" word="era"/>
        </node>
      </node>
      <node label="sn">
        <node label="j-ms">
          <node leaf="1" head="1" token="t1.2" word="el"/>
        </node>
        <node head="1" label="grup-sp">
          <node head="1" label="prep">
            <node leaf="1" head="1" token="t1.3" word="de"/>
          </node>
          <node label="sn">
            <node head="1" label="grup-nom-ms">
              <node head="1" label="n-ms">
                <node leaf="1" head="1" token="t1.4" word="chile"/>
              </node>
            </node>
          </node>
        </node>
      </node>
    </node>
  </constituents>
</sentence>
</sentences>
```

Imagen 4.4- Salida de freeling para el tweet presentado en la imagen anterior, con preprocesamiento

Dificultades encontradas

Si bien en primera instancia consideramos que la ejecución de Freeling es una tarea sencilla, nos encontramos con una serie de dificultades que insumieron un tiempo considerable para ser salvaguardadas.

En primer lugar, conseguir que Freeling funcione localmente lleva un esfuerzo en tiempo considerable. Adicionalmente, fue necesario experimentar con varios paquetes externos para conseguir un wrapper que permita procesar la salida de Freeling desde Python [31], e incorporar dicho paquete con el ambiente de ejecución de los notebooks IPython.

Dado que es una tarea que se realiza frecuentemente en varias materias y proyectos del Grupo de Procesamiento de Lenguaje Natural de la Facultad de Ingeniería, decidimos agregar un anexo al final del documento con las instrucciones necesarias para instalar tanto Freeling como el wrapper de Python localmente.

En segundo lugar, la ejecución de Freeling sobre el corpus completo es una tarea cuya ejecución toma varias horas. Para evitar las significativas demoras al ejecutar esta tarea, se implementó una función que persiste cada ejecución de Freeling sobre un tweet en formato XML. De esta forma, Freeling se ejecuta dinámicamente para analizar un tweet sólo si no se encuentra un archivo XML cuyo nombre se corresponda con el ID del tweet. Esta funcionalidad también se aplica al realizar análisis de reputación.

Dentro de los entregables del proyecto se encuentra resultado de todas las ejecuciones de Freeling sobre el corpus y los tweets analizados para el análisis de reputación.

4.4.2 Lematización

En esta sección se detalla el proceso de lematización, herramientas y algoritmo utilizado.

Incorporación de nuevas herramientas

Luego de estudiar los casos incorrectamente clasificados por la línea base se llegó a la conclusión de que es necesario contar con una solución de lematización para maximizar la utilidad del lexicón existente. Nuestro primer acercamiento a resolver este problema fue utilizar un lematizador popular llamado pattern.es [32], disponible de forma abierta, desarrollado por la universidad de Antwerp, Bélgica. Luego de experimentar con varios lematizadores, arribamos a la conclusión de que no existe una solución abierta que cuente con buenos niveles de desempeño en general para el español, especialmente al tener en cuenta las variantes de vocabulario existentes en latinoamérica.

Nuestro enfoque para mejorar la performance del lematizador fue cubrir con Freeling los casos que pattern.es no cubría exitosamente. Se implementó un método que crea un diccionario de lemas a partir de todas las palabras existentes en las ejecuciones persistidas en XML de Freeling (ver sección anterior). El resultado fue un lematizador que trabaja en 4 niveles, obteniendo un nivel de desempeño estrictamente superior a pattern.es o freeling por construcción.

Funciona iterativamente de la siguiente manera:

1. Se busca la palabra sin lematizar en el lexicón
2. Si no se encuentra, se busca el lema de la palabra según el lematizador de pattern.es
3. Si no se encuentra, se busca el lema de la palabra en el diccionario pre-procesado de freeling
4. Si no se encuentra, se ejecuta freeling dinámicamente sobre esa palabra para obtener el lema

Algoritmo mejorado para obtener la polaridad de una palabra

Se implementa un algoritmo que obtiene la polaridad de una palabra utilizando el lematizador descrito en la sección anterior. Este algoritmo presenta una mejora sobre la búsqueda de polaridad de una palabra en el lexicón existente en la línea base, donde solamente se buscaban palabras de forma literal.

Podemos expresar este algoritmo de la siguiente manera:

$$
\begin{aligned}
&polaridad(palabra) = \\
&\quad 0,\ si\ lema(palabra) \notin Lexicón\ Extendido \\
&\quad 0,\ si\ confianza_{lexicón}(lema(palabra)) > LIMITE_{confianza} \\
&\quad polaridad_{lexicón}(lema(palabra)),\ si\ confianza_{lexicón}(lema(palabra)) \leq LIMITE_{confianza}
\end{aligned}
$$

Donde "$lema(palabra)$" se obtiene mediante el lematizador de 4 niveles descrito en la sección anterior.

4.5 Clasificadores basados en reglas gramaticales

En esta sección se describen los clasificadores basados en reglas gramaticales utilizados: el clasificador basado en árboles de estructura gramatical y el clasificador de ventana gramatical. También se detalla la construcción de un nuevo lexicón para la clasificación de no opiniones (Antisenticon).

4.5.1 Clasificador con herramientas de análisis lingüístico

Se incorporó la solución de lematización y el Lexicón Extendido al algoritmo de la línea base para medir el valor agregado de estas funcionalidades.

Recordamos que el algoritmo de la línea base suma la polaridad de las palabras del tweet que aparecen en el Lexicón Original. Las mejoras consisten en buscar los lemas de las palabras a partir del lematizador de 4 niveles descrito anteriormente (en lugar de buscar las palabras directamente en la forma existente en el tweet), y el crecimiento del vocabulario del lexicón.

	Precision	Recall	Medida F1
Positivo	52,0%	62,4%	56,7%
Negativo	59,2%	60,3%	59,2%
No es opinón	71,0%	62,6%	66,5%
Total (ponderado)	60,7%	61,7%	60,8%

Tabla 4.3: Resultados

	Positivo	Negativo	No es opinión
Positivo	**419**	83	169
Negativo	125	**423**	153
No es opinón	262	208	**787**

Tabla 4.4: Matriz de confusión

El resultado de agregar estas herramientas fue ampliamente positivo, mejorando significativamente todas las medidas de performance.

Es fácil ver que el ampliar el vocabulario y agregar lematización aumenta el porcentaje de palabras que son encontrada en el lexicón (senticon). Dado que el algoritmo considera que un tweet es una opinión si encuentra al menos 1 palabra en el lexicón, se da un aumento en el porcentaje de tweets que son considerados opiniones (positivas o negativas) y por ende una disminución de los tweets que son clasificados como “no es opinión”.

En otras palabras, se mejora la performance de las clasificaciones “positivo” y “negativo”, y disminuye la performance de la clase “no es opinión”, siendo el balance general ampliamente positivo.

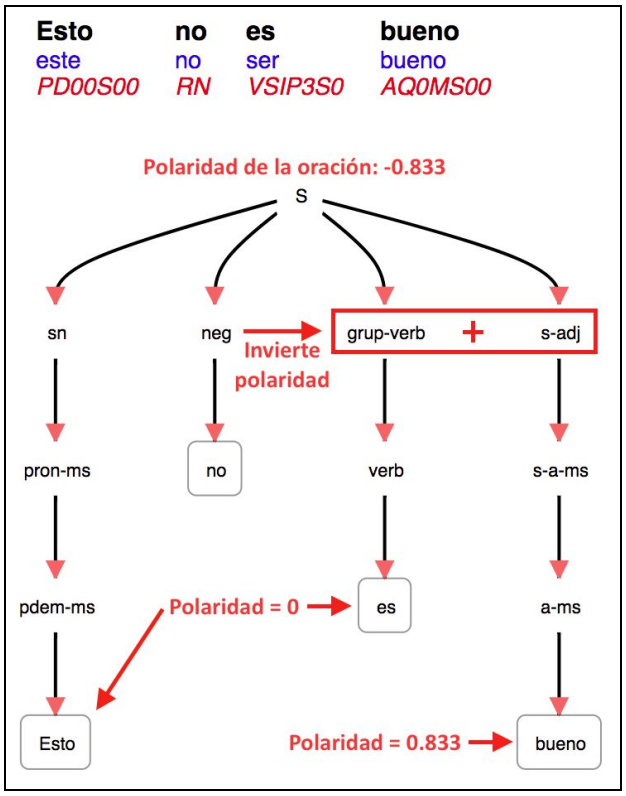
El problema principal pasa a ser ahora el gran número de falsos positivos para la clase “no es opinión”. Es decir, tweets que son incorrectamente clasificados como opiniones positivas o negativas cuando en realidad no son opiniones.

En las siguientes secciones se presenta una serie de medidas y 2 nuevos clasificadores, orientados a mejorar este punto y también orientados a mejorar la falta de manejo de recursos gramaticales tales como la negación, intensificación, etc.

4.5.2 Clasificador basado en árboles de estructura gramatical

Una de las ideas que queda pendiente desde las conclusiones de la línea base es incorporar el manejo de negación, conjunción e intensificadores. Nuestro primer intento de implementar un clasificador que incorpore estos conceptos se basa en el árbol de componentes gramaticales de un tweet proporcionado por Freeling.

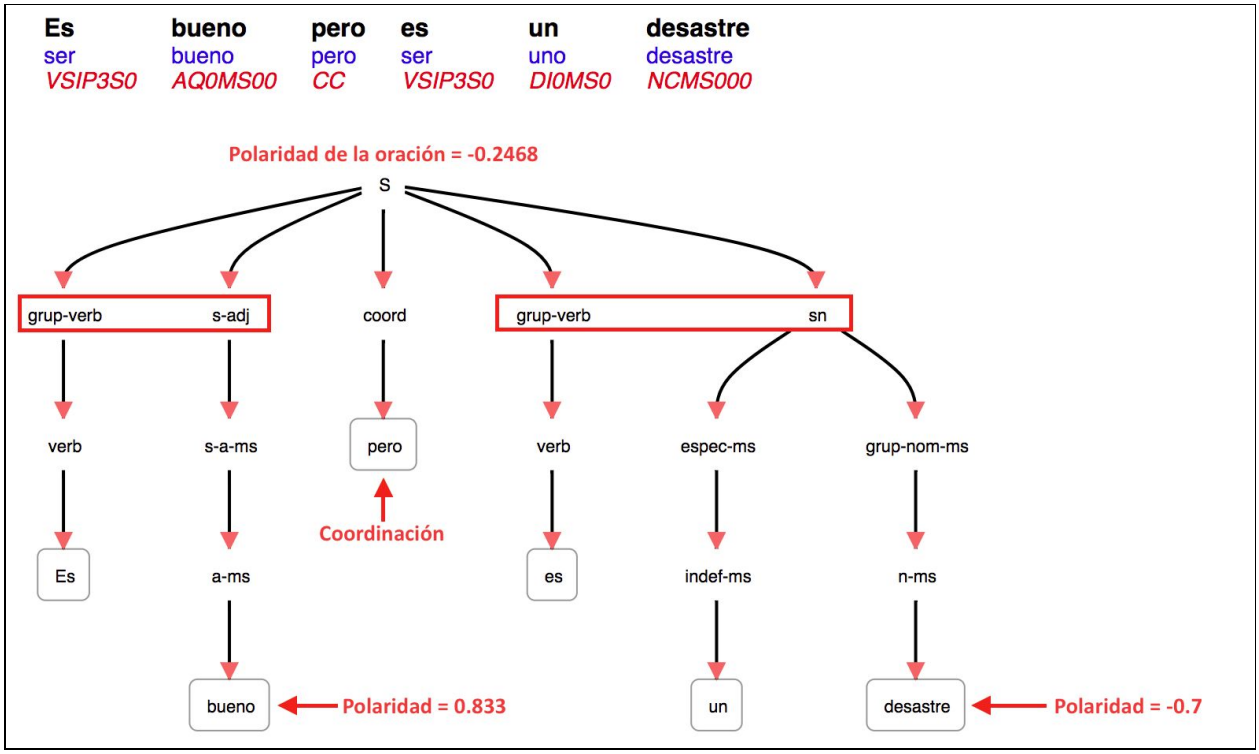
Se implementó un algoritmo recursivo que recorre el árbol de constituyentes gramaticales, invirtiendo las polaridades de los nodos que se encuentran a continuación de una negación, y dándole un peso mayor a los nodos que aparecen a la derecha de una conjunción. A continuación se presentan algunos ejemplos ilustrativos:



Imágen 4.5. Ejemplo: "esto no es bueno"

$$polaridad(S) = polaridad(sn) + (-1) * (polaridad(grup\_verb) + polaridad(s\_adj))$$
$$polaridad(S) = 0 + (-1) * (0 + 0.833) = -0.833$$
$$polaridad(grup\_verb) = polaridad(verb) = polaridad(verb) = polaridad(es) = 0$$
$$polaridad(s\_adj) = polaridad(s\_ams) = polaridad(a\_ms) = polaridad(bueno) = 0.833$$

Para calcular la polaridad de la conjunción coordinante "*pero*" nos inspiramos en el trabajo presentado en un proyecto de grado del año anterior [33], donde se le da un peso mayor a la parte derecha de la misma. En el ejemplo que aparece debajo, se multiplica la polaridad de la parte izquierda por 0.4 mientras que la polaridad de la parte derecha se multiplica por 0.6.



