CIMAT

Centro de Investigación en Matemáticas, A.C.

DETECCIÓN DE IRONÍA EN TEXTOS GENERADOS POR USUARIOS

T E S I S

Que para obtener el grado de

Maestro en Cómputo Estadístico

Presenta

Lizeth Adriana García Belmonte

Directores de Tesis:

Dra. Daniela Alejandra Moctezuma Ochoa
Dr. Víctor Hugo Muñíz Sánchez

Monterrey, N.L., 16 de julio de 2019

CIMAT

Centro de Investigación en Matemáticas, A.C.

DETECCIÓN DE IRONÍA EN TEXTOS GENERADOS POR USUARIOS

T E S I S

Que para obtener el grado de

Maestro en Cómputo Estadístico

Presenta

Lizeth Adriana García Belmonte

Directores de Tesis:

Dra. Daniela Alejandra Moctezuma Ochoa

Dr. Víctor Hugo Muñíz Sánchez

Autorización de la versión final

Monterrey, N.L., 16 de julio de 2019

Índice general

1 Introducción **3**

2 Preliminares **5**
2.1 Normalización 5
2.1.1 Normalización del formato de palabras 5
2.1.2 Segmentación 7
2.2 Métodos de representación de documentos 8
2.2.1 Enfoque léxico 8
2.2.2 Enfoque semántico 10

3 Detección de Ironía **11**
3.1 Definición ironía 11
3.2 Antecedentes 12
3.3 Definición del problema 14

4 Los datos **17**
4.1 Corpus en inglés 17
4.2 Corpus en español 18
4.3 Preprocesamiento 19

5 Metodología **23**
5.1 Baseline 23
5.2 Evaluación del modelo 24
5.3 Modelo de detección de ironía 26
5.3.1 Enfoque léxico 26
5.3.2 Enfoque semántico 31
5.4 Clasificadores 36

6 Resultados y Análisis **43**
6.1 Inglés 43
6.2 Español - Corpus Balanceado 46
6.3 Español - Corpus No Balanceado 49
6.4 Resumen de resultados 51

6.5 Desempeño de los clasificadores 52

7 Conclusiones y trabajos Futuros **55**

7.1 Conclusiones 55

7.2 Trabajos futuros 57

Bibliografía **59**

Agradecimientos

En primer lugar, quiero agradecer a mi familia, por su apoyo y confianza en todos mis proyectos.

A mis profesores y compañeros de generación, de quienes aprendí tanto y gracias a su apoyo pude llevar a cabo la maestría con éxito. A la Dra. Graciela González Farías por sus atenciones y apoyo durante estos dos años. Y en especial a mis asesores, el Dr. Víctor Muñíz Sanchez y la Dra. Daniela Alejandra Moctezuma Ochoa, sin ellos este trabajo no sería posible.

Agradezco al Centro de Investigación en Matemáticas A.C. (CIMAT) por haberme dado la oportunidad de estar en el programa de maestría y por la formación académica que me ofreció. Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca recibida para realizar mis estudios de Maestría.

Un agradecimiento especial para el Dr. Ivan Vladimir Meza Ruiz y a Max Sanchez Hernandez por su permitirnos utilizar su base de datos en la realización de esta tesis, la colaboración científica siempre es beneficiosa.

1 Introducción

El análisis de textos, a través del Procesamiento del Lenguaje Natural, se ha vuelto un elemento de mucha utilidad en distintos campos y para estudiar diferentes fenómenos, las aplicaciones abarcan una gran variedad de ejemplos como el análisis de expedientes clínicos, currículos, detección de personas, entidades o lugares en textos y la minería de opinión.

En aplicaciones como el análisis de sentimiento y la minería de opinión, las fuentes de información más comunes son aquellas generadas por usuarios, como publicaciones en redes sociales, opiniones de productos y los foros de preguntas. La ironía es un recurso comúnmente usado en textos generados por usuarios para expresar una declaración negativa de una manera aparentemente positiva, o viceversa. Por lo tanto, el incorporar la detección de ironía afecta directa o indirectamente el desempeño de los modelos en tareas tan diversas como el análisis de sentimientos, la extracción de opiniones, atención a clientes, segmentación de clientes, análisis de mercados financieros, el comercio electrónico, la gestión de foros, el marketing en línea y evaluación de productos.

La ironía es una figura literaria que encontramos en la cotidianidad en una variedad de formas y con diversos grados de evidencia. En un discurso hablado o en el contexto de una conversación, generalmente podemos detectar pistas que permiten la percepción de la ironía, como pueden ser la gesticulación, entonación o la duración de una pausa. Sin embargo, en los textos, el conjunto de estrategias lingüísticas que se usan para expresar la ironía no son tan evidentes, lo que hace a la detección de ironía, como tarea de aprendizaje automático, un problema interesante, pues los enfoques clásicos de aprendizaje automático no funcionan tan bien como en otras tareas de clasificación de textos.

El objetivo de este trabajo es explorar los modelos empleados en la tarea de detección de ironía en el estado del arte desde una perspectiva de aprendizaje supervisado. Adicionalmente, se busca comparar el desempeño de los modelos de acuerdo a qué tipo de características utilizan para abordar el problema, si es léxico o semántico. Para evaluar el desempeño de dichos modelos nos enfocaremos en textos generados por usuarios de la red social Twitter. Utilizamos textos en inglés como referencia, debido a la gran variedad de trabajos que se han desarrollado para este idioma, y consideramos textos en español para estudiar si estos modelos son aplicables en el caso particular del español.

Este documento se estructura de la siguiente manera: el Capítulo 2 se establecen algunas definiciones y conceptos útiles para el desarrollo de esta tesis, en el Capítulo 3 se define de manera formal lo que es la ironía y se presenta el marco teórico del problema de detección de ironía. En el Capítulo 4 se describen los conjuntos de datos (Corpora) utilizado para este trabajo, tanto para el caso de los textos en inglés como en español. En el Capítulo 5 se exponen los modelos utilizados en este trabajo, así como las métricas de evaluación de los mismos. Los resultados se presentan en el Capítulo 6. Por último, en el Capítulo 7 se dan las Conclusiones y los posibles trabajos futuros.

Preliminares

2

El Procesamiento del Lenguaje Natural (PLN) es un campo de las ciencias de la computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano. Algunas aplicaciones que tiene el PLN es el reconocimiento de voz, la traducción entre idiomas, la comprensión de oraciones completas, la corrección de ortografía, entre otros.

En este capítulo se presentan algunos conceptos y técnicas útiles para el procesamiento de texto desde la perspectiva del PLN. El uso de estas herramientas permite extraer características léxicas, sintácticas y semánticas.

2.1 Normalización

Antes de comenzar a trabajar los textos con modelos de lenguaje natural, es conveniente transformar el texto a una forma estándar. La normalización, o preprocesamiento, del texto consiste en aplicar una serie de transformaciones en los datos originales para dejar todos los textos en el mismo formato. La normalización de los textos dependerá de la tarea que se esté abordando, entre las transformaciones más usuales están el convertir todos los textos a minúsculas o mayúsculas, eliminar los signos de puntuación o convertir los números a sus equivalentes de palabras.

Para este trabajo abordaremos la normalización en dos pasos: la normalización del formato de las palabras y la segmentación del texto. [1]. Los textos que analizamos son generados por usuarios en Twitter, de manera que en estos se tiene gran variabilidad de recursos como emojis, mayúsculas o signos de puntuación para denotar algún sentimiento o idea; los procesos de normalización que aplicaremos buscan no perder esta información.

2.1.1 Normalización del formato de palabras

En todos los idiomas es posible encontrar, en mayor o menor medida, palabras que se derivan de otras. En gramática, se conoce como flexión a la alteración que experimentan las palabras para expresar diferentes categorías como tiempo, número ó género. A través de las flexiones es

[1] Para más información sobre otras tareas de normalización de textos puede consultar Jurafsky, 2000.

posible expresar una o más categorías gramaticales (con un prefijo, sufijo o infijo, o un cambio de vocal).

La derivación (***Stemming***) y la lematización (***Lemmatization***) son técnicas de normalización de texto que se utilizan para reducir las formas de flexión de cada palabra a una base o raíz común, con el fin de homologar términos.

El proceso de *Stemming* es con el que principalmente eliminamos los sufijos del final de la palabra para obtener su raíz. Esta es la manera más básica para hacer análisis morfológico de las palabras. Por su parte, la lematización consiste en, dada una forma flexionada (plural, femenino, conjugada, etc), encontrar el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. En la Tabla 2.1 se aprecia la diferencia entre la lematización y el stemming para palabras que son flexiones de la raíz "cantar".

El uso de la lematización o stemming puede ser más o menos útil, de acuerdo al idioma en el que se trabaje. El inglés es una lengua que presenta pocas flexiones (lengua analítica), mientras que en el español se cuenta con gran cantidad de morfemas por palabra (lengua sintética).

Tab. 2.1: Ejemplo de lematización y stemming para varias palabras con la misma raíz (cantar).

Palabra	Stemming	Lema
cantábamos	cant	cantar
canción	cancion	cantar
cantantes	cantant	cantar

La **conversión a minúsculas** es otro tipo de normalización. En algunas tareas de NLP como algunos casos de recuperación de información, el convertir a minúsculas no afecta tanto el análisis como puede afectarlo para tareas de análisis de sentimientos o reconocimiento de Entidades, ya que el uso de mayúsculas puede aportar pistas en estas tareas (generalmente se escriben los nombres propios en mayúsculas o se escriben palabras completas en mayúsculas para dar énfasis).

Existe además un conjunto de palabras denominadas ***stop words***, que son palabras que no contienen un significado importante para ser utilizadas en el modelo de lenguaje. Por lo general, estas palabras se eliminan del texto, ya que agregan una gran cantidad de información innecesaria. Cada idioma tiene su propia lista de *stop words*, pero generalmente contienen las palabras que se usan comúnmente en el idioma. En el caso del inglés son palabras como "*as, the, be, are*", y para español incluye palabras como "*a, un, el, la, en*".