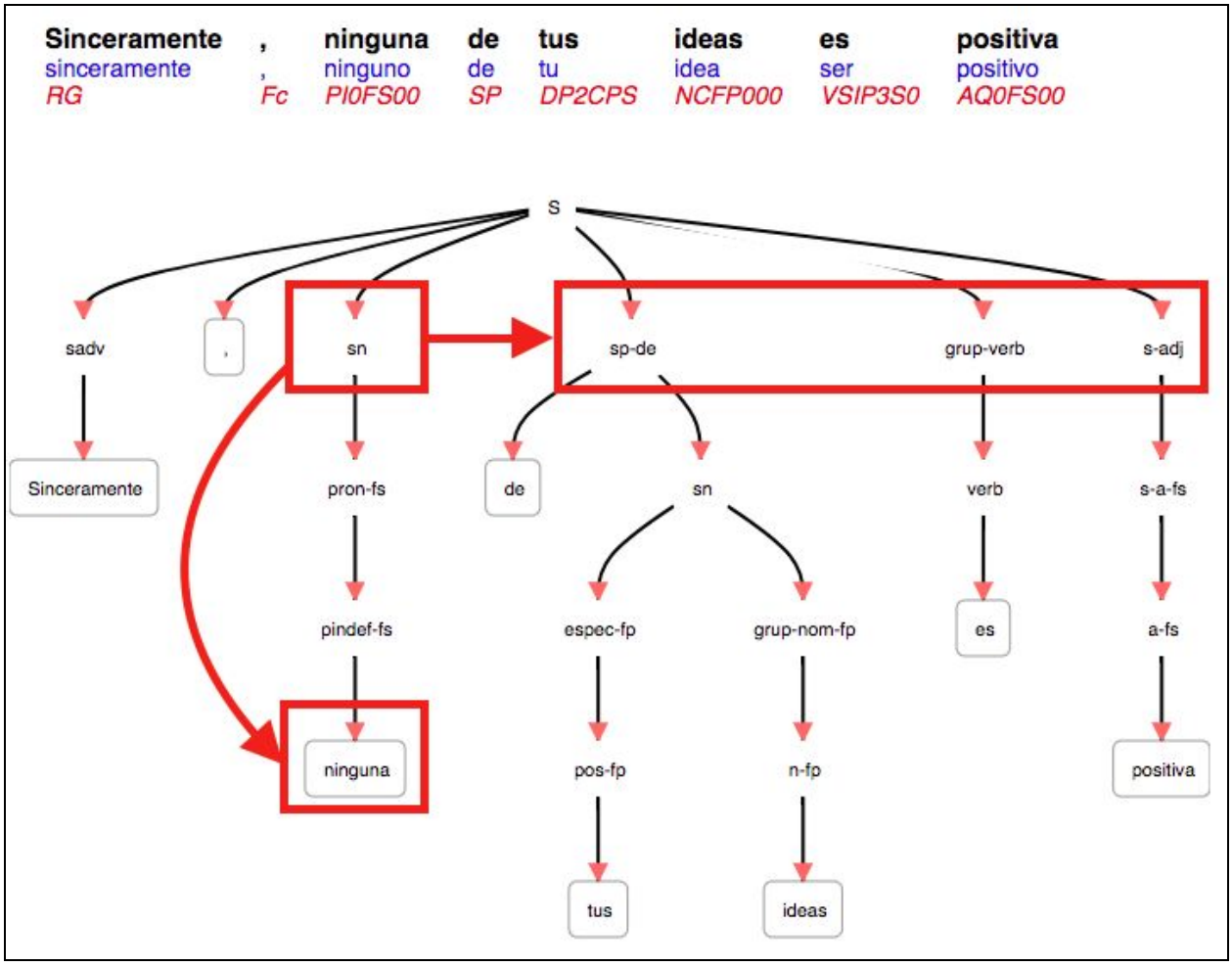
Imágen 4.6. Ejemplo: "Es bueno pero es un desastre"

$$polaridad(S) = (0.4)(polaridad(grup\_verb) + polaridad(s\_adj)) + (0.6)(polaridad(grup\_verb) + polaridad(sn))$$
$$polaridad(S) = (0.4)(0.833) + (0.6)(-0.7) = -0.2468$$

Cabe destacar que ambos ejemplos serían incorrectamente clasificados por todos los clasificadores presentados hasta el momento en este documento.

Dificultades encontradas

Una de las principales dificultades encontradas fue el hecho de que Freeling no siempre marca negaciones. Debajo puede visualizarse un ejemplo:



Imágen 4.7. Ejemplo: "Sinceramente, ninguna de tus ideas es positiva"

Como puede visualizarse, la palabra "ninguna" actúa como negación. Sin embargo, freeling no utiliza un tag "neg". Aún si contamos con un diccionario de palabras que marcan negación, no es claro hasta dónde llega la misma ya que falta el tag "neg" para desambiguar.

Para resolver este problema, optamos por adoptar las siguientes convenciones:

- Tomamos las palabras "no", "sin", "faltaba", "nunca", "jamás", "ningún", "ninguna" y "tampoco" como negadores.
- La negación comienza desde el nodo ancestro más alto que cuenta con un único hijo. De esta forma, tomamos que el alcance de la negación va desde "sn" y no desde el comienzo de la oración (ya que el comienzo de la oración tiene más de un nodo hijo).

Los resultados de la evaluación del sistema basado en el árbol de constituyentes se muestran en la tabla 4.5 y su matriz de confusión en la tabla 4.6.

Clasificación	Precision	Recall	Medida F1
Positivo	49,5%	72,9%	59,0%
Negativo	55,7%	61,3%	58,4%
No es opinón	79,5%	55,0%	65,0%
Total (ponderado)	61,5%	63,0%	60,8%

Tabla 4.5: Resultados de la evaluación del sistema basado en el árbol de constituyentes

	Positivo	Negativo	No es opinión
Positivo	**489**	87	95
Negativo	188	**430**	83
No es opinón	311	255	**691**

Tabla 4.6: Matriz de confusión del sistema basado en el árbol de constituyentes

Si bien en los casos de prueba manualmente creados por el grupo se constató una diferencia positiva, como puede apreciarse al ejecutar el clasificador sobre el corpus completo no se presentan mejoras comparadas al compararse con los resultados obtenido hasta el momento.

Resulta intuitivo pensar que el uso del árbol sintáctico no obtenga buenos resultados por estar trabajando con tweets, que son generalmente son textos mal escritos en donde muchas veces no hay oraciones correctas.

Es interesante destacar que el desempeño del clasificador cambia de forma significativa entre distintas clases, pero los resultados generales son equivalentes.

4.5.3 Creación de un nuevo lexicón (Antisenticon)

Como se mencionó anteriormente, luego de las mejoras realizadas al Lexicón Original se presentó un aumento en el porcentaje de tweets incorrectamente clasificados como opiniones positivas o negativas cuando en realidad no son opiniones.

Adicionalmente, se realizó un estudio de los casos cuya clasificación cambió luego de la incorporación de herramientas de análisis lingüístico. Notamos que la clasificación incorrecta de tweets que en realidad no son opiniones se debe a un hecho concreto: el Lexicón Extendido aporta mucha información para

discernir entre opiniones positivas y negativas, pero no aporta directamente información para identificar tweets que no son opiniones. Solamente clasifica que un tweet no es opinión por omisión.

Por estas razones surge la necesidad de la creación del Antisenticon mencionado en la sección 3.2.3 para poder discriminar los tweets que no son opinión.

4.5.4 Clasificador Ventana Gramatical

En esta sección se detalla un nuevo clasificador el cual llamamos "de ventana gramatical". Se explica el algoritmo, ejemplos, resultados y se presentan conclusiones del mismo.

Algoritmo

Como pudo observarse anteriormente, el clasificador basado en árboles de estructura gramatical no ofreció buenos resultados, presentando una mejora ínfima y marginal al compararse con los resultados que se habían obtenido en ese momento. Sin embargo, consideramos que incorporar conceptos tales como el manejo de negación e intensificadores es un concepto clave, y por ende decidimos buscar otra forma de implementar la misma idea sin caer en las dificultades encontradas hasta el momento.

Siguiendo las ideas presentadas en la sección anterior, implementamos un clasificador que introduce el manejo de recursos sintáctico-gramaticales pero utilizando una ventana de tamaño discreto en lugar de árboles.

De esta forma implementamos un algoritmo que clasifica tweets de la siguiente forma:

1. Si se cumple alguno de los patrones del Antisenticon o no contiene ninguna palabra cuyo lema pertenezca al Lexicón Extendido, entonces el tweet se clasifica como "no es opinión".
2. De lo contrario, suma la polaridad de los lemas de las palabras del tweet:
    - Para calcular la polaridad de cada palabra x : $\{palabra(x) \wedge x \in tweet\}$, se observan las n palabras anteriores a x del tweet, donde n es el tamaño de ventana.
    - Nótese que la ventana de una palabra incluye solamente las palabras anteriores. Es decir, si un tweet está compuesto por m palabras $(x_1, x_2, .., x_i, ...x_m)$, la ventana de x_i está compuesta por las palabras $(x_{i-n}, x_{i-n+1}, ..., x_{i-1})$
    - Por cada palabra de la ventana que sea un negador o un intensificador, se multiplica la polaridad de x por un valor asociado a ese negador o intensificador. Los valores correspondientes se encuentran especificados en la tabla debajo.

- La lista de negadores utilizados está compuesta por: "no", "faltaba", "jamás", "ningún", "sin", "ninguna", "tampoco" y "nadie".
- La lista de intensificadores utilizados está compuesta por: "muy", "demasiado", "bastante", "completamente", "algo", "poco", "sumamente", "ligeramente", "nada" y "nunca".

A continuación se presentan algunos ejemplos del clasificador de ventana gramatical que ayudan a visualizar su funcionamiento. El primero es “La verdad es que esto no me gusta” y puede apreciarse en la imágen 4.8. Se clasifica como negativo si tomamos un tamaño de ventana igual a 3. El procedimiento de clasificación es el siguiente:

1. No coincide con ningún patrón del Antisenticon, pero contiene una palabra cuyo lema (gustar) se encuentra en el lexicón, y por ende es una opinión.
2. Para descifrar si la opinión es positiva o negativa, se suma la polaridad de todas las palabras del tweet, tomando en cuenta los multiplicadores de las palabras que aparecen en sus ventanas correspondientes.
    a. La palabra “verdad” aparece en el lexicón con polaridad 0.3. Su ventana solamente contiene la palabra “la”, la cual no tiene asociado un multiplicador.
    b. El lema de la palabra “gusta” (gustar) aparece en el lexicón con polaridad 0.477. Su ventana contiene las palabras “esto”, “no” y “me”. La palabra “no” tiene un multiplicador de -1 asociado, por lo cual la polaridad total de la palabra “gusta” es de $(-1)(0.477) = -0.477$
    c. La suma de las polaridades resultante es:
    $polaridad(\text{"}La\ verdad\ es\ que\ esto\ no\ me\ gusta\text{"} \mid ventana = 3) = 0.3 - 0.477 = -0.177$.
    d. Dado que la polaridad es negativa, el tweet se clasifica como “opinión negativa”.

Imágen 4.8. Ejemplo: “La verdad es que esto no me gusta”

En el siguiente ejemplo de la figura 4.9 podemos visualizar el uso de tanto intensificación (“completamente”) como negación (“no”). Dado que la polaridad es negativa, este tweet se clasificaría como “opinión negativa”.

Imágen 4.9. Ejemplo: “Esto no es completamente bueno”

$polaridad(\text{"}Esto\ no\ es\ completamente\ bueno\text{"} \mid ventana = 3) = (-1)(1.5)(0.833) = -2.2495$

El tamaño de ventana puede configurarse dentro de las constantes existentes en el módulo 1 (carga de datos), mientras que los negadores e intensificadores pueden configurarse mediante archivos .csv que incluyen los valores para sus multiplicadores asociados.

Se experimentó con varios valores de configuración posible para el tamaño de ventana, encontrándose que se obtienen buenos resultados con un valor de 3. Adicionalmente, existe la opción de utilizar un tamaño de ventana distinto para intensificadores y negadores, aunque por defecto no se utiliza.

Se obtuvieron conclusiones interesantes en cuanto a los multiplicadores asociados a intensificadores y negadores, ya que algunos valores contra-intuitivos funcionan mejor. Por ejemplo, si bien intuitivamente el multiplicador asociado a "no" debería ser -1, se registraron mejores resultados para valores con un valor absoluto menor, tales como -0.5 o -0.2. Este mismo fenómeno se repite para otros negadores.

Por otro lado, los intensificadores toman valores mucho más intuitivos. Por ejemplo, intensificadores tales como "bastante" o "demasiado" tienen asociados multiplicadores con un valor positivo y mayor a 1, mientras que otros como "poco" o "algo" toman valores positivos entre 0 y 1.

Con la configuración por defecto, los resultados obtenidos se muestran en la tabla 4.10 y su matriz de confusión en la tabla 4.11:

Clasificación	Precision	Recall	Medida F1
Positivo	54,8%	70,5%	61,7%
Negativo	59,7%	65,6%	62,5%
No es opinón	78,2%	62,0%	69,2%
Total (ponderado)	64,2%	66,0%	64,4%

Tabla 4.10: Resultados clasificador Ventana Gramatical

	Positivo	Negativo	No es opinión
Positivo	**473**	77	121
Negativo	145	**460**	96
No es opinón	245	233	**779**

Tabla 4.11: Matriz de confusión clasificador Ventana Gramatical

Podemos observar que se obtuvo mejoras de alrededor de un 15% en comparación con la línea base inicial, y de un 5% en comparación con el segundo clasificador basado en la línea base que incorpora herramientas de análisis lingüístico.

Es importante recordar que los valores de los multiplicadores (intensificadores y negadores) y los patrones existentes en el Antisenticon fueron estimados de forma sencilla. En la sección de trabajo a futuro se propone extender y enriquecer estos recursos mediante procedimientos de análisis formal.

4.6 Clasificación basada en aprendizaje automático

Para los experimentos realizados con aprendizaje automático tomamos en cuenta los clasificadores más utilizados en las investigaciones con objetivos similares al nuestro. Luego del proceso de ingeniería de features, se implementan 6 clasificadores basados en aprendizaje automático. Dado que en el capítulo 2 "Marco de trabajo" en la sección 2.4.2, "Métodos basados en aprendizaje automático", ya explicamos el concepto teórico detrás de cada uno de ellos, en las siguientes secciones nos limitaremos a explicar el proceso por el cual seleccionamos los mejores features en cada caso. Presentamos mas adelante los ajustes realizados a cada clasificador, los resultados obtenidos y las conclusiones.

4.6.1 Ingeniería de features

La ingeniería de features intenta aumentar la eficacia predictiva de los algoritmos de aprendizaje creando características de los datos sin procesar que facilitan el proceso de aprendizaje. Normalmente, la ingeniería de features se aplica primero para generar características adicionales, y a continuación se realiza el paso de selección de features para eliminar features irrelevantes, redundantes o altamente correlacionados.

Para la creación de features nos basamos en trabajos como "*Sentiment analysis of Twitter data*" [34] y el proyecto de grado "*Determinación de la orientación semántica de las opiniones transmitidas en textos de prensa*" [33].

Los features creados se pueden dividir en 4 categorías:

- Features numéricos discretizados: son aquellos que se crean a partir de conteos de ocurrencias de palabras positivas y negativas en el corpus, tomando como referencia para definir el sentimiento de una palabra los distintos lexicones utilizados.
- Features gramaticales: son aquellos basados en la información que puede obtenerse utilizando un analizador sintáctico como freeling.
- Features obtenidos a partir del resultado de algoritmos de clasificación como los utilizados en los clasificadores basados en reglas.
- Features "Bag of words": son aquellos que se obtienen evaluando si una palabra o conjunto de palabras pertenece a uno de los lexicones utilizados.

Para construir los features numéricos discretizados, evaluamos si la cantidad de palabras positivas o negativas de un tweet dado se encuentra dentro de un rango de 10% al compararlo con el promedio de todos los tweets tomados del corpus. La cantidad de palabras positivas o negativas puede ser tomada tanto desde el Lexicón Extendido o del lexicón "ELH".

Al tomar todos los posibles casos, se crean los siguientes atributos:

- La cantidad de palabras positivas presentes en “senticon” es mayor al promedio
- La cantidad de palabras positivas presentes en “senticon” se encuentra dentro del promedio
- La cantidad de palabras positivas presentes en “senticon” es menor al promedio
- La cantidad de palabras negativas presentes en “senticon” es mayor al promedio
- La cantidad de palabras negativas presentes en “senticon” se encuentra dentro del promedio
- La cantidad de palabras negativas presentes en “senticon” es menor al promedio
- La cantidad de palabras positivas presentes en “ELH” es mayor al promedio
- La cantidad de palabras positivas presentes en “ELH” se encuentra dentro del promedio
- La cantidad de palabras positivas presentes en “ELH” es menor al promedio
- La cantidad de palabras negativas presentes en “ELH” es mayor al promedio
- La cantidad de palabras negativas presentes en “ELH” se encuentra dentro del promedio
- La cantidad de palabras negativas presentes en “ELH” es menor al promedio

Observamos que tweets de tamaño mayor naturalmente tienden a contar con valores mayores al promedio de palabras positivas y/o negativas. Para normalizar esta observación, se agregan features que evalúan la proporción de palabras positivas o negativas del tweet (normalizando por tamaño) y la comparan con el promedio tomado dentro de los tweets del corpus. Tomando como referencia el Lexicón Extendido, se crean los siguientes atributos:

- La proporción de palabras negativas (cant negativas / cant total) es mayor al promedio
- La proporción de palabras negativas (cant negativas / cant total) se encuentra dentro del promedio
- La proporción de palabras negativas (cant negativas / cant total) es menor al promedio
- La proporción de palabras positivas (cant positivas / cant total) es mayor al promedio
- La proporción de palabras positivas (cant positivas / cant total) se encuentra dentro del promedio
- La proporción de palabras positivas (cant positivas / cant total) es menor al promedio

Features que evalúan si el tweet contiene una cantidad mayor, igual o menor de palabras y retweets en comparación al promedio de los tweets del corpus:

- Cantidad de palabras en el tweet es mayor al promedio
- Cantidad de palabras en el tweet dentro del promedio
- Cantidad de palabras en el tweet es menor al promedio
- Cantidad de retweets del tweet es mayor al promedio
- Cantidad de retweets del tweet dentro del promedio
- Cantidad de retweets del tweet es menor al promedio

Por otro lado, y con el objetivo de identificar los textos que no son opiniones, se analizaron ejemplos de tweets y se pudo apreciar que en algunos casos las personas suelen escribir de cierta forma cuando están opinando y de otra cuando no lo hacen. Con el propósito de identificar este tipo de ocurrencias se crean features gramaticales extraídos a partir de realizar el POS tagging que nos brinda freeling.

Los features gramaticales son:

- Cantidad de verbos en tiempo presente
- Cantidad de verbos en tiempo pasado
- Cantidad de verbos en tiempo futuro
- Cantidad de verbos en tiempo condicional
- Cantidad de verbos en tiempo imperfecto
- Cantidad de adverbios de tipo general
- Cantidad de adverbios de tipo negativo
- Cantidad de nombres de tipo común
- Cantidad de verbos de tipo auxiliar
- Cantidad de verbos de tipo semi auxiliar
- Cantidad de verbos de tipo principal
- Cantidad de verbos en modo subjuntivo
- Cantidad de verbos en modo imperativo
- Cantidad de verbos en modo indicativo
- Cantidad de verbos en modo participio
- Cantidad de verbos en modo infinitivo
- Cantidad de gerundios
- Cantidad de adjetivos de tipo posesivo
- Cantidad de adjetivos de tipo ordinal
- Cantidad de adjetivos de tipo calificativo
- Cantidad de adjetivos con grado superlativo
- Cantidad de adjetivos con grado evaluativo

Features que evalúan el sentimiento del tweet a partir de los algoritmos y clasificadores de reglas gramaticales:

- Sentimiento dado por el algoritmo de “Ventana Gramatical”
- Sentimiento dado por el algoritmo de “Reglas Gramaticales”
- Sentimiento dado por el algoritmo del “Clasificador con herramientas de análisis lingüístico”.

Features “Bag of words”:

- Contiene entradas del Antisenticon: resultado de evaluar si alguno de los conjuntos de palabras pertenecientes al Antisenticon pertenece al tweet.
- Bag of words Antisenticon: se agrega un feature por cada entrada del Antisenticon, indicando si el conjunto de palabras dado por la entrada del Antisenticon pertenece o no al tweet.
- Bag of words Senticon: se agrega un feature por cada palabra perteneciente al Lexicon Extendido, indicando si el tweet la contiene.

Un problema que enfrentamos fue que algunos algoritmos como Naive Bayes utilizan categorías discretas como valores de los features. Esto es un inconveniente si se utilizan valores numéricos ya que este tipo de algoritmo no toma en cuenta ordenamiento o ponderación entre las categorías. Es decir que si tomamos como ejemplo el feature "cantidad de palabras positivas" en un tweet, y tenemos 3 tweets: uno con 1 palabra positiva otro con 4 palabras positivas y el tercero con 10 palabras positivas, para Naive Bayes estos valores serán diferentes en igual medida. Nótese que en realidad es claro que es mucho más probable que los tweets con 4 y 10 palabras positivas sean positivos que uno que solo contiene una sola palabra positiva. Sin embargo, la implementación de Naive Bayes no permite realizar esta distinción.

Para solucionar este tipo de problemas se optó por utilizar categorías discretas como valores de los features. A partir de los datos del corpus se calcularon los promedios de todas las variables numéricas, y se crearon 3 categorías: mayor al promedio, dentro del promedio y menor al promedio. Para el caso de la categoría "dentro del promedio" se toma una cota inferior y superior del 10% del valor promedio calculado.

Siguiendo con el ejemplo anterior, una vez discretizados los features si el promedio de palabras positivas por tweet es 1, entonces los tweets con 4 y 10 palabras positivas tendrán categoría "superior al promedio". El restante tendrá categoría "dentro del promedio". De esta forma se logra el objetivo de diferenciar de forma adecuada los tweets con distinta cantidad de features numéricos.

Otra mejora que realizamos luego de la discretización fue binarizar los features, es decir creamos un feature por categoría indicando si esa categoría es o no el valor del feature sin binarizar.

Tomando ejemplo anterior, para el tweet con 10 palabras positivas se crean 3 features: "cantidad de palabras positivas menor al promedio" = falso, "cantidad de palabras positivas dentro del promedio" = falso, "cantidad de palabras positivas mayor al promedio" = verdadero.

Seleccion de features

Luego de crear todos los features, se debió seleccionar aquellos que aportan la información más valiosa para entrenar al clasificador (sobre aquellos que pueden aportar información no tan importante o redundante). Es importante mencionar que la selección de features se realizó de forma independiente en cada clasificador, y se presentará junto con los resultados de los mismos más adelante. Sin embargo a continuación adelantamos algunos ejemplos de por qué el conjunto de features finalmente seleccionados en cada clasificador es muy parecido.

Algunos features como "*cantidad de palabras positivas*", y "*proporción de palabras positivas*" aportan el mismo tipo de información, por lo que es innecesario o hasta perjudicial utilizar ambos en el entrenamiento. Otros features tales como "*cantidad de retweets*", si bien indican la popularidad de un tweet, no dan indicio de la polaridad de este. Es decir: no aportan información útil a la hora de entrenar. Lo mismo sucede con features como "*cantidad de palabras en el tweet*".

Por otra parte el feature "*Senticon Bag of Words*" utiliza demasiada memoria a la hora de entrenar (el Lexicón Extendido cuenta con 12.000 palabras aproximadamente, por lo que se crean 12.000 features en cada tweet utilizado para entrenar). Por este motivo resultó inviable utilizarlo en la práctica.

El feature "*Antisenticon bag of words*" crea un feature por cada entrada del Antisenticon indicando si esta entrada pertenece o no al tweet. Dado que, en relación al tamaño del corpus, muy pocos tweets presentan mas que una entrada del Antisenticon, la información aportada por estos features no tiene el peso suficiente como para que el clasificador aprenda con ellos. Es por esto que se agrupan estos features, utilizando un único feature con el resultado de evaluar si alguno de los conjuntos de palabras pertenecientes al Antisenticon pertenece al tweet. De esta forma se unen varios features que aportan información importante pero sin datos suficiente como para que el clasificador aprenda, creando un nuevo feature que aporta casi la misma información y aparece en mayor medida en comparación a los features creados por "Antisenticon bag of words".

Otros features que brindan información interesante son aquellos relacionados a los emoticones. Lamentablemente, al igual que en el caso de "Antisenticon bag of words" no contamos con suficientes tweets con emoticones en el corpus como para hacer pesar la información brindada por el feature. Para compensar esto, optamos por agregar los emoticones a nuestro lexicón, utilizándolos como palabras con polaridad, tal como se explica anteriormente. De esta forma la información brindada por el sentimiento de los emoticones es añadida a los features que corresponden a sentimientos de palabras.

Para seleccionar el mejor subgrupo de features utilizamos una técnica llamada Eliminación Recursiva de Features (RFE según sus siglas en inglés) que consiste en tomar iterativamente conjuntos de combinaciones de features para evaluarlos, compararlos y descartar aquellos que son menos útiles a la hora de entrenar. El proceso se repite con conjuntos de features cada vez más pequeños hasta llegar al número de features deseado. Esta técnica se aplicó para seleccionar los features de cada clasificador por separado. Los resultados del algoritmo fueron obtenidos a partir de utilizar cross validation en 10 iteraciones sobre el corpus de entrenamiento.

4.6.2 Naive Bayes

Naive Bayes fue el primer clasificador basado en aprendizaje automático con el que experimentamos. Esto no solo se debe a que la mayoría de los trabajos relacionados que analizamos utilizan este clasificador, sino que adicionalmente nos encontramos con una gran cantidad de información y tutoriales al respecto. Todos estos factores fueron de gran ayuda para comenzar nuestra investigación. En esta sección presentaremos los resultados obtenidos con Naive Bayes.

Como mencionamos en la sección anterior, se utilizó RFE (Eliminación Recursiva de Features) para obtener el conjunto de features que brindan los mejores resultados. Dado que la implementación utilizada de Naive Bayes no soporta el uso del algoritmo provisto por NTLK para realizar RFE, se optó por implementar esta técnica manualmente, comparando los resultados de distintas combinaciones de features y obteniendo como resultado los features que brindan información más significativa al clasificador.

Con los siguientes 10 features se maximiza la performance del clasificador (eliminando el resto):

- Si la cantidad de palabras negativas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras negativas presentes en el lexicón ELH son mayores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son mayores al promedio
- La proporción de palabras positivas (cant positivas / cant total) es menor al promedio
- La proporción de palabras positivas (cant positivas / cant total) es mayor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es menor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es mayor al promedio
- Sentimiento dado por el algoritmo del “Clasificador con herramientas de análisis lingüístico”
- Si contiene entradas del Antisenticon

Los resultados completos se muestran en la tabla 4.12 y su matriz de confusión en la tabla 4.13.

Clasificación	Precision	Recall	Medida F1
Negativo	58%	74%	65%
No es opinón	80%	62%	70%
Positivo	62%	71%	66%
Total (ponderado)	**70%**	**67%**	**68%**

Tabla 4.12: Resultados del clasificador Naive Bayes

	Negativo	No es opinón	Positivo
Negativo	**100**	20	16
No es opinón	51	**156**	45
Positivo	21	19	**98**

Tabla 4.13: Matriz de confusión del clasificador Naive Bayes

Viendo los resultados podemos apreciar que los resultados de la medida F1 para las tres clases son parejos. La precisión de no opinión es bastante mayor a las otras, pero el recall menor. Esto puede indicar que el Antisenticon no se equivoca tanto cuando clasifica un tweet como no opinión, pero es incompleto (como ya se comentó en la sección 3.2.3), y eso resulta en un valor menor menor de recall.