2.1.2 Segmentación

La normalización también incluye la separación del texto en unidades más pequeñas. La tarea de separación o *tokenización* de los textos se puede realizar de diferentes maneras. Normalmente, las palabras se encuentran separadas entre sí por espacios en blanco, pero los espacios en blanco no siempre son suficientes. Existen casos particulares, como la palabra "Nueva York", que representan un solo concepto a pesar del hecho de que contiene un espacio. Para el procesamiento de textos se utilizan diferentes técnicas de tokenización. Se puede considerar a los *tokens* como representación de palabras, o secuencias de palabras (n-gramas de palabras), o extender el enfoque a secuencias de caracteres (n-gramas de caracteres), e incluso a *skip*-gramas.

n-gramas

Los n-gramas se refieren al proceso de combinar las palabras que se encuentran juntas con fines de representación, donde n representa el número de palabras que se desea combinar. Por ejemplo, si consideramos la oración:

El coche circulaba a gran velocidad

Un modelo de 1-grama, o unigrama, tokenizará la oración palabra por palabra, dando como resultado:

El	coche	circulaba	a	gran	velocidad

Similarmente, en la tokenización en bigramas se combinan de 2 palabras para formar un token, para nuestro ejemplo esto es:

El coche	coche circulaba	circulaba a	a gran	gran velocidad

De manera análoga, se puede tokenizar un texto mediante secuencia de caracteres. Por ejemplo, al tokenizar la oración "El coche circulaba" por pentagramas de caracter, se tendrá:

el co	l coc	coch	coche	oche	che c	he ci
e cir	circ	circu	ircul	rcula	culab	ulaba

k-skip-n-gramas

El proceso de tokenización en k-skip-n-gramas consiste en tomar las cadenas de texto que se forman entre "saltos" de palabras, es decir, omitiendo palabras que se encuentren entre ellas. Formalmente, un k-skip-n-grama es una subsecuencia de longitud n donde los componentes ocurren a una distancia máxima de k entre sí. Un 2-skip-1-grama incluye todos los bigramas que se forman al saltar una palabra, siguiendo el ejemplo anterior tendremos el siguiente resultado:

el circulaba	coche a	circulaba gran	a velocidad

2.2 Métodos de representación de documentos

Para que un algoritmo de clasificación pueda caputrar relaciones entre datos requiere pasar de un conjunto de textos a datos estructurados. Existen diversas formas de representar los documentos en un esquema estructurado, particularmente en este trabajo analizaremos diversos métodos desde dos enfoques: el léxico y el semántico. A continuación se abordan, de manera general, ambos enfoques.

2.2.1 Enfoque léxico

El léxico es el conjunto de palabras que conforma un determinado idioma. En el Procesamiento del Lenguaje Natural, a través de la normalización de los textos, es posible representar con unidades léxicas (los tokens) el contenido de un conjunto de documentos. Las características que se pueden extraer de la representación de los documentos en los tokens se conocen como características léxicas. En este apartado se presentan los modelos léxicos más comunes y los que se utilizan en este trabajo.

La **bolsa de palabras** (Bag of Words) es una de las formas más comunes de representar los datos en un formato tabular donde en las columnas se representa el vocabulario total del corpus y cada fila representa un documento. La celda (intersección de la fila y la columna) representa el recuento de la palabra representada por la columna en ese documento en particular. De esta manera, se pueden aplicar algoritmos sobre los datos para construir modelos predictivos.

Esta representación funciona muy bien en algunas tareas de aprendizaje automático como la detección de spam, el clasificador de sentimientos, entre otros. Sin embargo, hay dos grandes inconvenientes de esta representación:

- Ignora el orden y la gramática de los documentos, por lo que se pierde el contexto en el que se usa una palabra.

- La matriz generada por esta representación es altamente dispersa y sesgada hacia las palabras más comunes; las palabras con mayor frecuencia son las más comunes, y no siempre aportan información del significado del texto.

Para atender el segundo punto, surge la representación **TF-IDF**, la cual es una forma de ponderar las palabras del vocabulario para dar un peso en proporción al impacto que tiene en el significado de un documento. La puntuación es un producto de 2 medidas independientes: la frecuencia de término (TF) y la frecuencia inversa de documento (IDF). La ponderación para un token, o palabra, i en el documento j se calcula de acuerdo a la ecuación (2.1).

$$w_{ij} = \underbrace{tf_{ij}}_{TF} \times \underbrace{log\left(\frac{N}{df_i}\right)}_{IDF} \tag{2.1}$$

donde tf_{ij} es el número de ocurrencias del término i en el documento j, df_i es el número de documentos que contienen el término i y N es el número total de documentos. La IDF es una medida de cuánta información proporciona el token, es decir, si es común o raro en los documentos.

Existen además, **modelos** basados en rasgos o **características léxicas**, los cuales siguen la idea de caracterizar a los documentos mediante una listas de atributos que resumen los rasgos más significativos o relevantes para la tarea que se quiere estudiar. Estos modelos convierten los textos a una representación numérica para cada documento a través de estadísticas de texto (por ejemplo la longitud de palabra promedio, proporción de palabras largas/cortas, entre otras) y por características sintácticas (a nivel de gramática, como el uso de verbos, adjetivos o adverbios, por nombrar algunos ejemplos).

La construcción de estas características depende del fenómeno de estudio, por ejemplo, en la tarea de análisis de sentimiento en textos, una variable (característica) importante es el número de palabras positivas, el número de palabras negativas o alguna relación entre estas; sin embargo, esta variable puede no ser significativa para la segmentación de contenidos. Para la detección de ironía, en la sección 3.2 se hace una recopilación de características comúnmente utilizadas, y en la sección 5.3.1 se exponen, a mayor detalle, el conjunto de características que consideramos en la construcción de un modelo basado en características léxicas (al que llamamos el enfoque léxico).

2.2.2 Enfoque semántico

El término semántica se refiere a los aspectos del significado, sentido o interpretación de signos lingüísticos como símbolos, palabras, expresiones o representaciones formales. A pesar de que los modelos desarrollados desde el enfoque semántico se basan también en los tokens, estos modelos buscan capturar la mayor cantidad de información posible del contexto en una palabra, tratando de capturar información semántica.

Desde el enfoque semántico, los **word embeddings** son modelos utilizados para transformar las palabras a una representación estructurada. En estos modelos las palabras se mapean a un espacio vectorial n-dimensional, de manera que palabras semánticamente similares se encontrarán cerca entre ellas.

Existe una gran gama de propuestas de *word embeddings* que han mejorado el comportamiento de muchas tareas de Procesamiento de Lenguaje Natural, cómo es la identificación de nombres, identificación de idiomas, traducción automática, entre otros. En la sección 5.3.2, se dará una explicación más completa sobre cómo funcionan los *word embeddings* y cuáles son los modelos que utilizamos para esta tesis.

Detección de Ironía

3

3.1 Definición ironía

La ironía es una forma sofisticada de uso del lenguaje en la que se reconoce una incongruencia entre el sentido literal de una expresión y el significado que el emisor pretende dar. De acuerdo a cómo es la inconsistencia en el mensaje, la ironía se puede dividir en dos grandes categorías: la ironía situacional y verbal (Forslid y Wikén, 2015).

Ironía situacional o cómica: Se refiere a las instancias que describen situaciones que no cumplen con algunas expectativas o que presentan una incongruencia aguda entre nuestras expectativas de un suceso y lo que ocurre. Algunos ejemplos de ironía situacional son los siguientes:

- Los bomberos que tienen un incendio en su cocina mientras están afuera para responder a una alarma de incendio.

- Un día como el de hoy 14 de marzo, pero de 1883 falleció Karl Marx. Lo enterraron en el cementerio de Highgate, Londres. y visitar su tumba cuesta 4 euros. Visitar la tumba de Adam Smith es gratis . #ironia zurda

Ironía verbal: esta categoría se aplica a los casos que contienen una expresión evaluativa cuya polaridad (positiva, negativa) se invierte entre el sentido literal y el deseado. Es decir, en aquellos textos en que se dice lo contrario de lo que quiere expresar. Como ejemplos tenemos:

- No te preocupes Salo, se que será difícil, pero gracias a estos partidos de gran nivel, nuestra selección subirá y mejorará su juego y así podremos a aspirar a una actuación histórica en el próximo mundial de Qatar #Not #Sarcasmo

- No quiero terminar el día sin darle las gracias a mi peluquera por destrozar el pelo que tanto tiempo me costó cuidar. #gracias #ironía

Al hacer uso de la ironía, el emisor espera que el receptor perciba el valor irónico del mensaje sin necesidad de indicaciones explícitas. Generalmente, el receptor reconocen una incongruencia

entre una expresión y lo que se sabe (o se espera) sobre el hablante basado en un contexto o conocimiento adicional. Sin embargo, a diferencia de la comunicación oral, en la que el emisor emplea una serie de códigos que evidencian la ironía, en la comunicación escrita no existe un consenso sobre patrones que revelen la intención irónica.

Una forma particular de ironía verbal es el sarcasmo. En el sarcasmo, la persona busca dar a entender algo contrario o diferente de lo que se dice, pero a manera de burla, es decir, es una categoría negativa de ironía verbal focalizada en la ofensa de algo o alguien. Si bien, en el estado del arte se ha abordado el problema de detección de sarcasmo como una tarea diferente a la de detección de ironía, en este trabajo se abordará la ironía verbal en general, sin discriminar por la presencia de sarcasmo.

3.2 Antecedentes

La detección automática de ironía ha sido estudiada ampliamente en los últimos años. Principalmente se han desarrollado trabajos en textos en inglés, pero se pueden encontrar referencias en idiomas como italiano (Barbieri et al., 2014), checo (Ptáček et al., 2014), griego (Charalampakis et al., 2016), español (Jasso y Meza, 2016), entre otros.

Los primeros trabajos para abordar este problema se formularon a partir de características basadas en la interpretación de la ironía, o de la estructura que se considera sigue esta figura literaria. Los modelos se basaban en un enfoque léxico, pues utilizaban, principalmente, como fuentes de información o características, la bolsa de palabras, patrones sintácticos, información sobre sentimientos o elementos de POS-TAGGING.

Entre los trabajos más destacados se encuentra el de Reyes et al., 2013, que aborda el problema de la detección de la ironía como una tarea de clasificación; los autores propusieron un modelo multidimensional que emplea cuatro tipos de características conceptuales: marcadores o firmas textuales (signos de puntuación, emoticons, ect), imprevistos (desequilibrio temporal en el texto cuando se encuentra un cambio entre tiempos presente y pasado ó desequilibrio contextual para capturar inconsistencias semántica de los conceptos de un texto), estilo y escenarios emocionales. Con este enfoque buscan caracterizar las propiedades textuales de la ironía para identificarla de política, educación y humor. En este trabajo se utilizan textos extraídos de Twitter, mismos que contienen *hashtag*[1] específicos: *#irony, #education, #humor, y #politics*; los cuales sirven a los autores como etiquetas de categorías. De esta manera, no se realiza un etiquetado manual de los textos ni se determina un tipo de ironía en particular.

[1]Palabra o frase precedida por un símbolo de numeral (#), utilizado en las redes sociales, especialmente en Twitter, para identificar mensajes sobre un tema específico.