4.6.3 Árboles de Decisión

Se implementó un clasificador de árboles de decisión, el cual recibe el conjunto de features que se da como salida de la sección de ingeniería de features. En esta sección se presentan los distintos ajustes de parámetros con los que se experimentó durante la creación del clasificador, sus resultados y conclusiones.

Como se mencionó en la sección de ingeniería de features, se realiza Eliminación Recursiva de Features (RFE) con el objetivo de eliminar los features que no aportan valor al clasificador. Luego de ejecutar el algoritmo RFE sobre un clasificador de árboles de decisión sencillo, se registraron los siguientes resultados mostrados en la imágen 4.14.

Imagen 4.14: Resultados de eliminación recursiva de features

De esta forma, se maximiza la performance con los siguientes 5 features (eliminando el resto):

- Si la cantidad de palabras negativas presentes en el lexicón ELH son menores al promedio
- La proporción de palabras negativas (cant negativas / cant total) es menor al promedio
- La proporción de palabras positivas (cant positivas / cant total) es menor al promedio
- Sentimiento dado por el algoritmo utilizado en el "Clasificador con herramientas de análisis lingüístico".
- Sentimiento dado por el algoritmo de "Ventana Gramatical"

Los resultados completos se muestran en la tabla 4.15 y su matriz de confusión en la tabla 4.16.

Clasificación	Precision	Recall	Medida F1
Negativo	57%	66%	61%
No es opinón	74%	69%	71%
Positivo	63%	61%	62%
Total (ponderado)	**67%**	**66%**	**66%**

Tabla 4.15: Resultados del clasificador de Árboles de Decisión

	Negativo	No es opinón	Positivo
Negativo	**90**	36	10
No es opinón	39	**174**	39
Positivo	28	26	**84**

Tabla 4.16: Matriz de confusión del clasificador de Árboles de Decisión

Viendo los resultados mostrados por este clasificador cabe destacar otra vez la buena precisión de la clase no opinión. Sin embargo, en este caso se dispara el valor de su medida F1 en comparación con los positivos y negativos por el bajo recall de estos.

4.6.4 SVM

Se implementó un clasificador de máquinas de vectores de soporte (SVM), el cual recibe el conjunto de features que se da como salida de la sección de ingeniería de features. En esta sección se presentan los distintos ajustes de parámetros con los que se experimentó durante la creación del clasificador, sus resultados y conclusiones.

Dado que la implementación utilizada de SVM tampoco soporta el uso del algoritmo RFE (Eliminación Recursiva de Features) provisto por NTLK para realizar esta técnica, se optó por implementar RFE manualmente, comparando los resultados de distintas combinaciones de features y obteniendo como resultado los features que brindan información más significativa al clasificador.

Con los siguientes 11 features se maximiza la performance del clasificador (eliminando el resto):

- Si la cantidad de palabras negativas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras negativas presentes en el lexicón ELH son mayores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son mayores al promedio
- La proporción de palabras positivas (cant positivas / cant total) es menor al promedio
- La proporción de palabras positivas (cant positivas / cant total) es mayor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es menor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es mayor al promedio
- Sentimiento dado por el algoritmo de “Ventana Gramatical”
- Sentimiento dado por el algoritmo del “Clasificador con herramientas de análisis lingüístico”
- Si contiene entradas del Antisenticon

Los resultados completos se muestran en la tabla 4.17 y su matriz de confusión en la tabla 4.18.

Clasificación	Precision	Recall	Medida F1
Negativo	61%	61%	61%
No es opinón	73%	71%	72%
Positivo	62%	64%	63%
Total (ponderado)	**67%**	**67%**	**67%**

Tabla 4.17: Resultados del clasificador SVM

	Negativo	No es opinón	Positivo
Negativo	**83**	37	16
No es opinón	34	**180**	38
Positivo	19	31	**88**

Tabla 4.18: Matriz de confusión del clasificador SVM

Viendo los resultados del clasificador podemos observar que son muy similares al de lo 2 clasificadores anteriores, apenas superior que el de Árbol de Decisión y levemente inferior al clasificador Naive Bayes, otra vez cabe destacar los buenos resultados de la clase no opinión.

4.6.5 KNN

Se implementó un clasificador de K-vecinos más cercanos (KNN), el cual recibe el conjunto de features que se da como salida de la sección de ingeniería de features. En esta sección se presentan los distintos ajustes de parámetros con los que se experimentó durante la creación del clasificador, sus resultados y conclusiones.

Al igual que Naive Bayes y SVM, la implementación utilizada por KNN tampoco no soporta el uso del algoritmo RFE (Eliminación Recursiva de Features) provisto por NTLK para realizar esta técnica. También se optó por implementar RFE manualmente.

Con los siguientes 10 features se maximiza la performance del clasificador (eliminando el resto):

- Si la cantidad de palabras negativas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras negativas presentes en el lexicón ELH son mayores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son mayores al promedio
- La proporción de palabras positivas (cant positivas / cant total) es menor al promedio
- La proporción de palabras positivas (cant positivas / cant total) es mayor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es menor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es mayor al promedio
- Sentimiento dado por el algoritmo del “Clasificador con herramientas de análisis lingüístico”
- Si contiene entradas del Antisenticon

Los resultados completos se muestran en la tabla 4.19 y su matriz de confusión en la tabla 4.20.

Clasificación	Precision	Recall	Medida F1
Negativo	57%	65%	61%
No es opinón	73%	72%	73%
Positivo	62%	55%	58%
Total (ponderado)	**66%**	**66%**	**66%**

Tabla 4.19: Resultados del clasificador KNN

	Negativo	No es opinón	Positivo
Negativo	**89**	35	12
No es opinón	37	**181**	34
Positivo	31	31	**76**

Tabla 4.20: Matriz de confusión del clasificador KNN

Viendo los resultados de este clasificador observamos que sigue sin variar mucho los resultados con los clasificadores vistos anteriormente, en este caso el clasificador KNN obtuvo resultados ligeramente más bajos. Nuevamente cabe destacar que los resultados de la clase no opinión son sumamente positivos.

4.6.6 Regresión Logística

Se implementó un clasificador de regresión logística, el cual recibe el conjunto de features que se da como salida de la sección de ingeniería de features. En esta sección se presentan los distintos ajustes de parámetros con los que se experimentó durante la creación del clasificador, sus resultados y conclusiones.

Como se mencionó en la sección de ingeniería de features, se realiza Eliminación Recursiva de Features (RFE) con el objetivo de eliminar los features que no aportan valor al clasificador. Luego de ejecutar el algoritmo RFE sobre un clasificador de árboles de decisión sencillo, se registraron los siguientes resultados:

Imagen 4.21: Resultados de eliminación recursiva de features

De esta forma, se maximiza la performance con los siguientes 11 features (eliminando el resto):

- Si la cantidad de palabras negativas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras positivas presentes en el lexicón ELH son menores al promedio
- Si la cantidad de palabras negativas presentes en el lexicón ELH son mayores al promedio
- Si la cantidad de palabras del tweet es mayor al promedio.
- La proporción de palabras positivas (cant negativas / cant total) es menor al promedio
- La proporción de palabras positivas (cant negativas / cant total) es mayor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es menor al promedio
- La proporción de palabras negativas (cant negativas / cant total) es mayor al promedio
- Si la cantidad de retweets es mayor al promedio
- Sentimiento dado por el algoritmo de “Ventana Gramatical”
- Si contiene entradas del Antisenticon

Los resultados completos se muestran en la tabla 4.22 y su matriz de confusión en la tabla 4.23.

Clasificación	Precision	Recall	Medida F1
Negativo	62%	62%	62%
No es opinón	73%	70%	71%
Positivo	62%	66%	64%
Total (ponderado)	**67%**	**67%**	**67%**

Tabla 4.22: Resultados del clasificador de Regresión Logística

	Negativo	No es opinón	Positivo
Negativo	**84**	37	15
No es opinón	35	**177**	40
Positivo	13	30	**91**

Tabla 4.23: Matriz de confusión del clasificador Regresión Logística

Viendo los resultados del clasificador cabe destacar que se mantiene la pequeña variación de resultados con los clasificadores anteriores. Se obtiene el mismo promedio ponderado que SVM y se mantienen los buenos resultados en la clasificación de las no opiniones.

4.6.7 Redes Neuronales

Se implementó un clasificador de redes neuronales, el cual recibe la totalidad del conjunto de features que se da como salida de la sección de ingeniería de features dada la naturaleza del algoritmo. En esta sección se presentan los distintos ajustes de parámetros con los que se experimentó durante la creación del clasificador, sus resultados y conclusiones.

Los resultados completos se muestran en la tabla 4.24 y su matriz de confusión en la tabla 4.25.

Clasificación	Precision	Recall	Medida F1
Negativo	60%	60%	62%
No es opinón	74%	74%	74%
Positivo	63%	67%	65%
Total (ponderado)	**67%**	**69%**	**69%**

Tabla 4.24: Resultados del clasificador de Redes Neuronales

	Negativo	No es opinón	Positivo
Negativo	**82**	37	17
No es opinón	29	**186**	37
Positivo	18	27	**93**

Tabla 4.25: Matriz de confusión del clasificador Redes Neuronales

Viendo los resultados de este clasificador y en comparación con los anteriores podemos observar es el que obtuvo mejores valores de recall y medida F1, siendo levemente superior que el resto de los clasificadores.

4.6.8 Resultados y conclusiones

En la tabla 4.26 se muestra el resumen de los resultados obtenidos por los clasificadores basados en aprendizaje automático.

Clasificador	Precision (promedio)	Recall (promedio)	Medida F1 (promedio)
Naive Bayes	**70%**	67%	68%
Árbol de Decisión	67%	66%	66%
SVM	67%	67%	67%
KNN	66%	66%	66%
Regresión Logística	67%	67%	67%
Redes Neuronales	67%	**69%**	**69%**

Tabla 4.26: Resumen de resultados

Es fácil ver que los resultados de los resultados de todos los clasificadores son similares, encontrándose en $\pm 2\%$ de diferencia de performance. El clasificador basado en Naïve Bayes obtuvo un valor ligeramente mayor de precisión mientras que el clasificador basado en Redes Neuronales obtuvo mejores valores de recall y medida F1.

Al comparar estos resultados con los obtenidos por los clasificadores basados en reglas, podemos concluir que la incorporación de clasificadores de aprendizaje automático fue positiva, con un incremento de performance de alrededor de 5.5%.

Si consideramos que los resultados son sumamente similares para todos los clasificadores, esto nos permite concluir que se ha llegado a una performance cercana al tope de lo que puede deducirse a partir de los datos con los que cuentan los clasificadores. Es decir: para continuar mejorando la performance, sería necesario obtener un volumen adicional significativo de datos, o bien un conjunto de features que aporte información nueva que no puede deducirse a partir de los features ya existentes.

En otras palabras, quizás este sea el tope de lo que puede deducirse sin contar con información sobre el autor, contexto o circunstancias temporales y geográficas.

Finalmente en la imagen 4.27 se puede observar un esquema detallado de la solución implementada y el flujo completo desde la carga del corpus hasta el análisis de reputación.

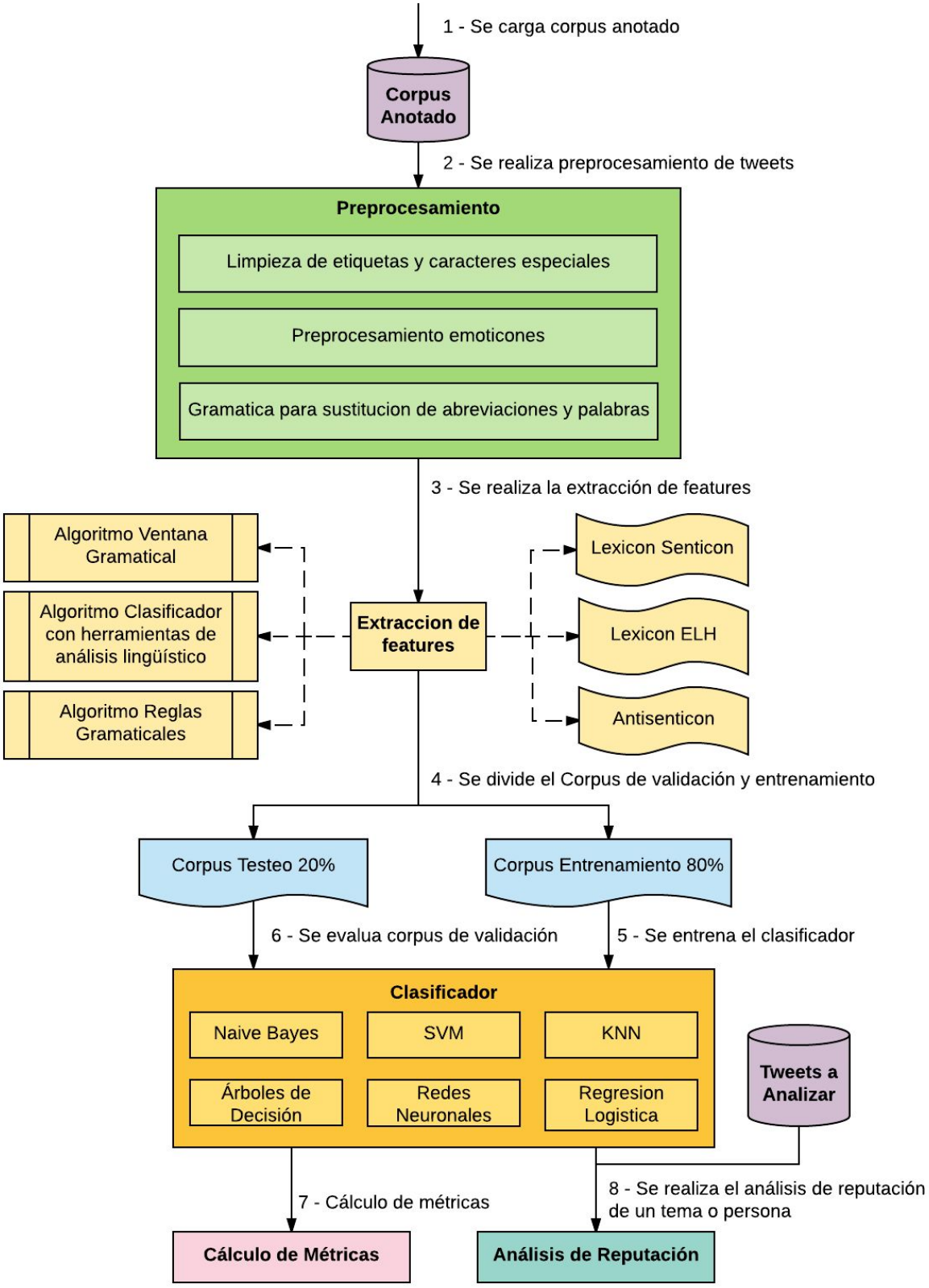


Imagen 4.27: Esquema detallado de la solución implementada

Capítulo 5

Análisis de reputación

En este capítulo se detallan los análisis de reputación realizados, temas y personas seleccionadas, su implementación y resultados.