De los primeros trabajos en abordar la tarea para otro idioma está Carvalho et al., 2009, quienes introdujeron pistas para detectar automáticamente ironía utilizando textos, en idioma portugués, generados por usuarios. Aunque este estudio se centra únicamente en reconocer la ironía en oraciones aparentemente positivas que involucran entidades humanas (personas), las características utilizadas pueden ser empleadas en cualquier otro contexto. El conjunto de datos empleado es una recopilación de opiniones generadas por usuarios en el sitio web de un periódico portugués popular. Así, distinguen en 4 categorías: comentarios irónicos, no irónicos, ambiguos y dudosos. Entre sus características más significativas en la detección de ironía se encuentran: la puntuación especial, las comillas y las expresiones de risa.

Un modelo para la detección de ironía en italiano es el de Barbieri et al., 2014, los cuales consideran cinco tipos de características: Frecuencia (palabra más rara, promedio de frecuencia de las palabras del texto, y la brecha entre estas dos medidas), Sinónimos, Ambigüedad, Partes del discurso y Sentimientos. En su trabajo, construyeron un corpus con textos de Twitter, y las etiquetas de ironía fueron asignadas a los tuits de cuentas consideradas como sarcásticas o irónicas en materia de política, y la etiqueta de no ironía a los tuits publicados en las cuentas de los 7 periódicos más populares del país. Así, los textos tanto irónicos como los no irónicos tienen los mismos temas: política y noticias.

En árabe, Karoui et al., 2017 proponen un modelo para detección de ironía en Twitter que se compone por cuatro grupos de características: características de superficie (signos de puntuación, emoticons, comillas, nexos adversativos, entre otros), sentimiento (palabras positivas/negativas), cambio (exageraciones) y contexto interno (verifican la presencia/ausencia de pistas contextuales internas, como la presencia de pronombres personales). Debido a que la política es uno de los temas más criticados en las redes sociales, estos autores construyen su corpus con tuits que contengan los nombres de los candidatos para la presidencia de Estados Unidos: Trump y Hillary, y los candidatos en las elecciones de Egipto: Al-sis, Moubarak y Morsi. Las etiquetas, al igual que en otros casos, se asignaron de acuerdo a la presencia del *hashtag irony*, y se considera el mismo número de textos en cada categoría.

Sin embargo, los modelos basados en características léxicas no han logrado buenos resultados en la tarea de desambiguación entre interpretación literal e irónica. Debido a que una misma oración puede interpretarse como irónica en algunos contextos, y genuina en otros, la diferenciación en la tarea debe considerar información externa a la expresión, es decir, conocimiento contextual y otras suposiciones de fondo, y no tanto la forma o el contenido de la expresión en sí. Así, el enfoque semántico busca mejorar la modelación de la ironía incorporando información del contexto al significado de las palabras. Desde este enfoque, se ha trabajado principalmente con el uso de *word embeddings*.

Desde este enfoque, el trabajo de Joshi et al., 2016 analiza la similitud/discordancia entre los *word embeddings* de cada texto para modelar particularmente el sarcasmo en textos generados por usuarios. El fundamento de este trabajo es que la similitud/discordancia en los vectores que representan cada palabra es indicativo de similitud semántica, la presencia de discordancia se asocia a incongruencia del contexto, característica común en la presencia de sarcasmo. El conjunto de datos utilizado fue construido a partir opiniones de usuarios en una página web, donde los usuarios pueden etiquetar el tipo de contenido que publican, incluida una etiqueta para *sarcasmo*; para la categoría de no sarcasmo se utiliza la etiqueta de *filosofía*.

Para el caso particular del español, los trabajos en la detección de ironía aun son escasos. El trabajo más significativo en español fue presentado por Jasso y Meza, 2016. Son además, los primeros en construir un corpus de ironía en español. En su trabajo abordan la tarea de detección de ironía desde el enfoque léxico con un modelo basado en n-gramas, y desde un enfoque semántico a través de word embeddings (word2vec).

3.3 Definición del problema

Desde la perspectiva de clasificación computacional, la ironía se ve como un problema de clasificación supervisada, generalmente binaria, para el cual se busca determinar las características y el algoritmo de clasificación que separe las categorías. Para el propósito de este trabajo entenderemos por **modelo de detección de ironía** al conjunto de características y algoritmo que permiten clasificar un texto en si es irónico o no.

Tradicionalmente se ha recurrido a Twitter para extraer información generada por usuarios y crear los corpus con los que se evalúa el desempeño de los modelos de detección de ironía. Sin embargo, los experimentos planteados difieren de trabajo en trabajo, haciendo que los resultados observados en las métricas no sean comparables directamente. Los principales factores que diferencian los experimentos son:

- Definición de ironía: es necesario determinar si la tarea de detección de ironía considerará toda clase de ironía o se enfocará solo en la ironía verbal, pues esto modifica el experimento. Además, algunos autores no consideran al sarcasmo como una forma de ironía.

- Subjetividad a la hora de etiquetar un corpus: A falta de una definición precisa de ironía, la tarea de etiquetado de las categorías (ironía/no ironía) tendrá siempre un grado de subjetividad. De manera que, para cada corpus, la tarea que en realidad se plantea es de detectar la ironía según la definición del etiquetador.

- Determinación de la categoría “No ironía”: la elección de los textos que se consideran en la categoría de no ironía son decisivos para determinar qué características serán influyentes en el problema de clasificación. En algunos trabajos la categoría de ironía se contrasta contra un tema en particular mientras que en otros casos se utilizan textos del mismo conjunto de usuarios pero que no contengan las palabras clave.

- La distribución de las categorías: en los trabajos revisados, el rendimiento de los clasificadores depende también de qué tan balanceadas están las categorías.

A manera de resumen, la Tabla 3.1 presenta los resultados obtenidos en los trabajos mencionados en la sección 3.2 [2]. Aún cuando los experimentos llevados a cabo para cada trabajo difiere de los demás, los resultados nos permite entender la complejidad de la tarea.

Tab. 3.1: Resumen de la revisión de literatura.

Autor	Idioma	Datos	Balanceados	Accuracy	Precision	Recall	F1 score
Reyes et al., 2013	Inglés	Twitter (política, educación, humor, ironía)	Si	0.723	0.736	0.695	0.715
Reyes et al., 2013	Inglés	Twitter (política, educación, humor, ironía)	No	0.804	0.661	0.447	0.533
Barbieri et al., 2014	Italiano	Twitter (política)	Si		0.75	0.76	0.76
Jasso y Meza, 2016	Español	Twitter	Si				0.68
Jasso y Meza, 2016	Español	Twitter	Si				0.78
Karoui et al., 2017	Árabe	Twitter (política)	Si	0.723	0.728	0.707	0.718

Tomando en cuenta lo anterior, este trabajo busca estudiar los modelos existentes en el estado del arte, principalmente en la extracción y/o generación de características, para comparar el desempeño de los modelos de acuerdo al tipo de características que utilizan y analizar el desempeño de modelos basados en una mezcla de características. Se trabaja con textos en inglés y textos en español, en este último no se no se ha trabajado mucho, por lo que se espera crear un marco de referencia sobre la dificultad que la tarea representa en este idioma.

[2] La definición de las métricas se proporciona en la Sección 5.2

Los datos

4

Existen muchas fuentes de información para extraer textos generados por usuarios, las redes sociales, blogs, foros de preguntas, secciones de comentarios en periódicos o páginas de productos, entre otros. En particular, Twitter se ha convertido en un recurso muy utilizado para obtención de datos en el estudio de la detección de ironía, debido a que es de fácil acceso y proporciona un método rápido y conveniente para encontrar potenciales mensajes irónicos haciendo uso de las etiquetas que los mismos usuarios asignan al contenido, es decir, los *hashtags* (como #irony, #not y #sarcasm).

En esta tesis se trabaja con un corpus en inglés y uno en español. Es importante señalar que, a pesar de que los datos provienen de Twitter en cada idioma, los resultados no son comparables debido a las características de los corpus que se detallan a continuación.

4.1 Corpus en inglés

El conjunto de datos en inglés con el que trabajamos se obtuvo de la Tarea 3 en el 12º Taller internacional sobre evaluación semántica (SemEval-2018, por sus siglas en inglés); esta tarea tenía como propósito la detección de ironía en tuits en inglés (Van Hee et al., 2018).

Para dicha tarea se extrajeron tuits en inglés filtrando por aquellos mensajes que contenían los *hashtags* #irony, #sarcasm y #not en cualquier parte del texto. El periodo de recolección fue entre diciembre de 2014 y marzo de 2015. Para minimizar el ruido introducido por *hashtags* de ironía sin fundamento, todos los tuits fueron revisados y etiquetados manualmente. La limpieza del conjunto de datos consistió en: eliminar retuits, duplicados y tuits que no estaban en inglés, y reemplazar los caracteres de escape XML (por ejemplo, &). En total se obtuvieron 3,000 tuits correspondientes a $2,676$ usuarios únicos.

El proceso de anotación del corpus fue realizado por tres estudiantes de lingüística hablantes de inglés como segunda lengua; cada estudiante hizo la anotación de un tercio del corpus. Después de las anotaciones se determinaron $2,396$ tuits (de los $3,000$) como irónicos, mientras que 604 no fueron considerados así. En un paso de limpieza adicional, se eliminaron aquellos tuits ambiguos, es decir, que requieren un contexto adicional para comprender su naturaleza irónica, quedando

al final $1,911$ tuits. Para equilibrar la distribución de clases en el corpus, se agregaron $1,792$ tuits para la categoría de no irónicos; los cuales fueron recopilados del mismo conjunto de usuarios de Twitter que los usados en el corpus con *hashtag* de ironía, y dentro del mismo periodo de tiempo.

El corpus final consiste en $3,834$ tuits ($1,911$ irónicos y $1,923$ no irónicos). La distribución de los tuits fue un $70\,\%$ de los datos en el conjunto de entrenamiento y $30\,\%$ para el conjunto de prueba. Para ambos conjuntos, las categorías (ironía/no ironía) se encuentran balanceadas.

4.2 Corpus en español

Para el caso de español, utilizamos el Corpus creado por Jasso y Meza, 2016, el cual fue generado siguiendo la misma metodología que el corpus en inglés, es decir, mediante el filtrado de tuits que contenían los *hashtag* #ironía #sarcasmo.

Para la parte del corpus correspondiente a la categoría de ironía, se consideraron los tuits con el *hashtag* #ironía #sarcasmo, escritos en español. En el artículo solo se consideraron los *hashtag* como señal de presencia de ironía; sin embargo, después de la publicación de su artículo, el corpus fue revisado y validado manualmente, descartando como irónicos algunos tuits con el *hashtag*. Esto ocasiona que los resultados del artículo ya no sean comparables directamente con lo que se expone en este trabajo, pero continúa siendo la principal referencia en el caso del idioma español.

Los tuits no irónicos fueron recolectados mediante una búsqueda con términos que no se relacionaran con la etiqueta de ironía, utilizando términos como *quién, cómo, cuándo, cómo, dónde* y *por qué*, de manera que cualquier tuit que no esté etiquetado explícitamente como irónico se considera no irónico. Este tipo de etiquetado tiene la intención de no sesgar a la clase de no ironía hacia determinado tema o dominio. Para la parte del corpus que corresponde a los tuits no irónicos no se realizó la validación manual. De esta manera, el corpus total consiste en $38,258$ tuits ($3,806$ irónicos y $34,452$ no irónicos), es decir que se cuenta con un conjunto de datos muy desbalanceado, con solo el $10\,\%$ de los datos en la categoría de irónicos.

Para tener un panorama más amplio de la detección de ironía en español, el Corpus en español fue tratado de dos maneras:

- Corpus Balanceado: Se realizó un subsampling sobre la categoría de “No ironía”; de manera aleatoria se obtuvieron 5 subconjuntos de la clase de no ironía del mismo tamaño de la case de ironía, y se unieron con los tuits irónicos para crear 5 corpus de tamaño $7,612$ y con

clases balanceadas. De ahí se separan los conjuntos de entrenamiento y prueba dejando las proporciones del caso en inglés (70-30).

- Corpus no Balanceado: Dejamos el conjunto de datos tal cual está, con la proporción de $10\,\%$ de los datos en la categoría de irónicos. Este corpus se dividió, de manera aleatoria, en un $50\,\%$ de datos de entrenamiento y $50\,\%$ para datos de prueba.

La idea de realizar un corpus balanceado de esta manera es que nos permite realizar un comparativo entre el problema de detección de ironía entre los textos en inglés y los textos en español. Los resultados no serán comparables directamente, pues no fueron etiquetados por las mismas personas, y la definición utilizada para considerar ironía no es exactamente la misma. Sin embargo, al tener características similares en la estructura del corpus y la fuente de obtención de los datos, es una buena aproximación para analizar si la caracterización de la ironía varía conforme al idioma; es decir, si para el caso de español algún enfoques es más o menos efectivo en contraste con los resultados que se tienen en inglés.

4.3 Preprocesamiento

El preprocesamiento de los textos es un paso esencial en cualquier tarea de clasificación de texto, los pasos descritos en esta sección se realizaron en ambos idiomas.

Debido a que el enfoque léxico se construye a través de la forma en que los usuarios escriben, es necesario hacer el preprocesamiento en dos etapas, la primera consiste en:

- Los nombres de usuarios, los @user de los tuits se reemplazan con una etiqueta especial única.

- Omitimos del análisis las URLs, correos electrónicos y números.

- En cuanto a las *stop words*, debido a que los textos con los que trabajamos son muy cortos, las *stop words* no se eliminan. Además, consideramos que los modelos semánticos pueden capturar información de dichas palabras en el contexto.

- No realizamos lematización ni stemming, debido a que las flexiones pueden tener información útil para la detección de ironía. En los textos en español, que es una lengua sintética, se pierde mucha información al analizar solo la raíz de las palabras.

Estas transformaciones se realizan para quitar información no relevante para los propósitos de esta tesis, por ejemplo las ligas a páginas o comentarios no son de utilidad pues no es posible extraer la información que contienen dichas ligas. En la Figura 4.1 se ilustra la transformación que se realiza a los textos en la primera etapa, cabe mencionar que hasta este punto los signos de puntuación no se retiran, pues esta información se toma en cuenta en los modelos de detección de ironía que se utilizan en este trabajo.

(a) Ejemplo 1

Texto original:	Texto procesado:
@josecdiez Los economistas actuales estáis al tanto de esta teoría? https://goo.gl/PwXzc7 Nadie habla de la problemática del interés, ¿xq?	usuario Los economistas actuales estáis al tanto de esta teoría? Nadie habla de la problemática del interés, ¿xq?

(b) Ejemplo 2

Texto original:	Texto procesado:
@ramona_almonte @joeltorresrd Descuida... que el próximo que venga por 4 años va a resolver todo eso...	usuario usuario Descuida... que el próximo que venga por años va a resolver todo eso...

Fig. 4.1: Ejemplo de la primera etapa del preprocesamiento en los textos.

Con los textos hasta esta etapa se obtienen las características léxicas (detalladas en la Sección 5.3.1). Después de esto, se realizan la segunda parte del preprocesamiento, que consiste en:

- Los hashtags se procesan eliminando el símbolo especial # y manteniendo el texto.
- Omitimos los signos de puntuación.
- Unificamos los textos, convirtiendo todos los caracteres a minúsculas.
- Como paso final, tokenizamos los textos en unigramas. Utilizamos una representación de palabras para explotar tanto la información semántica como lingüística de cada palabra en la medición de ironía en el texto.

La Figura 4.2 muestra un ejemplo, para los textos en español, de cómo cambian los textos de la primera a la segunda etapa del preprocesamiento. Si bien los ejemplos que se presentan son en los textos en español, las mismas técnicas fueron utilizadas en los textos en inglés.

Para cada documento del corpus pasamos de tener iniciar con una cadena de texto a tener un vector de tokens, cuyo tamaño depende del número de palabras que quedaron después del preproceso completo. Así, para el texto en la Figura 4.2a, el vector asignado al texto es de tamaño 12, mientras que en para el texto en la Figura 4.2 el tamaño del vector es 5.

(a) Ejemplo 1

Resultado primera etapa:	Representación final:
La bolsa europea se tambalea, pero viva el NO de Grecia. ¡viva!	[la, bolsa, europea, se, tambalea, pero, viva, el, no, de, grecia, viva]

(b) Ejemplo 2

Resultado primera etapa:	Representación final:
Maravilloso. Estrenando calcetines. #Irene #bautizo	[maravilloso, estrenando, calcetines, irene, bautizo]

Fig. 4.2: Ejemplo de la segunda etapa del preprocesamiento en los textos en español.

Con esta representación vectorial de los textos se construyen los modelos semánticos descritos en la Sección 5.3.2.

5 Metodología

En este Capítulo se describen los modelos que se utilizarán en este trabajo, tanto desde un enfoque léxico como uno semántico, para analizar cómo puede caracterizarse la ironía. Asimismo, se presentan las herramientas utilizadas para la modelación. Los mismos modelos se aplicarán a los textos en inglés y en español. Además, se exponen las métricas utilizadas para medir el desempeño de los modelos.

5.1 Baseline

Para tener un punto de comparación con los modelos presentados en este trabajo, se obtuvo (para ambos idiomas) un modelo de referencia (baseline) utilizando un clasificador de textos multipropósito llamado microTC (Eric S. Tellez et al., 2018).

MicroTC (μTC) es capaz de categorizar texto independientemente del dominio e idioma. Su objetivo es explorar una serie de transformaciones de los textos, determinar un esquema de ponderación en la representación de los textos y producir una clasificación efectiva a través de la Maquina de Soporte Vectorial (SVM).

Las funciones de transformación que microTC realiza en los textos pueden ser simples, como la eliminación de símbolos diacríticos o convertir en minúsculas todas las letras, pero también puede hacer uso de técnicas sofisticadas como la lematización, la corrección de errores ortográficos, entre otros. Después, se realiza la representación de los textos a vectores de alta dimensión a través de *tokens*, los cuales pueden ser por n-gramas de palabras o de caracteres, ó mediante skip-gramas. Para cada *token* del vocabulario se obtiene el número de ocurrencia en cada documento, y se genera una matriz de características en la que cada documento se representa en función de las veces que aparecen los *tokens* del vocabulario, esto es la frecuencia término (TF). La frecuencia inversa de documento (IDF) es una medida de si el *tokens* es común en el corpus completo. Se obtiene dividiendo el número total de documentos por el número de documentos que contienen el *token*. De manera que microTC decide si representar los documentos desde un enfoque TF, IDF o como una combinación de ellos (TF-IDF). Por último, ya con un esquema de ponderación sobre tokens, se aplica SVM como algoritmo de aprendizaje automático sobre un

conjunto de datos de entrenamiento buscando minimizar una función de error o maximizar una función score.

La tarea que resuelve microTC puede ser vista como un problema de optimización combinatoria, donde hay un espacio de búsqueda que contiene todas las combinaciones posibles de diferentes transformaciones de texto, procedimientos de ponderación con sus respectivos parámetros y, en este espacio de búsqueda, se utiliza una meta-heurística para buscar una configuración que produzca una alta eficacia.

Por tratarse de un modelo que no requiere ningún contexto sobre el problema y que puede aplicarse a cualquier conjunto de textos sin importar el idioma, se considera como un modelo de referencia para este trabajo.

5.2 Evaluación del modelo

Considerando la tarea de detección de ironía como un problema de clasificación binaria, el rendimiento del modelo se mide como la capacidad de predicción en los datos de prueba. Las predicciones ahora se dividen en cuatro grupos en función de la comparación entre la categoría real a la que pertenecen los datos y el valor de la predicción: predicciones positivas correctas (verdadero positivo), predicciones negativas correctas (verdadero negativo), predicciones positivas incorrectas (falso positivo) y predicciones negativas incorrectas (falso negativo).

Los resultados del modelo serán analizados utilizando las métricas de evaluación estándar descritas a continuación.

Accuracy

La Exactitud (Accuracy) es simplemente la proporción de datos clasificados correctamente. Es la primera métrica que se comprueba al evaluar un clasificador; sin embargo, si los datos están desbalanceados o si se está más interesado detectar una de las clases, la Exactitud no captura realmente la eficacia de un clasificador. Esta se calcula como en la ecuación (5.1).

$$Accuracy = \frac{VP + VN}{TOTAL} \tag{5.1}$$

donde VP es el número de predicciones positivas correctamente clasificadas, VN es el número de predicciones negativas correctamente clasificadas y $TOTAL$ es el número total de casos u observaciones.

Precision

La precisión (Precision) nos ayuda a cuantificar la fracción de positivos verdaderos entre el total de los clasificados como positivos. Se utiliza principalmente para medir qué tan efectivo es el modelo en detectar la categoría de interés, es decir, la categoría positiva; se obtiene con la ecuación (5.2).

$$\textit{Precisión (Precision)} = \frac{VP}{VP + FP} \tag{5.2}$$

donde FP es el número de predicciones positivas calificadas incorrectamente y VP es como en la ecuación (5.1).