5.1 Introducción

El objetivo del análisis de reputación es conocer cómo evoluciona la opinión del público general en el tiempo con respecto a un tema o entidad determinado. Para cumplir este objetivo se descargan todos los tweets disponibles en Uruguay correspondiente a múltiples usuarios que opinan sobre el tema o entidad mencionado anteriormente.

Se trata de una aplicación de los clasificadores implementados anteriormente a un conjunto de tweets externos, de tamaño desconocido y no clasificados.

Adicionalmente es importante destacar que tanto en el análisis de reputación como en toda aplicación real de un proceso de aprendizaje automático, resulta difícil o imposible evaluar objetivamente el desempeño de la tarea ya que se trabaja con datos no clasificados. Por lo tanto, en esta sección nos limitaremos a mostrar los resultados obtenidos y brindaremos nuestra opinión, aplicando el sentido común para estudiar si la salida obtenida se asemeja a la salida esperada.

5.2 Implementación

El primer paso es descargar todos los tweets correspondientes a la entidad o tema elegido. Para ello se implementó un método el cual descarga todos los tweets uruguayos que mencionan a un usuario de Twitter o hashtag dado.
Nótese que en la construcción del corpus se descargaron tweets redactados por un usuario (autor), mientras que para el análisis de reputación se descargan tweets que contengan menciones al usuario (sujeto) incluyendo el nombre de la cuenta del sujeto en el cuerpo del tweet.

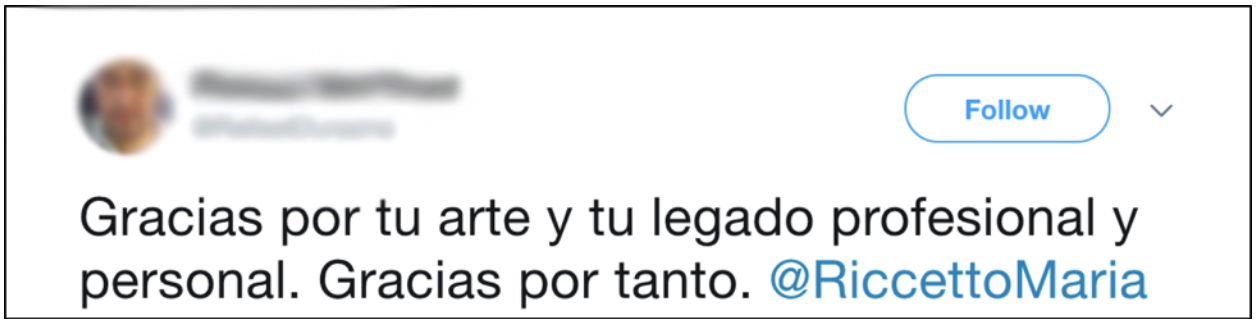


Imagen 5.1- Ejemplo de mención a un usuario

Recordamos que la API de Twitter permite descargar solamente tweets de hasta 7 días de antigüedad, por lo que solamente es posible realizar análisis de reputación de la última semana de un tema o entidad. Adicionalmente, la API solamente retorna un máximo de 100 resultados por cada llamada, por lo que fue necesario implementar un servicio el cual realiza múltiples llamadas incrementalmente hasta conseguir todos los tweets de la última semana. Optamos por incluir respuestas a tweets y no incluir retweets en el estudio del análisis de reputación, aunque ambos parámetros son configurables.

Al final de este documento se encuentra un anexo donde se detalla los aspectos técnicos de los servicios implementados en la creación de esta solución.

Una vez descargados los tweets, se aplica el clasificador ya entrenado sobre los mismos. Se utiliza el clasificador basado en redes neuronales ya que demostró un nivel de performance ligeramente superior al resto, y en la práctica obtuvo muy buenos resultados como se verá más adelante en esta sección. Nótese que si bien se eligió este clasificador para evaluar los resultados del proceso de análisis de reputación, es posible elegir otro clasificador a utilizar mediante configuración.

Una vez concluida la clasificación de los tweets, se calcula y grafica la reputación por día mediante la siguiente fórmula:

$$reputación(tweet, fecha) = \sum_{t \in tweets(fecha)} reputación(t)$$, donde:

$$reputación(t) =$$

1, *si t es una opinión positiva*
− 1, *si t es una opinión negativa*
0 *en caso contrario*

5.3 Ejemplos y resultados

A continuación se presentan algunos resultados obtenidos junto con comentarios sobre los acontecimientos que dan contexto a la reputación de cada tema o sujeto. Nótese que debajo de la línea roja indica reputación negativa, mientras que encima de la misma indica reputación positiva.

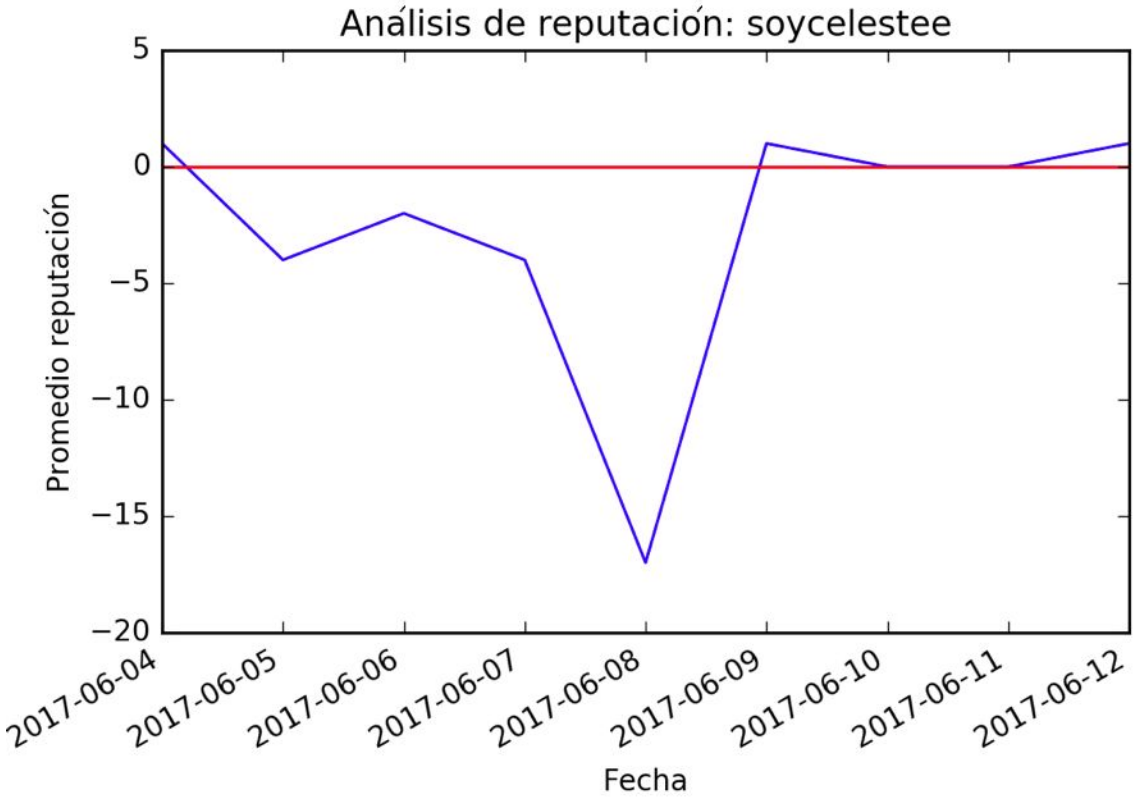


Imagen 5.2- Ejemplo de mención a un usuario

En la imágen 5.2 se puede visualizar el estudio de reputación de la selección uruguaya de fútbol en la semana del 4 al 12 de junio realizado con un total de 193 tweets. Es posible visualizar una marcada caída el 8 de junio, fecha en la cual la selección sub 20 fue eliminada ante Venezuela en la copa mundial.

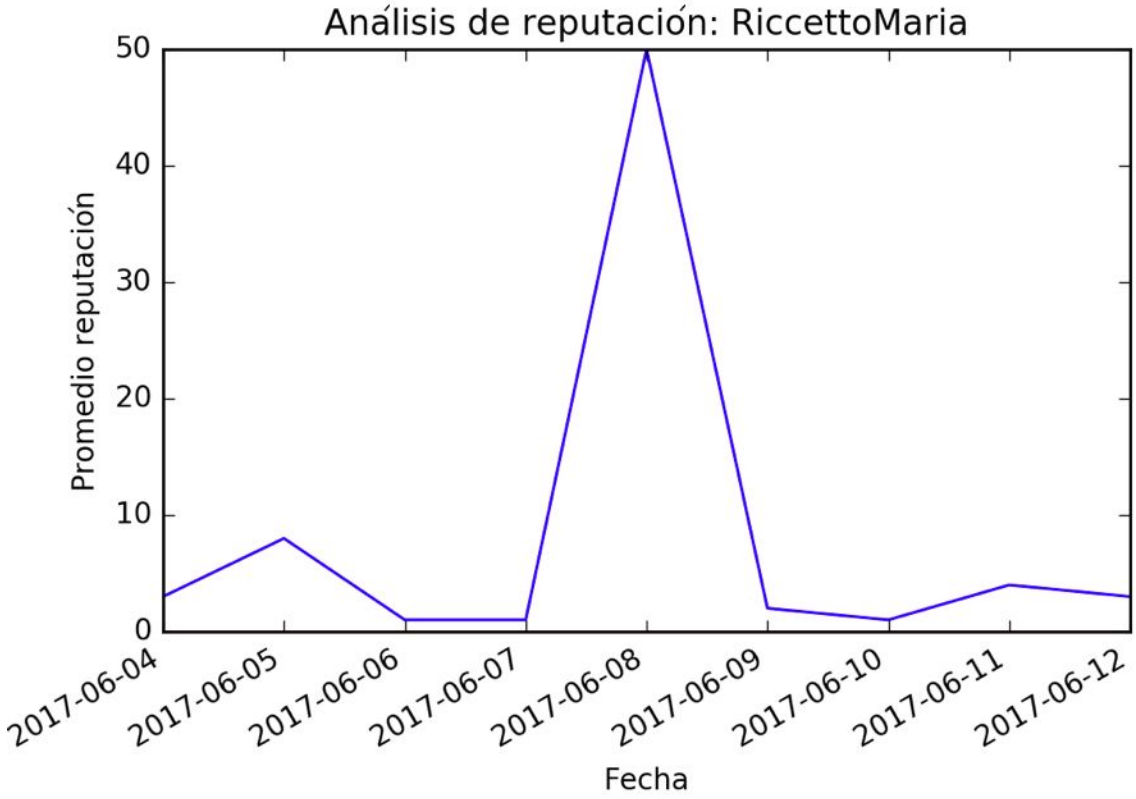


Imagen 5.3- Ejemplo de mención a un usuario

Por otro lado, en la gráfica 5.3 se puede apreciar el estudio de reputación de la bailarina de ballet María Noel Riccetto en la semana del 4 al 12 de junio con un total de 251 tweets. Podemos ver una marcada tendencia positiva con un pico el 8 de junio, donde se anunció en las noticias que había ganado un importante premio.

Es sumamente interesante notar que si bien el premio fue obtenido algunos días antes, el pico no fue registrado hasta que fue difundido mediante las noticias en Twitter.