Recall

La recuperación (Recall) mide cuántos positivos verdaderos se predijeron como positivos. La fórmula para calcularlo se presenta en la ecuación (5.3), donde se observa que mientras más cercano a 1 se encuentre el recall, más datos de la categoría real fueron bien clasificados.

$$\textit{Recuperación (Recall)} = \frac{VP}{VP + FN} \tag{5.3}$$

donde FN es el número de predicciones negativas calificadas incorrectamente y VP es como en la ecuación (5.1).

Medida F1

La medida F1 (F1 score) es la media armónica de la precisión y la recuperación, ecuación (5.4). Esta medida es una buena forma de resumir la evaluación en un número único.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.4}$$

Existe una relación entre la precisión y el recall: un modelo que predice todo como positivo tendrá un recall de 1, pero una precisión muy baja pues tendría muchos falsos positivos, mientras que un modelo que solo predijera un positivo y el resto negativos tendría un recall muy bajo, pero una precisión muy alta.

Es por ello que se recurre a la medida F1, ya que mitiga el impacto de las tasas altas y acentúa el de las tasas bajas.

5.3 Modelo de detección de ironía

Tanto para los textos en inglés como para los textos en español, se plantea el problema desde un enfoque léxico y uno semántico. De esta manera, se busca comparar el desempeño que tiene la incorporación de información semántica con respecto a lo que se puede obtener del enfoque léxico. Además, se plantea la combinación de ambos enfoques en un modelo híbrido. Los resultados son también comparados con el baseline descrito al inicio de este capítulo.

5.3.1 Enfoque léxico

Desde la perspectiva léxicas se busca caracterizar la ironía con base en patrones de los datos, es decir, en el cómo escribe la gente cuando quiere dar a entender ironía.

Dentro del conjunto de posibles características léxicas en los textos usados en el estado del arte, se identifican diferentes conjuntos de variables: características de superficie (signos de puntuación, emojis, comillas, nexos adversativos, entre otros), sentimiento (palabras positivas/negativas), cambio (exageraciones) y contexto interno (verifican la presencia/ausencia de pistas contextuales internas, como la presencia de pronombres personales).

Basados en este esquema, nuestra propuesta se centra en la forma en que se determinan las características de sentimiento, y complementadas con algunas correspondientes características de superficie y contexto interno.

Determinación de las características de sentimiento

En este trabajo se propone analizar los patrones de cambio en el sentimiento dentro de la oración. Una de las características que comúnmente se aprecia en una frase con ironía es el cambio de sentimiento dentro del texto. Por ejemplo, en la frase:

"Despertarse con la dulce voz de mi padre regañándome por que deje un plato sucio ayer ha sido lo mejor de mi mañana. "

se observa que al inicio el texto es positivo (señalado en color verde), y luego cambia el sentido a negativo (lo cual se aprecia en color rojo), para finalizar la frase con una idea positiva (en color verde).

Esto se ha abordado en la literatura de diferentes maneras, considerando como características el número de palabras positivas y/o negativas, la polaridad del texto, la presencia de conectores negativos (pero, sin embargo, etc.) entre otras. Nuestra propuesta es considerar subgrupos de palabras dentro de cada texto, e identificar el sentimiento asociado a cada uno. Como se puede observar en la Figura 5.1, para cada subgrupo de palabras se tendrá asociada una categoría: positivo, negativo o neutro. De manera que el sentimiento asociado a cada subgrupo se agregará al conjunto de variables de los textos para ser consideradas como características dentro del clasificador.

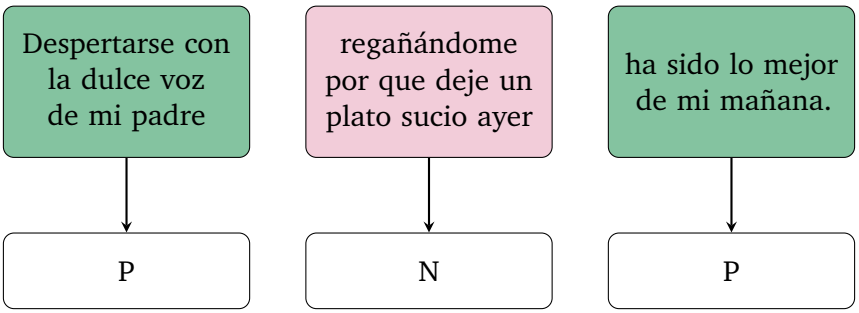


Fig. 5.1: Detección de sentimiento por subgrupos de palabras.

Para la determinación del sentimiento en los textos, utilizamos B4MSA (Eric S Tellez et al., 2017), que es un clasificador de análisis de sentimientos de python para textos cortos (como es el caso

de los tuits). Debido a que nos interesa usar el modelo en textos en inglés y en español, B4MSA resulta de gran utilidad, pues se puede utilizar en cualquier idioma, ya que es casi independiente del idioma, pero puede aprovechar las particularidades del mismo.

Características adicionales

Como complemento a las características de sentimiento de los textos, consideramos las siguientes variables léxicas para caracterizar a los documentos:

- Tamaño: Los textos se encuentran acotados debido a que Twitter tiene un límite de caracteres permitido. Por esta razón, el tamaño del tuit puede jugar un papel importante por sí mismo, pero también afecta a otras variables consideradas dentro de las características que consideramos. Para tomar en cuenta el efecto del tamaño del tuit, utilizamos el número de palabras en el tuit, el número de caracteres en el tuit y el promedio de letras en cada palabra como características.

- Marcadores textuales: Estas características son consideradas como pistas o recursos que se utilizan en la comunicación escrita para expresar emociones o dar énfasis. El uso de estos marcadores permite también al emisor denotar la presencia de ironía. Las características que consideramos son: Porcentaje del texto que está en mayúsculas, la presencia de signos de puntuación especiales ($! \cdot "? * ”$), la presencia de emojis y si los textos contienen *hashtag*; las últimas 3 variables fueron tratadas como dicotómicas, es decir que solo consideramos si hay ausencia o presencia de estas características.

- Entidades: Una de las partes más relevantes de un texto es saber de quién o de qué se está hablando. Con Entidades nos referimos a las personas, organizaciones o lugares. En el modelo, consideramos como característica una variable dicotómica para la presencia de Entidades en el tuit.

- Part-of-speech tagging (PoS Tagging): El etiquetado gramatical (PoS Tagging) explica cómo se usa una palabra en una oración, asignándole una etiqueta a cada palabra en las categorías: sustantivos, pronombres, adjetivos, verbos, adverbios, preposiciones, conjunciones e interjecciones. Con las características que consideramos dentro de este grupo se pretende capturar el estilo de los tuits, bajo la hipótesis de que los textos con ironía utilizan mucho los adjetivos y carecen de pronombres personales. Así, para cada tuit consideramos como variables la proporción de: verbos, sustantivos, adjetivos, adverbios y pronombres. Todas estas variables se encuentran acotadas en el intervalo $[0, 1]$.

Para dar claridad a cómo es que se extraen en estas variables, consideremos los ejemplos que se muestran en la Figura 5.2. Se muestra una serie de ejemplos de tuits de los corpus de español y de inglés con los que se trabajaron en esta tesis.

Lo primero que es importante notar es que al tratarse de textos extraídos de una red social, la forma de escribir no sigue las reglas de gramática, y los errores de ortografía son muy comunes, como se puede notar en el ejemplo 5.2a, donde el usuario escribió cada primera letra de la palabra en mayúscula, además de agregar una vocal de más a la última palabra. Asimismo, se puede observar que en este tipo de documentos se recurre a los signos de puntuación y mayúsculas para dar énfasis a cierta parte de los textos. Este comportamiento no es particular de algún idioma, y es este tipo de información la que se desea capturar con las variables léxicas.

(a) Ejemplo 1

A Mi Me Encanta Que Sea Bonito Por Face Y Que En La Vida Real Sea Bien Secoo !! ENAMORAA

(b) Ejemplo 2

CHICOS!!! Hoy echan en A Memorias de. . . . UN ASESINATO. No os lo podeis perder!!!!

(c) Ejemplo 3

user oh yeah!!! nd :D #HoldTheApplause hahaha lol <

Fig. 5.2: Ejemplos de tuits.

Por otro lado, y recordando que no se llevó a cabo ningún proceso de lematización o stemming, para el idioma español se tiene la particularidad de tener dos palabras para la misma conjugación verbal, como se observa en el ejemplo 5.2b, donde encontramos la frase "*No os lo podeis*" que es equivalente a "*No se lo pueden perder*". La diferencia radica en que el texto original fue escrito en la variante de español de España. Así, en el español, además de tener la conjugación de los verbos por la primera, segunda y tercera persona, se tiene una variante en la segunda persona dependiendo del país del emisor: vos o tú; lo que ocasiona que aunque el significado sea el mismo, al no ser la misma palabra, como características no sean iguales.

En la Tabla 5.2 se detalla el conjunto de características resultado para cada ejemplo presentado en la Figura 5.2. En las características de tamaño cabe mencionar que los signos de puntuación cuentan como caracter en el tuit pero no como palabra. Para el ejemplo 5.2b se puede observar que se detecta una entidad, que corresponde a la palabra "Memorias", debido a que la palabra fue escrita con mayúscula inicial, y basado en el algoritmo de etiquetado gramatical, se reconoce como una entidad; lo mismo sucede con la palabra "*HoldTheApplause*" en el ejemplo 5.2c.

Tab. 5.1: Características adicionales para los ejemplos de la Figura.

Características	Ejemplo 1	Ejemplo 2	Ejemplo 3
Número de palabras en el tuit	19	14	7
Número de caracteres en el tuit	88	66	50
Promedio de caracteres por palabra	3.45	3.47	4.2
Porcentaje del texto que está en mayúsculas	0.29	0.25	0.06
Signos especiales	1	1	1
Emojis	0	0	1
Hashtag	0	0	1
Entidades	0	1	1
Adjetivos	0.11	0	0
Adverbios	0.05	0.12	0
Verbos	0.16	0.24	0
Sustantivos	0.11	0.06	0.5
Pronombres	0.05	0.06	0

Una vez que se extraen todas las variables se crea la matriz de características léxicas. En la Figura 5.3 se resume el conjunto de características léxicas que son el insumo para los algoritmos de clasificación, el cual se conforma por 16 variables. Una característica importante de este conjunto de datos es que, por la naturaleza de las variables, no es muy homogéneo: en algunas variables solo hay valores 0 y 1, otras variables se encuentran entre 0 y 1 y otras son variables enteras.

Fig. 5.3: Resumen de características consideradas en el enfoque léxico.

5.3.2 Enfoque semántico

En general, los modelos creados desde el enfoque léxico provocan que las palabras pierdan el sentido del contexto; por ejemplo, en un esquema de bolsa de palabras que contiene las palabras perro, gato y azul, existe la misma relación entre "perro" con "gato" que con "azul". Desde el enfoque semántico se busca usar la relación entre las palabras y sus contextos, a fin de que con ello se pueda caracterizar la ironía. Los modelos semánticos utilizados en este trabajo se basan en modelos probabilísticos del lenguaje.

Modelos del lenguaje

En el lenguaje natural, las palabras no aparecen de manera aislada o aleatoria, dependen de otras palabras y van formando una secuencia de acuerdo con una estructura gramatical. Los modelos del lenguaje tratan de capturar este condicionamiento, con el objetivo de poder entender y predecir la estructura lingüística en elementos como frases u oraciones. Además, estos modelos buscan capturar la relación existente entre las palabras. Así, un modelo de lenguaje puede ser representado por la probabilidad condicional de la siguiente palabra dada las anteriores, como

$$\hat{P}(w_1^T) = \prod_{i=1}^{T} \hat{P}(w_t|w_1^{t-1}), \tag{5.5}$$

donde w_t es la t-ésima palabra, y la subsecuencia $w_i^j = (w_i, w_{i+1}, \ldots, w_{j-1}, w_j)$.

Con esta representación, se puede expresar la función de probabilidad como un producto de probabilidades condicionales de la siguiente palabra dada las anteriores, es decir que para determinar la probabilidad tomamos en cuenta un contexto de T palabras. Esta función tiene parámetros que se pueden ajustar de forma iterativa para maximizar la probabilidad de los datos de entrenamiento o algún criterio de regularización.

Dentro de los modelos del lenguaje, la incorporación del aprendizaje profundo ha generado un importante avance para el Procesamiento del Lenguaje Natural. El modelo del lenguaje natural propuesto por Bengio et al., 2003, es un modelo basado en una red neuronal profunda para estimar las probabilidades de transición a partir de n-gramas. A grandes rasgos, el enfoque que propone se resume en tres puntos principales:

- Representar cada palabra del vocabulario a partir de un vector de características (un vector con entradas en $\mathbb{R}^m$).

- Expresar la función de probabilidad conjunta de secuencia de palabras en términos de los vectores de características de las palabras en la secuencia.

- Aprender simultáneamente los vectores de características y los parámetros de la función de probabilidad.

A partir de un conjunto de entrenamiento, una secuencia de $w_1 \cdots w_T$ de palabras, con $w_t \in V$, donde el vocabulario V es un conjunto grande pero finito, el objetivo es entrenar un modelo $f(w_t, \cdots, w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$. Como la Figura 5.4 muestra, el modelo consiste en utilizar una función C que toma una palabra $w_t \in V$ y la lleva a una representación vectorial $C(w_t)$, esta función se representa por una matriz C de dimensión $|V| \times m$, con m el hiperparámetro de dimensionalidad de los vectores. Estos vectores se pasan a una capa oculta, donde los $(n-1)$ vectores de contexto se representan como un sólo vector, la activación de esta capa oculta se da por medio de una función tangente hiperbólica. Posteriormente, se usa la función Softmax para determinar una probabilidad de que, dado el contexto, siga una palabra del vocabulario. De manera que la salida es un vector de dimensión $|V|$, donde el i-ésimo elemento es la probabilidad de que la $w_t = i$ dado el contexto.

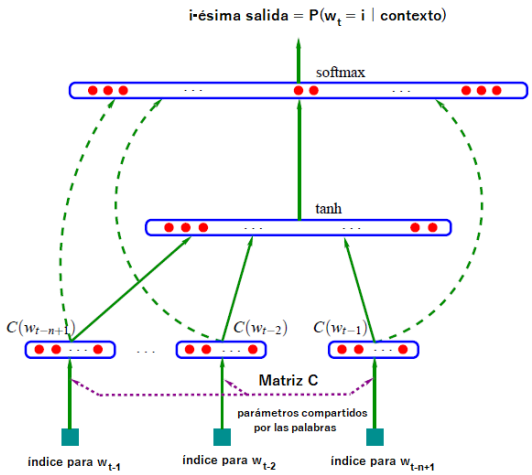


Fig. 5.4: Arquitectura Neuronal para estimar la distribución de probabilidad.

Fuente: Bengio et al., 2003

Este modelo fue el parteaguas para lo que hoy en día se conoce como los modelos de espacio vectorial, o *word embeddings*, debido a que al mismo tiempo que se estiman las probabilidades, se obtiene una representación vectorial de las palabras, la cual permite calcular dichas probabilidades.

Pero fue con el trabajo de Mikolov, Sutskever et al., 2013 que la idea de plasmar información semántica en un espacio vectorial a partir de redes neuronales se volvió más popular. La arquitectura neuronal que propusieron es conocida como *word2vec*.

En la arquitectura propuesta en Bengio et al., 2003, se buscaba inferir la probabilidad de una palabra dado un contexto. Es decir, la entrada de la red neuronal eran palabras contexto y su salida era la probabilidad de que una palabra w_t del vocabulario correspondiera a dicho contexto; a este enfoque se le llamado *Continous Bag-of-Words* o CBOW. En el trabajo de Mikolov, Sutskever et al., 2013 se toma como entrada una palabra, w_t, y se infiere la probabilidad del contexto, a este enfoque se conoce como *skip-gram*. En la Figura 5.5 se presenta, a manera de comparativo, arquitectura de ambos modelos.

El modelo *word2vec* utiliza una red neuronal para estimar las probabilidades. Una vez entrenada la red, utiliza los pesos de la capa oculta como las representaciones vectoriales de las palabras (los *word vectors*), de ahí que el modelo es llamado *word2vec*, pues convierte una palabra en un vector.

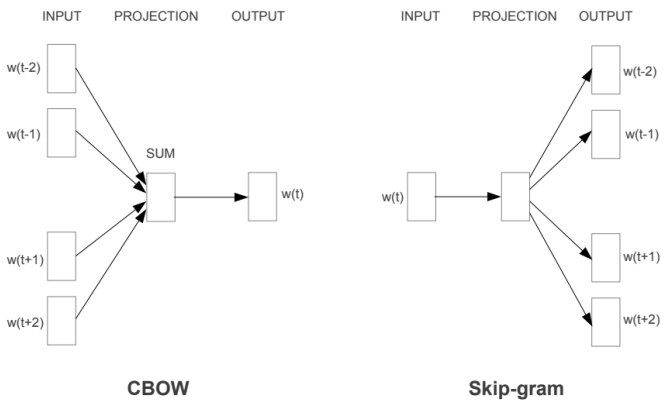


Fig. 5.5: Arquitectura de los modelos CBoW y Skip-gram.

Fuente: Mikolov, Chen et al., 2013

A partir de estas arquitecturas, otros métodos basados en redes neuronales se han propuesto. En este trabajo utilizamos dos modelos de *word embeddings*: Doc2vec (Le y Mikolov, 2014) y ELMo (Peters et al., 2018).

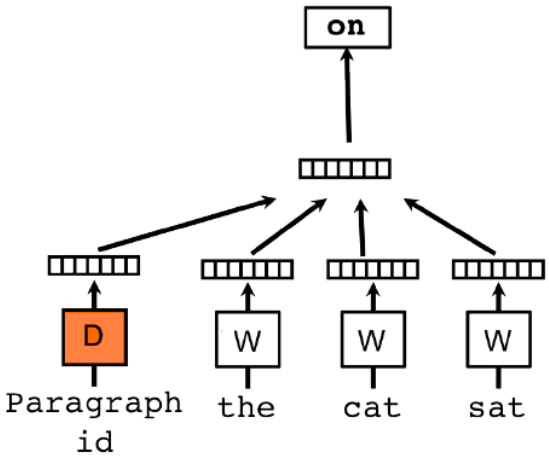


Fig. 5.6: Esquema para la estimación de la representación vectorial de los párrafos en doc2vec.

Fuente: Le y Mikolov, 2014

Doc2Vec

El modelo *Paragraph Vector*, conocido también como *doc2vec*, está inspirado en los trabajos de Bengio et al., 2003 y Mikolov, Sutskever et al., 2013, para llevar la representación en espacios vectoriales a estructuras de un nivel más alto que las palabras, como las oraciones o párrafos.

El objetivo de doc2vec es crear una representación vectorial de un documento, independientemente de su longitud. Para tratar con ello, al modelo de word2vec se agregada otro vector, que corresponde al párrafo. Como se observa en la Figura 5.6, la concatenación del vector de párrafo (D) con el contexto de las $n-1$ palabras se usa para predecir la n-ésima palabra. El vector de párrafo, que es un único para cada documento, representa la información faltante del contexto y actúa como la memoria del tema del párrafo.

De manera que cuando se entrenan los vectores de palabras (W), el vector del documento (D) también se entrena; al final del entrenamiento, se tiene una representación numérica del documento. Una vez entrenados, los vectores párrafo se pueden ver como características del párrafo, por ello es llamado *doc2vec*.

Estos vectores de párrafo heredan una propiedad importante de los vectores de palabras: la semántica. De manera que vectores de párrafos cercanos se asocian con documentos que, semánticamente, tienen un contenido similar. Y es por esta característica que creemos que el modelo de doc2vec puede ser de utilidad en la tarea de detección de ironía.

ELMo

El *Embeddings from Language Models (ELMo)* es un modelo de representación vectorial profundo desarrollado por Peters et al., 2018, en el que se tiene una representación contextualizada de las palabras. Busca modelar, al mismo tiempo, las características complejas del uso de palabras (por ejemplo, sintaxis y semántica), y cómo los usos varían en los contextos lingüísticos (es decir, para modelar la polisemia).

A diferencia de los *word embeddings* previamente expuestos, las representaciones de palabras de ELMo son funciones de la oración de entrada completa. ELMo se basa en los modelos de lenguaje bidireccionales (biLM).

Un biLM consisten en la combinación de dos partes: un modelo de lenguaje hacia adelante (*forward LM*) y un modelo de lenguaje hacia atrás (*backward LM*).