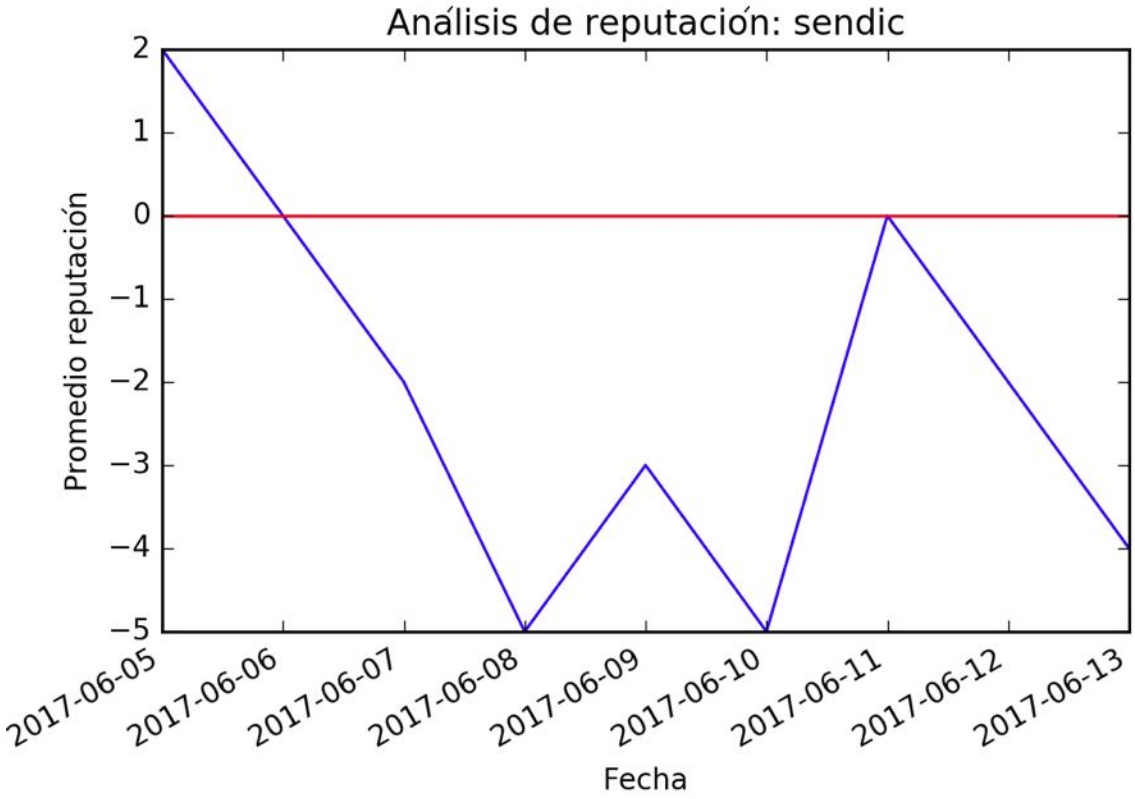


Imagen 5.4 - Ejemplo de mención a un usuario

En la imágen 5.4 se puede visualizar el estudio de reputación del político uruguayo Raúl Sendic en la semana del 5 al 13 de junio con un total de 311 tweets. Es posible visualizar una marcada caída entre el 8 y el 10 de junio, fechas en las cuales circuló una noticia en la que supuestamente utilizó una tarjeta corporativa de un ente estatal para gastos personales.

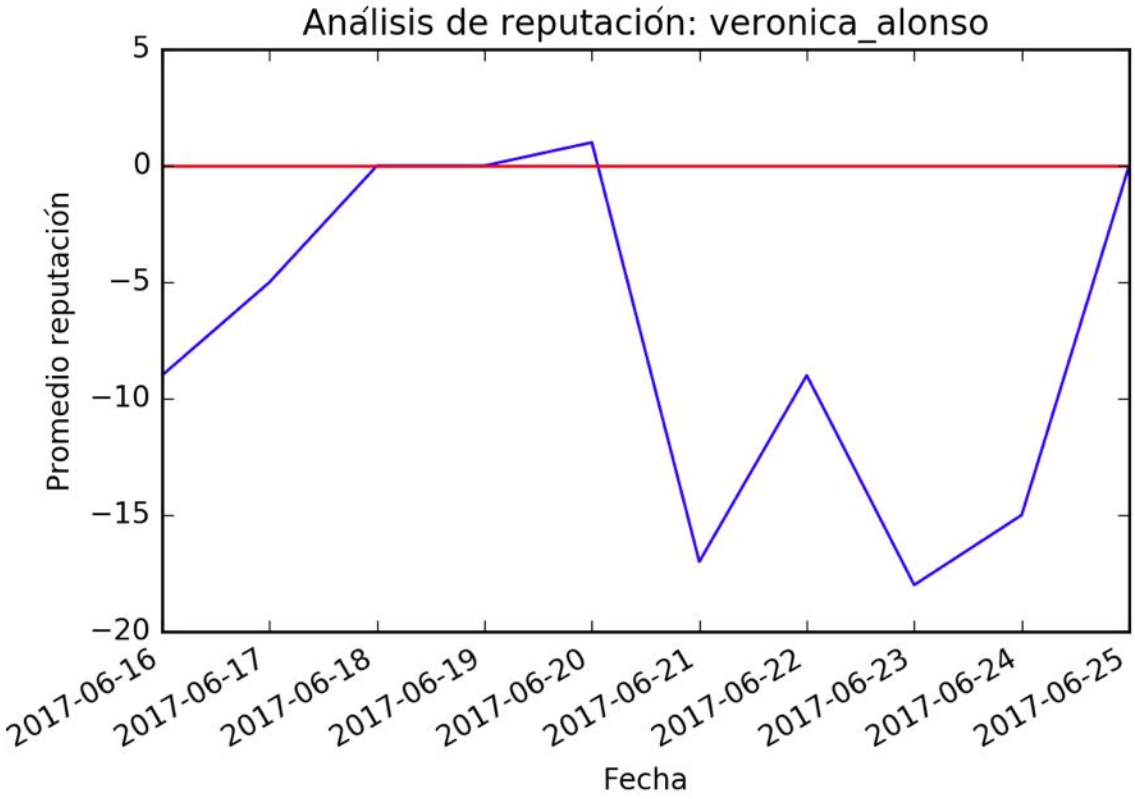


Imagen 5.5- Ejemplo de mención a un usuario

En la imágen 5.5 se puede visualizar el estudio de reputación de la política uruguaya Verónica Alonso en la semana del 16 al 25 de junio con un total de 436 tweets. Es posible visualizar una marcada caída entre el 21 y el 24 de junio, fechas en las cuales circularon una serie de noticias sobre supuestos escándalos políticos.

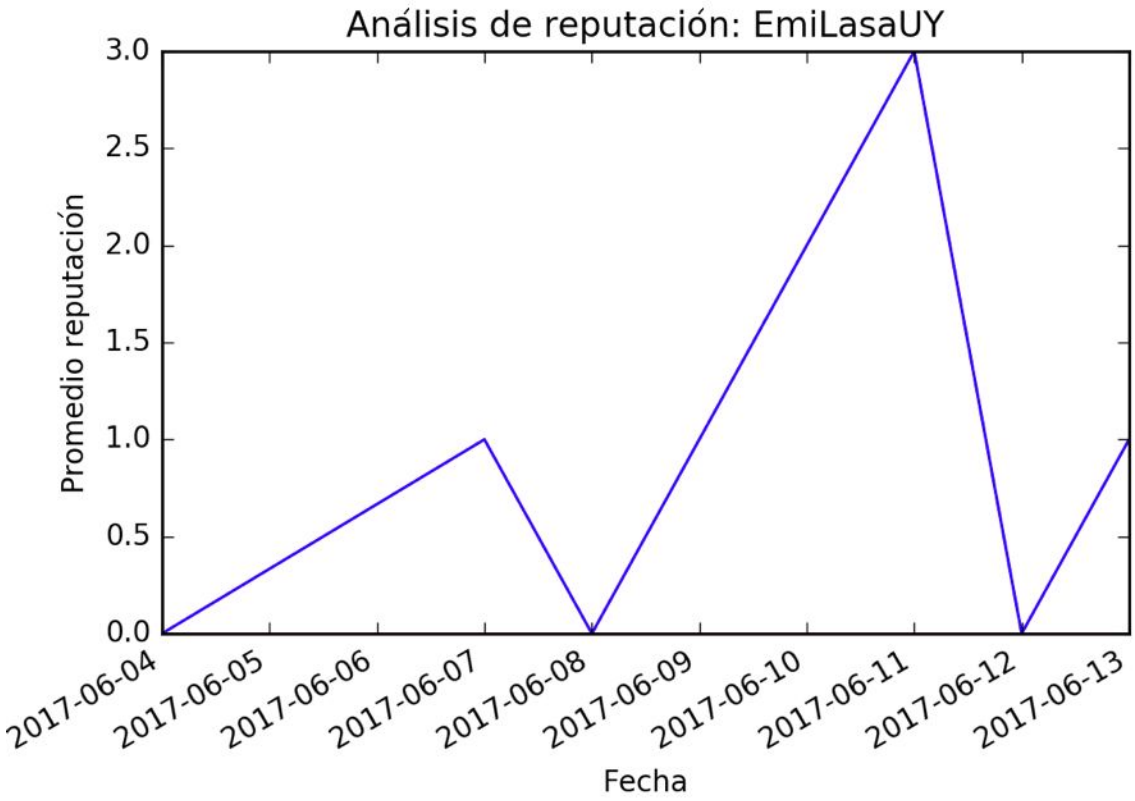


Imagen 5.6- Ejemplo de mención a un usuario

En la imágen 5.6 se puede visualizar el estudio del deportista uruguayo Emiliano Lasa en la semana del 4 al 13 de junio con un total de 207 tweets. Es posible visualizar una marcada tendencia positiva en los días 7 y 11 de Junio, fecha en las cuales clasificó y compitió respectivamente en la final de atletismo internacional disputada en Hengelo, Holanda.

Capítulo 6

Conclusiones finales y trabajo futuro

En este capítulo se presentan las conclusiones finales del trabajo realizado en el marco del proyecto y describe las posibles mejoras a realizar en el futuro.

6.1 Conclusiones finales

Este trabajo se centró en la investigación, análisis y construcción de algoritmos capaces de detectar el sentimiento de textos escritos en la red social Twitter y clasificarlos como opiniones positivas, negativas o no opinión. Siendo el objetivo final generar un estudio de reputación, de un tema o una persona, en un periodo de tiempo dado.

En la búsqueda de diseñar una solución, se estudiaron trabajos relacionados que abordaron el problema de distintas formas. En el contexto del Procesamiento de Lenguaje Natural, en general se pueden distinguir dos grandes enfoques para el análisis de sentimiento: algoritmos basados en reglas y aprendizaje automático. Es claro ver que factores como el medio utilizado, el contexto, el idioma y la región, afectan la calidad y forma de redacción. Twitter en particular, permite enviar textos de corta longitud llamados tweets. Del trabajo de investigación y experimentos realizados se desprende la conclusión de que los tweets generalmente son escritos de manera informal, con abreviaciones y faltas ortográficas, a excepción de algunos usuarios como periodistas o comunicadores que utilizan la red como parte de su trabajo utilizando un lenguaje más formal. Es por estos factores que resultó esencial crear un corpus con tweets de Uruguay que contuviera las jergas y expresiones propios del país.

Así como fue necesario crear un corpus localizado en Uruguay, también fue necesario utilizar un lexicón que contuviera palabras y modismos utilizados en Uruguay. Tomando como base un lexicón elaborado en la Universidad de Sevilla para la SEPLN (Sociedad Española para el Procesamiento de Lenguaje Natural), se agregaron de forma metódica algunas palabras que no se encuentran en el lexicón original.

También se construyó un lexicón para poder discernir cuando un tweet se trata de una opinión o no, ya que los resultados reflejaban que este era uno de los mayores problemas a la hora de clasificar. A este "lexicón de no opiniones" lo llamamos Antisenticon. Cabe destacar que los patrones existentes en el Antisenticon fueron estimados de forma sencilla. En la sección de trabajo a futuro se propone extender y enriquecer estos recursos mediante procedimientos de análisis formal.

Para medir los resultados de los clasificadores utilizamos el 80% de los tweets para entrenar y 20% (526 tweets) para testear. Construimos 3 clasificadores basados en reglas gramaticales y 6 clasificadores de aprendizaje automático.

Para el primer clasificador de reglas gramaticales que se construyó utilizamos el mismo algoritmo que en la línea base, al cual se le incorporó las herramientas de análisis lingüístico y se utilizó el Lexicón Extendido en lugar del Lexicón Original. Este clasificador obtuvo una medida F1 de 60,8%. El segundo clasificador de reglas gramaticales fue basado en árboles de estructura gramatical el cual obtuvo una medida F1 promedio de 60.7%, presentando una leve desmejora al clasificador anterior. Luego construimos un clasificador de ventana gramatical que obtuvo una medida F1 de 64.4%. Se puede observar que se obtuvieron mejoras de alrededor de un 10% en comparación con la línea base, y de un 5% en comparación con el primer clasificador de reglas gramaticales.

También se construyeron 6 clasificadores de aprendizaje automático. Ellos fueron: Naive Bayes, Árbol de Decisión, SVM, KNN, Regresión Logística y uno de Redes Neuronales. Éste último fue el que obtuvo mejores resultados, con un 69% de medida F1 promedio. Éste resultado supera 20% los obtenidos por la línea base. Tomando como referencia la concordancia interna entre 3 clases, el resultado se encuentra cercano al tope de performance estimado en 80%, aunque creemos que aún se puede obtener mejoras mediante la aplicación de alguno de los conceptos que se mencionan en la sección 6.2 (trabajo futuro).

Para realizar el análisis de reputación utilizamos redes neuronales, ya que con este clasificador se obtuvieron resultados ligeramente mejores a los anteriores. Las limitaciones de la API de Twitter resultaron cruciales para seleccionar el periodo de tiempo utilizado en cada caso. Para enriquecer el trabajo se buscó que en el rango de tiempo seleccionado hubiera ocurrido algún evento en el contexto de la persona o tema analizado que pudiera generar opiniones al respecto, como fueron María Noel Riccetto¨ obteniendo un importante premio de ballet; Emiliano Lasa participando en instancias finales de una competición deportiva; la Selección Uruguaya de Fútbol que fue eliminada del mundial sub 20, y los políticos Raúl Sendic y Verónica Alonso que estuvieron implicados en distintos escándalos políticos.

Dado que el conjunto de tweets utilizados para la realización de cada uno de los análisis de reputación no están anotados no contamos con una forma objetiva de poder evaluar el desempeño de la tarea. Sin embargo, aplicando el sentido común y teniendo en cuenta los hechos puntuales en cada caso, del análisis podemos concluir que en todos los casos se obtuvieron los resultados esperados.

Además de los clasificadores construidos y el respectivo análisis de reputación, creemos que se aportaron varios recursos de gran valor a la comunidad gracias a este proyecto. Algunos de ellos son: la creación del corpus localizado, la ampliación del lexicón con modismos uruguayos, la inclusión de emojis al mismo, la construcción de un lexicón para discriminar no opiniones junto con la implementación de la ventana gramatical con la lista de intensificadores y negadores, la limpieza de etiquetas proporcionadas por Twitter, y la sustitución de siglas o abreviaciones para que los textos pudieran ser utilizados para entrenar.

También cabe destacar que la mayoría de los parámetros del programa son altamente configurables, permitiendo adaptar fácilmente cualquier funcionalidad del mismo al enfoque del trabajo, pudiendo ser utilizado en futuros proyectos. Algunos de ellos son: features utilizados, categorías a clasificar, nivel de confianza en los lexicones, el uso o no de herramientas externas como freeling, modos de preprocesamiento.

Concluimos que se cumplieron satisfactoriamente los objetivos planteados por este proyecto: creamos un corpus anotado con tweets locales, entrenamos y experimentamos con distintos clasificadores que mediante un proceso de mejora continua permitieron obtener una herramienta para visualizar la evolución de la reputación de un tema o entidad dado.

Hoy en dia, donde hay cada vez más formas de comunicación y expresión, el análisis de opiniones es una temática muy amplia donde se pueden aplicar muchos enfoques dependiendo del contexto que se quiera analizar. No nos cabe la menor duda de que el análisis de sentimiento y sobre todo el aprendizaje automático estarán presentes en el futuro de varios aspectos de la academia y la vida humana en general.

6.2 Trabajo futuro

En esta sección se describen algunas de las mejoras que se podrían realizar en múltiples áreas del proyecto.

En primer lugar, consideramos se podrían lograr mejores resultados aumentando el tamaño del corpus. En general, proveer más datos a un algoritmo de aprendizaje automático mejora sus resultados aunque sea de forma marginal. Por supuesto, asumiendo que se encuentran correctamente clasificados. De la misma manera, sería positivo perfeccionar y aumentar los recursos léxicos.

Otra mejora potencial vendría de expandir el Antisenticon mediante un proceso sistemático y automatizado. De esta forma se detectan nuevos patrones para identificar si el tweet en cuestión no es una opinión. Una posible alternativa sería experimentar con algoritmos semi-supervisados, tales como bootstrapping.

De forma análoga, aumentar la cantidad de intensificadores y negadores resultaría positivo, así como aumentar su utilidad mediante la incorporación de una solución de lematización para los mismos.

En cuanto a la ingeniería de features de los clasificadores, sería interesante experimentar con nuevas soluciones de generación de features tales como word embeddings, y algoritmos de compresión de features y reducción de dimensionalidad tales como PCA (Principal Component Analysis).

Para mejorar la utilidad de la solución de análisis de reputación, sería deseable incorporar una forma de mitigar la limitante de la API de Twitter que no permite obtener los tweets con más de una semana de antigüedad. Ya sea con herramientas de scraping que accedan directamente a la web de Twitter donde sí se encuentran disponibles estos datos, o mediante alguna otra alternativa. De este modo podría aumentarse los márgenes de tiempo en los que se puede visualizar la progresión de la reputación de un sujeto dado. Por otro lado, también sería deseable incorporar una herramienta que nos permita identificar de quién o qué tema se está hablando en un tweet; unificando temas, personas, hashtags, menciones, etc.

Por último, se podría expandir la accesibilidad del proyecto para cualquier usuario mediante la implementación de una interfaz web que permita obtener el análisis de reputación a demanda, con clasificadores pre-entrenados.

Bibliografía

[1] Antonio Moreno Sandoval, "*Análisis de opinión y contenido en los medios sociales*", Instituto de Ingeniería del Conocimiento, Madrid, España.

[2] Página oficial de Twitter, URL: *https://support.twitter.com/articles/76621*, Accedido Noviembre de 2016.

[3] Liu, B, "*Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data*", Springer, Alemania, 2007

[4] Strachan, Donald, "*Twitter: How To Set Up Your Account*", The Daily Telegraph, 2009.

[5] Pagina oficial de Twitter, URL: *https://support.twitter.com/articles/247830*, Accedido Junio de 2017.

[6] Pagina oficial de Twitter, URL: *https://support.twitter.com/articles/230754*, Accedido Junio de 2017.

[7] Press, Europa , "*Twitter libera 872 emojis en código abierto*", URL: http://www.europapress.es/portaltic/socialmedia/noticia-twitter-libera-872-emojis-codigo-abierto-20141107155031.html , 2014

[8] Alec Go, Richa Bhayani, Lei Huang, "*Twitter Sentiment Classification using Distant Supervision*" URL: *http://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf, Stanford University 2009*

[9] Apoorv Agarwal Boyi Xie Ilia Vovsha Owen Rambow Rebecca Passonneau, "*Sentiment Analysis of Twitter Data*" URL: *http://dl.acm.org/citation.cfm?id=2021114* , Department of Computer Science Columbia University New York, 2011

[10] Eugenio Martínez-Cámara, Maria Teresa Martín-Valdivia, José Manuel Perea-Ortega, L. Alfonso Ureña López, "*Técnicas de clasificación de opiniones aplicadas a un corpus en español*", URL: *https://www.researchgate.net/publication/230736870_Tecnicas_de_clasificacion_de_opiniones_aplicadas_a_un_corpus_en_espanol* , 2011

[11] Fermín Cruz Mata, José Antonio Troyano Jiménez, Fernando Enriquez, F. Javier Ortega, "*Clasificación de documentos basada en la opinión: Experimentos con un corpus de críticas de cine en español*", URL: *https://www.researchgate.net/publication/28238383_Clasificacion_de_documentos_basada_en_la_opinion_Experimentos_con_un_corpus_de_criticas_de_cine_en_espanol* , 2008

[12] Guillermo Dufort y Álvarez, Fabián Kremer, Gabriel Mordecki, "*Determinación de la orientación semántica de las opiniones transmitidas en textos de prensa*", Proyecto de grado Ingeniería en Computación, Universidad de la República, Montevideo, Uruguay, 2016

[13] Duyu Tang, Furu Wei, Bing Qin, Ting Liu, Ming Zhou, "*Coooolll: A Deep Learning System for Twitter Sentiment Classification*", URL: *http://www.aclweb.org/anthology/S14-2033* , 2014

[14] Oscar S. Siordia, Mario Graff, "*Sentiment Analysis for Twitter: TASS 2015*" URL: *http://ceur-ws.org/Vol-1397/centroGEO_Infotec.pdf , 2015*

[15] Miguel Ángel García Cumbreras, Julio Villena Román, Eugenio Martínez Cámara,Manuel Carlos Díaz Galiano, M. Teresa Martín Valdivia, L. Alfonso Ureña López, "*Overview of TASS 2016*", URL: *http://ceur-ws.org/Vol-1702/tass2016_proceedings_v24.pdf* , Universidad de Jaén 2016

[16] Jonathon Read, "*Using emoticons to reduce dependency in machine learning techniques for sentiment classification*", URL: *http://dl.acm.org/citation.cfm?id=1628969* , University of Sussex, United Kingdom 2005

[17] Flach, Peter, "*The Art and Science of Algorithms that Make Sense of Data*", Cambridge University Press, 2012.

[18] Rokach, Lior, Maimon, O, "*Data mining with decision trees: theory and applications*", World Scientific Pub Co Inc. ISBN 978-9812771711, 2008

[19] Gustavo A. Betancourt. "*Las Máquinas de Soporte Vectorial (SVMs)* ", URL: *http://revistas.utp.edu.co/index.php/revistaciencia/article/view/6895, 2005*

[20] Oliver Sutton, "*Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction*", URL: *http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Talk.pdf,* 2012

[21] Daniel Jurafsky & James H. Martin, "*Speech and Language Processing*", URL: *https://web.stanford.edu/~jurafsky/slp3/7.pdf,* 2016

[22] S. Mittal, ACM Computing Surveys, "*A Survey Of Techniques for Approximate Computing*", URL: *https://www.academia.edu/20201007/A_Survey_Of_Techniques_for_Approximate_Computing*, 2016

[23] Real academia española, URL: *http://www.rae.es/*

[24] Fermín L. Cruz, José A. Troyano, Beatriz Pontes, F. Javier Ortega, "*Un lexicón multilingüe de polaridades semánticas a nivel de lemas*", URL: *http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5041*

[25] Apoorv Agarwal Boyi Xie Ilia Vovsha Owen Rambow Rebecca Passonneau Department of Computer Science Columbia University New York

[26] "TAS 2014", URL: *http://users.dsic.upv.es/~lhurtado/TMSM/* , 2014

[27] Dorian Pyle, "*Data Preparation for Data Mining",* Morgan Kaufmann Publishers, 1999

[28] Diccionario Panhispánico de Dudas, Real Academia Española, "*Emoticono*", Accedido Mayo 2017 ,URL: *http://lema.rae.es/dpd/?key=emoticono*

[29] Unicode, "*Emoji and Dingbats*", Accedido Mayo 2017, URL: *http://www.unicode.org/faq/emoji_dingbats.html*

[30] Tom M. Mitchell, *" Machine Learning"*, WCB/McGraw-Hill, 1997

[31] *Pyfreeling*, Accedido Setiembre 2016. URL: *https://github.com/malev/pyfreeling*

[32] *Pattern.es,* Accedido Agosto 2016. URL: *http://www.clips.ua.ac.be/pages/pattern-es*

[33] Guillermo Dufort y Álvarez, Fabián Kremer, Gabriel Mordecki, "*Determinación de la orientación semántica de las opiniones transmitidas en textos de prensa*", Proyecto de grado Ingeniería en Computación, Universidad de la República, Montevideo, Uruguay, 2016

[34] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau, "*Sentiment analysis of Twitter data*", URL: *http://dl.acm.org/citation.cfm?id=2021114,* 2011

Anexo 1: Aspectos técnicos de los servicios implementados

En esta sección se describen los aspectos técnicos de la arquitectura de los servicios implementados con el objetivo de poblar el corpus y realizar análisis de reputación. Estos servicios suministran información a ambas aplicaciones mencionadas en la sección de construcción del corpus (web y Android), y a los notebook IPython donde se encuentra la implementación de los clasificadores y el análisis de reputación.

Los tweets obtenidos para la construcción del corpus son almacenados en una base de datos, mientras que los tweets obtenidos para el análisis de reputación se obtienen a demanda para un tema o entidad dada, y son desechados una vez que el análisis de reputación ha finalizado.

Arquitectura de la solución y servicios

Los servicios implementados extraen tweets directamente desde la API de Twitter. Se trata de servicios REST que devuelven datos en formato JSON. Fueron implementados en Python sobre un servidor web Django con hosting en Heroku. Los tweets descargados se almacenan en una base de datos PostgreSQL.

Los servicios cuyo propósito es poblar el corpus permiten descargar y almacenar los últimos 200 tweets que cuentan con un cierto hashtag, que fueron escritos por un cierto usuario, o que mencionan a un cierto usuario.

Para cumplir con el objetivo de trabajar exclusivamente con tweets locales, todos estos servicios cuentan con un filtro por geolocalización que obtiene exclusivamente tweets escritos en el Uruguay. Adicionalmente, los servicios que persisten tweets en la base de datos filtran los retweets o menciones de usuarios para evitar obtener resultados duplicados.

Por otro lado se cuenta con servicios que obtienen tweets para análisis de reputación, pero estos son descargados y retornados directamente a demanda, sin persistir en la base de datos. Es importante notar que estos servicios no filtran retweets o respuestas ya que son relevantes para la medición y el estudio de reputación.

En el diagrama de la página siguiente se puede visualizar una vista física de la arquitectura implementada.

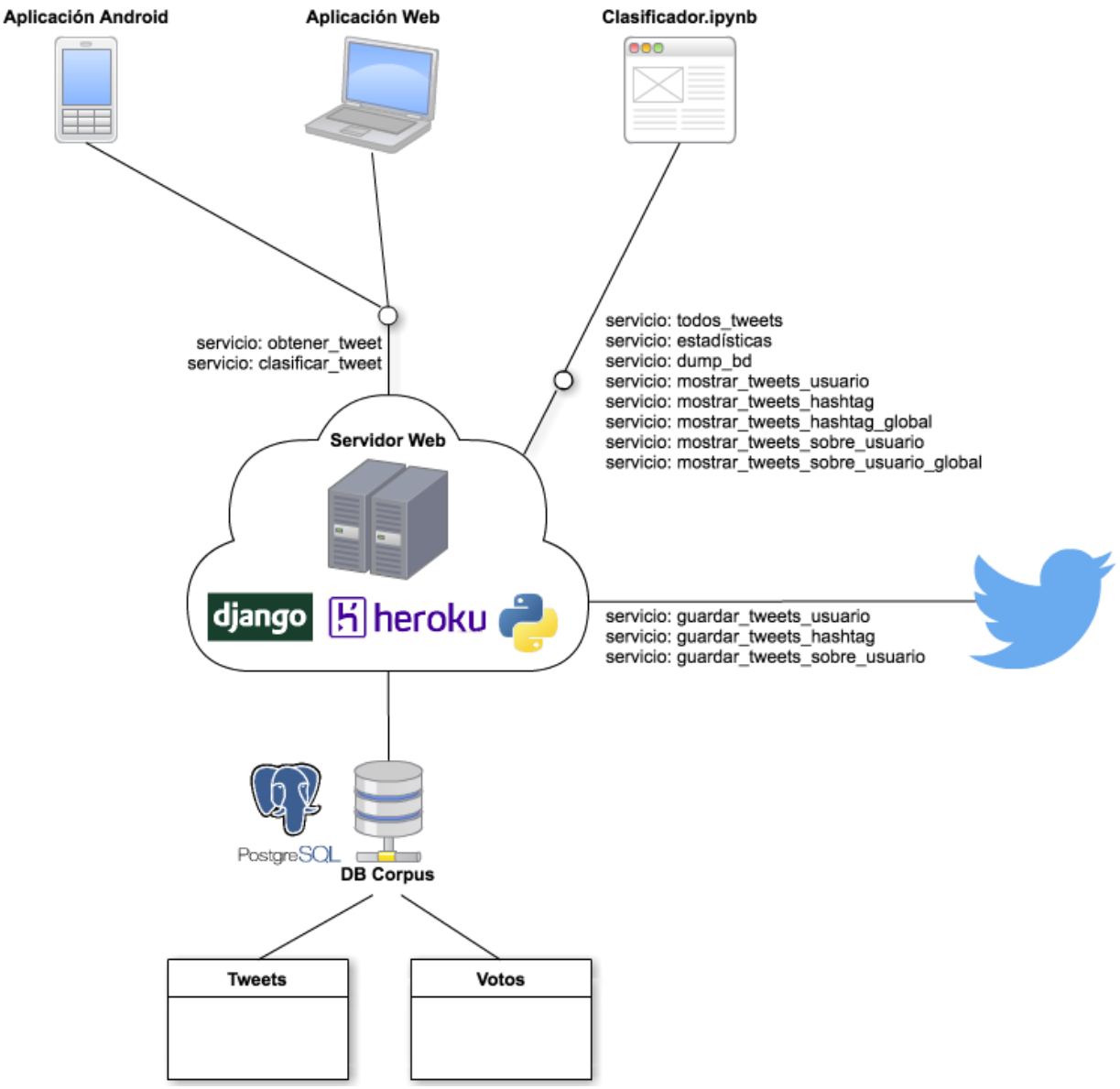


Diagrama de la arquitectura de servicios implementada

Lista de servicios implementados

A continuación se detallan los servicios implementados con el objetivo de poblar el corpus y obtener los datos sobre los cuales aprenden los clasificadores y se realiza el análisis de reputación.