Dada una secuencia de N palabras $(w_1, w_2, \ldots, w_N)$, un LM hacia adelante calcula la probabilidad de una de la secuencia mediante el modelado de la probabilidad de la palabra w_k dada la historia $(w_1, w_2, \ldots, w_{k-1})$:

$$p(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} p(w_k | w_1, w_2, \ldots, w_{k-1})$$

La representación de las palabras se obtiene con word embeddings o mediante el uso de una red neuronal convolucional (CNN), y se pasan a través de L capas de una red neuronal LSTM (Long Short Term) hacia adelante. En cada posición k, cada capa LSTM genera una representación dependiente del contexto $\overrightarrow{h}_{kj}^{LM}$ donde $j = 1, \ldots, L$. La salida LSTM de la capa superior, $\overrightarrow{h}_{kL}^{LM}$, se utiliza para predecir la siguiente palabra t_{k+1} con una capa Softmax.

Un LM hacia atrás es similar a un LM hacia adelante, excepto que se ejecuta en la secuencia en sentido inverso, prediciendo la palabra anterior dado el contexto futuro:

$$p(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} p(w_k | w_{k+1}, w_{k+2}, \ldots, w_N)$$

La implementación es análoga al caso de una LM hacia adelante, con cada capa LSTM hacia atrás generando una representación $\overleftarrow{h}_{k,j}^{LM}$ de w_k dado $(wk+1,\ldots,w_N)$.

Las representaciones de palabras obtenidas a través de ELMo son profundas, ya que son una función de todas las capas internas del biLM. La combinación de estos estados internos permite tener representaciones de palabras muy ricas. Así, algunos estados de la LSTM capturan aspectos dependientes del contexto del significado de las palabras (por ejemplo, se pueden usar para tareas de desambiguación del sentido de las palabras) mientras que otros estados modelan aspectos de sintaxis (por ejemplo, se pueden usar en tareas como *PoS Tagging*).

En este trabajo, utilizamos los vectores pre-entrenados de ELMo proporcionadas por Che et al., 2018, que fueron entrenados con un corpus de 20 millones de palabras como parte de la tarea CoNLL 2018 (wikidump + common crawl) para cada idioma. En nuestro caso, solo usamos la capa superior para cada palabra, y luego usamos el promedio como una función para agrupar las palabras dentro de un texto.

A diferencia de otros enfoques de representación vectorial, con ELMo podemos obtener múltiples incrustaciones para cada palabra, según el contexto en el que se usó la palabra. La incorporación de los ELMo ha mostrado significativas mejoras en algunas tareas como respuesta a preguntas, vinculación textual y análisis de sentimientos. El objetivo en este trabajo es analizar si también se puede mejorar el desempeño de los modelos en la tarea de detección de ironía utilizando los vectores contextualizados de ELMo.

5.4 Clasificadores

En esta tesis, abordamos el problema de detección de ironía desde el punto de vista del Aprendizaje Supervisado. Una vez que se extraen las características expuestas en las secciones 5.3.1 y 5.3.2 para los textos, asignamos los textos a las categorías de ironía o no ironía, mediante el uso de un algoritmo de clasificación.

En términos generales, no existe un clasificador o tipo de clasificador que sea mejor para todas las tareas, depende siempre del problema que se esté estudiando. El desempeño de cada algoritmo puede variar debido a factores como el tamaño de la muestra, número de características, la naturaleza de los datos, cantidad de categorías, entre otros.

Puesto que estaremos usando distintos conjuntos de características, probamos diferentes tipos de clasificadores. Para la selección de los algoritmos se tomaron en cuenta los clasificadores más utilizados en el estado del arte (SVM y Random Forest), y se agregaron algoritmos que se

han desempeñado bien en diversas tareas de aprendizaje supervisado (Regresión Logística y AdaBoost). Los modelos como Regresión Logística y SVM son útiles cuando la relación entre la variable dependiente (clases) y las independientes (características) se aproxima a un modelo lineal. Sin embargo, si la relación es compleja y altamente no lineal, entonces los árboles de decisión tendrán mejores resultados de que un método lineal.

Máquinas de soporte vectorial (SVM)

Las SVM fueron introducidos como algoritmo de aprendizaje por Cortes y Vapnik, 1995. Es un algoritmo de clasificación lineal, el cual para un conjunto de datos de k clases, en un espacio N dimensional, induce separadores lineales o hiperplanos (N–1) dimensionales para separar esos puntos, ya sea en el espacio original o en una transformación del mismo (mediante Kernels), en k grupos.

Para separar las k clases dentro de los datos existen múltiples hiperplanos posibles, ya que un hiperplano se puede mover un poco hacia arriba o hacia abajo sin tocar ninguna de las observaciones, estos hiperplanos no-óptimos se observan en la parte izquierda de la Figura 5.7. La distancia entre cada hiperplano y la observación más cercana a cualquiera de las clases se conoce como el margen. Como se ilustra en la parte derecha de la Figura 5.7, el objetivo de SVM es encontrar un hiperplano que tenga el margen máximo, ya que al maximizar el margen se tiene más confianza para que los datos fuera de la muestra de entrenamiento puedan clasificarse correctamente. Los vectores de soporte son los datos que se encuentran más cerca del hiperplano separador, e influyen en la posición y orientación del mismo. Estos vectores de soporte se utilizan para maximizar el margen del clasificador.

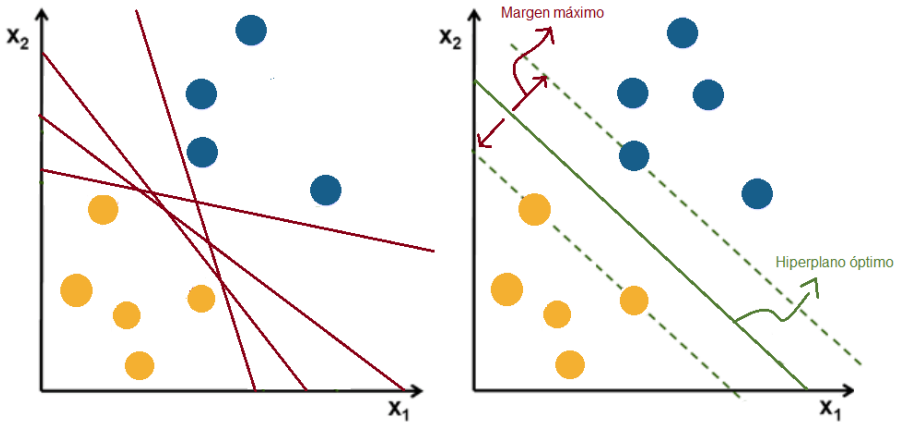


Fig. 5.7: Esquema de funcionamiento de SVM.

Regresión Logística

Es un modelo de regresión que permiten estudiar si una variable, generalmente binomial, depende de un conjunto variables o características. Desde la perspectiva de aprendizaje supervisado, la regresión logística es un algoritmo de gran utilidad en los problemas de regresión y de clasificación. En esta tesis, utilizamos la regresión logística para el problema de clasificación de los textos.

Dado un problema de clasificación binario, con categorías (0,1), la regresión logística resuelve la tarea aprendiendo a partir de un conjunto de entrenamiento (x), un vector de pesos (β) y un término de sesgo (ϵ). Cada peso β_i es un número real y está asociado con una de las características de entrada x_i. El peso β_i representa cuán importante es x_i para la decisión de clasificación. Para asignar los valores predichos a probabilidades se utiliza la función sigmoide ((5.6)), la cual mapea cualquier valor real a un valor entre 0 y 1.

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.6}$$

donde $z = \beta x + \epsilon$.

Para la determinación de la clase, se requiere pasar de estas probabilidades a valores discretos (clases). Estos se obtiene al asignar un valor de umbral o punto de inflexión en las probabilidades, de manera que por encima de dicho umbral o frontera de decisión clasificaremos la observación a la clase 0 y debajo de dicho valor asignaremos el dato a la clase 1.

Random Forest

Los árboles de decisión son un tipo de algoritmo de aprendizaje supervisado principalmente usados en problemas de clasificación. La construcción del árbol sigue un enfoque de división binaria recursiva, donde la tasa de error de clasificación se utiliza como criterio para la división binaria. En cada proceso de división analiza la mejor variable para ramificación sólo en el proceso actual, y se divide la muestra en conjuntos basados en la variable de entrada más significativa.

Como mejora a los árboles de decisión, surgen los modelos de ensamble, que como su nombre lo dice, consisten en combinan varios árboles de decisión para producir un mejor rendimiento predictivo que utilizar un solo árbol de decisión. El principio fundamental detrás de estos modelos es que la unión de muchos clasificadores débiles pueden formar un clasificador más fuerte.

Al construir un árbol de decisión pequeño se obtiene un modelo con baja varianza y alto sesgo. Se puede reducir el sesgo de predicción al incrementar la complejidad del modelo, pero haciendo que la varianza aumente. El uso de modelos de ensamble es una forma de lidiar con el ajuste entre varianza y sesgo. Los principales modelos de ensamble son Bagging, Random Forest y Boosting.

La introducción de los bosques aleatorios (Random Forest) se realizó por primera vez en Breiman, 2001, aunque su trabajo se vio influenciado por los trabajos de Ho, 1995, Amit y Geman, 1997 y Dietterich, 2000. Random Forest es un método de ensamble que combina múltiples árboles de decisión independientes probados sobre conjuntos de datos aleatorios con igual distribución. La idea esencial es promediar este conjunto de árboles ruidosos pero aproximadamente imparciales, y por tanto reducir la varianza. Como se muestra en la Figura 5.8, el método consiste en generar varios árboles, los cuales consideran aleatoriamente un subconjunto del total de las características para la división. Al final del proceso, en cada árbol se asigna una categoría a cada observación, esto es, cada árbol genera un voto por una categoría. Y para determinar la categoría final se considera que aquella con mayor número de votos en todo el bosque (forest) es la categoría, o clase, asignada.

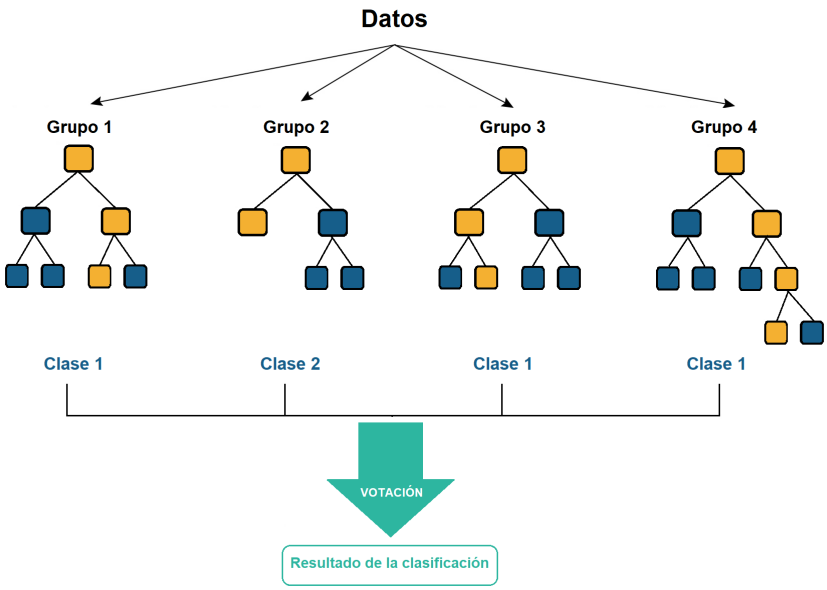


Fig. 5.8: Esquema del funcionamiento de Random Forest.