- obtener_tweet: obtiene un tweet aleatorio para que sea clasificado por un usuario. Se utiliza tanto por la aplicación web como por la aplicación de Android.
- clasificar_tweet: envía la clasificación emitida por el usuario para almacenarla en la base de datos. se utiliza tanto por la aplicación web como por la aplicación de Android.
- todos_tweets: se utiliza en el notebook python para obtener los datos de todos los tweets del corpus junto con sus votos correspondientes
- estadisticas: presenta datos estadísticos (cantidad total de tweets, cantidad total de votos, etc)
- dump_bd: hace un dump completo de la base de datos en formato JSON para poder respaldar su contenido localmente.
- guardar_tweets_usuario: consulta la API de twitter, descarga y persiste los últimos 200 tweets de un usuario dado. Se utiliza para poblar el corpus. Filtra respuestas y retweets.
- mostrar_tweets_usuario: consulta la API de twitter y retorna todos los tweets de la última semana de un usuario dado. Se utiliza para realizar análisis de reputación.
- guardar_tweets_hashtag: consulta la API de twitter, descarga y persiste los últimos 200 tweets que contienen un hashtag dado. Se utiliza para poblar el corpus. Filtra respuestas y retweets.
- mostrar_tweets_hashtag: consulta la API de twitter y retorna todos los tweets de la última semana que contienen un hashtag dado. Se utiliza para realizar análisis de reputación.
- mostrar_tweets_hashtag_global: idem al servicio anterior, pero sin filtro por geolocalización.
- guardar_tweets_sobre_usuario: consulta la API de twitter, descarga y persiste los últimos 200 tweets que contienen menciones a un usuario dado. Se utiliza para poblar el corpus. Filtra respuestas y retweets.
- mostrar_tweets_sobre_usuario: consulta la API de twitter y retorna todos los tweets de la última semana que contienen menciones a un usuario dado. Se utiliza para realizar análisis de reputación.
- mostrar_tweets_sobre_usuario_global: idem al servicio anterior, pero sin filtro por geolocalización.

Anexo 2: Instalación de Freeling y pyfreeling

En el siguiente anexo se presentan las instrucciones seguidas por el grupo para instalar localmente Freeling y conectarlo al código Python mediante la interfaz expuesta por pyfreeling.

Instalación de Freeling

Es importante destacar que Freeling es una herramienta independiente, y no está diseñada para integrarse nativamente con Python. Por ese motivo, el primer paso es instalar Freeling de manera independiente.

Instalación en GNU/Linux

La instalación sigue el procedimiento usual para cualquier instalación autoconfigurable GNU (configure, make y make install).

1. Instalar los pre-requisitos:
   ```
   $ sudo apt-get install build-essential automake autoconf libtool
   $ sudo apt-get install libboost-regex-dev libicu-dev zlib1g-dev
   $ sudo apt-get install libboost-system-dev
   $ sudo apt-get install libboost-program-options-dev
   ```
2. Descargar el archivo .tar.gz correspondiente al código fuente. En nuestro caso, utilizamos la versión 4.0. Luego ejecutar:
   ```
   $ tar xzvf freeling-4.0.tar.gz
   $ cd freeling-4.0
   $ autoreconf --install
   $ ./configure
   $ Make
   $ sudo make install
   ```
3. La librería de Freeling queda completamente contenida dentro del archivo libfreeling.so dentro de `/usr/local/lib` (por defecto).

Instalación en MacOS

La instalación en Mac resulta más sencilla al utilizar el gestor de paquetes Homebrew. El único pre-requisito es contar con Ruby instalado.

1. Instalación de Homebrew. Copiar el siguiente texto en la consola:
   `$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
2. Instalar freeling mediante brew:
   `$ brew install freeling`

Instalación y uso de pyfreeling

La instalación de la librería pyfreeling se realiza mediante el gestor de paquetes pip:

```
$ pip install pyfreeling
```

Alternativamente, puede realizarse directamente desde Anaconda mediante el siguiente comando:

```
$ conda install pyfreeling -c malev
```

Una vez instalado, se utiliza desde el código de la siguiente manera, donde la ruta de la configuración dependerá de cada instalación:

```python
from pyfreeling import Analyzer

RUTA_FREELING_CFG = '/usr/local/Cellar/freeling/4.0_3/share/freeling/config/es.cfg'

def obtener_analisis_freeling(texto):
    analyzer = Analyzer(config=RUTA_FREELING_CFG, lang='es')
    return analyzer.run(texto.strip().encode('utf-8'), 'flush', outlv='shallow --noloc')
```

Anexo 3: Ajuste de hiperparámetros

En el siguiente anexo se presentan los ajustes de hiperparámetros realizados para cada uno de los clasificadores de aprendizaje automático.

Árboles de Decisión

Se experimentó con las clases GridSearchCV y RandomizedSearchCV para realizar ajuste de hiperparámetros mediante búsqueda de grilla y búsqueda aleatoria respectivamente.

Utilizando un modelo de cross-validation dentro del corpus de entrenamiento, se experimentó con las siguientes combinaciones de hiperparámetros:

- Criterio: entropía o impureza de Gini
- Mínima cantidad de muestras (datos) requeridos para separar un nodo: se experimentó con valores entre 20 y 500
- Máxima profundidad del árbol: se experimentó con valores entre 10 y 70
- Mínima cantidad de muestras (datos) requeridos para construir una hoja del árbol: se experimentó con valores entre 1 y 50
- Máxima cantidad de hojas: se experimentó con valores desde 50 en adelante

Los hiperparámetros encontrados que maximizan la performance del clasificador, son:

- Criterio: entropía
- Mínima cantidad de muestras (datos) requeridos para separar un nodo: 50
- Máxima profundidad del árbol: 10
- Mínima cantidad de muestras (datos) requeridos para construir una hoja del árbol: 1
- Máxima cantidad de hojas: sin límite

SVM

Utilizando un modelo de cross-validation dentro del corpus de entrenamiento, se experimentó con las siguientes combinaciones de hiperparámetros:

- El tipo de kernel que puede utilizar el algoritmo: lineal, polinómico, RBF (radial basis function) o sigmoide
- Gamma, el coeficiente de kernel para utilizar. Se experimentó con los siguientes valores: automático, 0.01, 0.001 o 0.0001
- El parámetro C de penalización del término del error. Se probó con valore desde 0.01, a 100.
- Peso de clase: balanceado o ninguno

Los hiperparámetros encontrados que maximizan la performance del clasificador, son:

- Tipo de kernel: RBF
- Gamma: 0.01
- Mejor C: 50.0
- Peso de clase: ninguno

KNN

Utilizando un modelo de cross-validation dentro del corpus de entrenamiento, se experimentó con las siguientes combinaciones de hiperparámetros:

- Algoritmo: ball tree, kd-tree, fuerza bruta y automático.
- Peso: uniforme o distancia
- Tamaño de hoja para los algoritmos de ball tree y kd-tree: se experimentó con valores entre 1 y 100
- Parámetro de potencia para el algoritmo de Minkowski: se experimentó con valores 1 y 2

Los hiperparámetros encontrados que maximizan la performance del clasificador, son:

- Algoritmo: fuerza bruta
- Peso: uniforme
- Tamaño de hoja: 1
- Parámetro de potencia para el algoritmo de Minkowski: 1

Regresión logística

Utilizando un modelo de cross-validation dentro del corpus de entrenamiento, se experimentó con las siguientes combinaciones de hiperparámetros:

- Inverso de la fuerza de regularización C: se experimentó con valores entre 0.001 y 100
- Algoritmo de optimización numérica: se experimentó con Newton-CG (Conjugate Gradient), LBFGS y SAG (Stochastic Average Gradient)
- Multi clase: entrenar un problema para cada etiqueta o no.

Los hiperparámetros encontrados que maximizan la performance del clasificador, son:

- Inverso de la fuerza de regularización C: 0.03
- Algoritmo: Newton-CG
- Multi clase: si (no entrenar un problema por etiqueta)

Redes Neuronales

Utilizando un modelo de cross-validation dentro del corpus de entrenamiento, se experimentó con las siguientes combinaciones de hiperparámetros:

- Estructura de capas: se experimentó con entre 1 y 5 capas de entre 30 y 60 nodos cada una
- Alfa (término de regularización): se experimentó con valores entre 0.0001 y 0.1
- Función de activación: se experimentó con la función identidad, tangente hiperbólica, ReLU (Rectified Linear Unit) y logística.
- Algoritmo de optimización numérica: se experimentó con LBFGS, SGD y Adam (Adaptative Moment Estimation)
- Algoritmo de cambio de tasa de aprendizaje. Se experimentó con las siguientes opciones: constante, escalado inverso (Inverse Scaling) y adaptativo.

Los hiperparámetros encontrados que maximizan la performance del clasificador, son:

- Estructura de capas: 1 capa de 50 nodos (cabe destacar que también se obtuvieron resultados equivalentes con 3 capas de 30 nodos)
- Alfa (término de regularización): 0.0001
- Función de activación: ReLU
- Algoritmo de optimización numérica: LBFGS
- Algoritmo de cambio de tasa de aprendizaje: constante

Anexo 4: Estructura del código entregado

A continuación se detalla la estructura del código entregado. Trabajar en un único IPython notebook resulta sumamente costoso en cuanto a performance, complejidad del código y tiempo en gestión de la configuración. Por este motivo, dividimos el código en múltiples notebooks IPython, los cuales se ejecutan en un orden definido. En la primera línea de cada notebook se ejecutan todos los notebooks anteriores al mismo.

Los siguientes notebooks contienen las principales secciones del proyecto y se ejecutan en el orden especificado.

- `NotebookUtils.py`: contiene el código utilizado para importar y ejecutar un notebook desde otro. Se utilizó para implementar la arquitectura modular de notebooks presentada en esta sección.
- `Modulo_1_Carga_Datos.ipynb`: define todas las constantes y parámetros, carga datos desde archivo tales como los lexicones, y descarga todos los tweets junto con sus votos.
- `Modulo_2_Preprocesamiento.ipynb`: implementa y ejecuta todas las operaciones relacionadas al preprocesamiento de los tweets descargados.
- `Modulo_3_Linea_Base.ipynb`: calcula métricas tales como la concordancia entre los votantes, implementa y ejecuta la línea base.
- `Modulo_4_Gramatica.ipynb`: ejecuta freeling sobre todo el corpus, implementa los métodos relativos a la lematización, y ambos clasificadores basados en reglas gramaticales.
- `Modulo_5_Extraccion_Features.ipynb`: implementa los métodos que definen los features, los ejecuta y almacena los resultados en formato CSV.
- A continuación se detallan los módulos que implementan los distintos clasificadores. Cada uno puede ser ejecutado de forma independiente.
    - `Clasificador_ANN.ipynb`: implementa el clasificador basado en redes neuronales.
    - `Clasificador_Decision_Tree.ipynb`: implementa el clasificador basado en árboles de decisión
    - `Clasificador_KNN.ipynb`: implementa el clasificador basado en k-vecinos más próximos.
    - `Clasificador_Logistic_Regression.ipynb`: implementa el clasificador basado en regresión logística.
    - `Clasificador_Naive_Bayes.ipynb`: implementa el clasificador basado en naïve bayes.
    - `Clasificador_SVM.ipynb`: implementa el clasificador basado en máquinas de vectores de soporte.
- `Modulo_6_Analisis_Reputacion.ipynb`: dentro de ete módulo se elige uno de los clasificadores anteriores y se realiza el estudio de análisis de reputación para temas variados.

Adicionalmente, se entregan los siguientes archivos a destacar

- Dumps BD: contiene un dump de la base de datos, el cual contiene todos los tweets y todos los votos.
- EjecucionesFreeling: aquí se almacenan todas las ejecuciones de freeling de los tweets post-procesdos. Cada archivo es el ID de un tweet, y la salida de freeling se almacena en formato XML.
- EjecucionesFreelingSinPreprocesamiento: de forma análoga, aquí se almacenan todas las ejecuciones de freeling sobre los tweets en crudo (sin preprocesamiento).
- AnalisisReputacion: aquí se almacenan los resultados del análisis de reputación.
- Lexicón: aquí se almacenan los diccionarios y lexicones utilizados a lo largo del proyecto.
- Métricas: aquí se almacenan los resultados de las ejecuciones de un clasificador luego de aplicar el método print_metricas.