AdaBoost

Su nombre es una contracción de *Adaptive Boosting*. Estos algoritmos fueron formulados por Freund y Schapire, 1997, buscando encontrar una hipótesis fuerte a partir de utilizar hipótesis simples y débiles.

AdaBoost es una forma de aprendizaje secuencial, la idea es entrenar un modelo con todo el conjunto de entrenamiento, y los modelos posteriores se construyen ajustando los valores de error residual del modelo inicial. De esta manera, se intenta dar mayor peso a aquellas observaciones que el modelo anterior estimó pobremente, con lo que el algoritmo se adapta y logra obtener mejores resultados.

El funcionamiento del algoritmo se describe en la Figura 5.9. En el intento (1), el clasificador hace un corte tratando de separar los datos, pero como se observa en la primera parte de la gráfica, se equivoca en 3 datos azules. En el siguiente intento, el peso de estos 3 datos azules mal clasificados se incrementa, de modo que el nuevo clasificador le da mayor importancia a clasificar bien estos 3 puntos. En el intento (2) logra discriminar correctamente todos los puntos azules, pero ahora se incrementa el peso para los puntos naranjas que clasificó mal, y los puntos que fueron bien clasificados reciben un peso menor. Se realiza un tercer clasificador que de igual manera tratará de hacer el mejor corte para resolver el problema. Al final, las predicciones hechas por los modelos son ponderadas por sus puntuaciones de precisión (α_i) y los resultados se combinan para crear una estimación final.

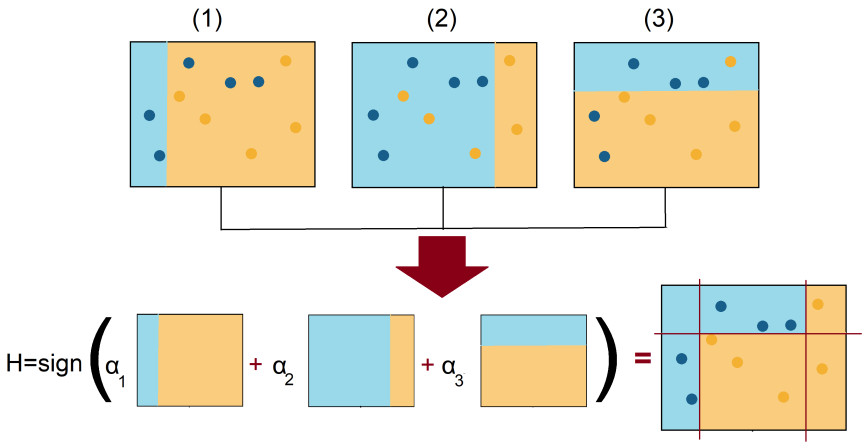


Fig. 5.9: Esquema del funcionamiento de AdaBoost.

Selección de parámetros

En la construcción de un modelo de clasificación, uno de los factores más importantes al momento de seleccionar un algoritmo de clasificación es determinar qué transformaciones se realizarán sobre los datos y el ajuste de los hiperparámetros propios de cada algoritmo, ya que la eficiencia de cada algoritmo depende en gran medida de la optimización de los hiperparámetros.

En este trabajo hacemos uso de Auto-sklearn (Feurer et al., 2015), que es una herramienta de aprendizaje automático que funciona de manera automatizada. Dado un conjunto de datos de

entrenamiento, Auto-sklearn determina pasos de preprocesamiento de los datos y establece los respectivos hiperparámetros para optimizar el desempeño de cada uno de los algoritmos de clasificación que se consideren.

A través de métodos de optimización bayesianos, Auto-sklearn ajusta un modelo probabilístico para capturar la relación entre la configuración del hiperparámetro y su rendimiento medido; luego utiliza este modelo para seleccionar la configuración de hiperparámetro más prometedora (intercambiando la exploración de nuevas partes del espacio en comparación con la explotación en regiones bien conocidas), evalúa esa configuración de hiperparámetro, actualiza el modelo con el resultado y lo repite. Al final del proceso, Auto-sklearn arroja la mejor configuración para cada clasificador.

En nuestro caso, se consideraron los valores de los hiperparámetros de cada clasificador considerando la configuración de los hiperparámetros que la herramienta determinó; de este modo, podemos comparar los algoritmos de clasificación con los mejores parámetros en cada caso.

6 Resultados y Análisis

En este capítulo se presenta el desempeño de los modelos propuestos en el Capítulo 5. Para los datos en inglés se presenta el comparativo con los resultados del ganador del concurso SemEval, así como las referencias bibliográficas. Para el caso del idioma español, los trabajos de detección de ironía son muy escasos, y no se cuenta con referencias de desempeño de modelos basados en características léxicas. Por esta razón, los resultados obtenidos en este trabajo para los tuits en español pueden ser de utilidad en el entendimiento de la tarea de detección de ironía en este idioma. Aun cuando no se trabajó con exactamente el mismo corpus, los resultados en español se comparan con el resultado obtenido por Jasso y Meza, 2016 a manera de referencia bibliográfica.

Los resultados para español se presentan utilizando los dos corpus mencionados en la Sección 4.2. El conjunto de datos balanceado se contrasta con los resultados en inglés, para analizar si la caracterización de la ironía varía conforme al idioma; mientras que el conjunto no Balanceado se compara solo con el Corpus Balanceado.

Los modelos de detección de ironía fueron implementados en Python. Para ambos idiomas se utilizaron las características descritas en la Secciones 5.3.1 y 5.3.2, y presentamos los resultados utilizando los 4 clasificadores mencionados en la Sección 5.4 (SVM, Regresión logística, Random Forest y AdaBoost), cada uno con los parámetros obtenidos mediante la optimización utilizando Auto-sklearn.

6.1 Inglés

Considerando únicamente las características léxicas, podemos observar en la Tabla 6.1 que el mejor desempeño se tiene utilizando Random Forest, superior al resto de los clasificadores en todas las métricas. En el caso de SVM, éste logra un buen resultado en la precisión (0.68) pero con un Recall (0.6181) más bajo, lo cual se debe a que tiene menos casos de Falso positivo. AdaBoost tiene mayor cantidad de textos irónicos clasificados correctamente, pero la mayor cantidad de textos fueron clasificados como irónicos.

Tab. 6.1: Desempeño de los clasificadores con las características léxicas en textos en inglés.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.6985	0.6886	0.6181	0.6166
Reg. Logística	0.6916	0.6683	0.6209	0.6226
Random Forest	0.7228	0.7063	0.6651	0.6723
AdaBoost	0.7055	0.6791	0.6624	0.6673

En el segundo ejercicio utilizamos, para cada documento, las características semánticas obtenidas a través de doc2vec. En la Tabla 6.2 se presenta el desempeño de cada clasificador, en este caso tenemos que el mejor desempeño se obtiene al utilizar Random Forest, aunque la diferencia con SVM y la Regresión logística no varía mucho. Utilizando los embeddings de doc2vec, el clasificador con el peor desempeño es AdaBoost, el cual tuvo un alto número de aciertos en los textos irónicos, pero también un alto número de textos no irónicos mal clasificados (FP), lo que se ve reflejado en la precisión.

Tab. 6.2: Desempeño de los clasificadores con las características semánticas con Doc2vec en textos en inglés.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.7176	0.702	0.6554	0.6617
Reg. Logística	0.7246	0.7076	0.6686	0.6759
Random Forest	0.7107	0.6854	0.6784	0.6812
AdaBoost	0.6872	0.6607	0.6590	0.6598

El tercer ejercicio consiste en utilizar los embeddings obtenidos mediante ELMo, Tabla 6.3. En este caso, Regresión Logística es el clasificador con mejor resultado. SVM obtiene métricas similares, pero tiene mejor predicción en la clase de no ironía, lo que afecta un poco más al Recall, y por consiguiente se observa una medida F1 más baja. Por su parte, Random Forest es el clasificador que logra etiquetar correctamente más elementos de la categoría de no ironía, pero los textos irónicos no los pudo etiquetar correctamente. Al igual que en el tercer ejercicio, AdaBoost fue el modelo que menor desempeño presentó.

Tab. 6.3: Desempeño de los clasificadores con las características semánticas con ELMo en textos en inglés.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.7142	0.6906	0.6656	0.6717
Reg. Logística	0.7202	0.6971	0.6760	0.6821
Random Forest	0.7159	0.7169	0.6379	0.6402
AdaBoost	0.6638	0.6305	0.6214	0.6241

Por último, decidimos probar un modelo híbrido, basado en una combinación de los dos enfoques. El objetivo es analizar si las características léxicas añaden información a los modelos semánticos,

para complementar el conjunto de características. Los resultados se presentan en la Tabla 6.4, donde se tiene que el mejor modelo se tiene usando como clasificador SVM y juntando las características léxicas con los vectores que se obtienen de doc2vec. Este modelo es considerablemente mayor que el resto de los analizados, pues es el único caso en el que se supera el 0.7 en la medida F1, además de tener también el mejor Recall y precisión. En cuanto a los modelos que combinan características léxicas con los vectores de ELMo, el mejor desempeño lo tiene el modelo que utiliza como clasificador a la Regresión logística. Sin embargo, ninguno de los modelos utilizando los vectores de ELMo superan el resultado obtenido con los vectores obtenidos con doc2vec.

Tab. 6.4: Desempeño de los clasificadores considerando tanto las características léxicas como las semánticas en textos en inglés.

Modelo	Clasificador	Accuracy	Precision	Recall	F1 score
Léxicas + doc2vec	SVM	0.7524	0.7484	0.6950	0.7053
Léxicas + doc2vec	Reg. Logística	0.7368	0.7251	0.6802	0.6888
Léxicas + doc2vec	Random Forest	0.7072	0.6817	0.6762	0.6785
Léxicas + doc2vec	AdaBoost	0.7011	0.6764	0.6761	0.6763
Léxicas + ELMo	SVM	0.7388	0.7151	0.6809	0.6888
Léxicas + ELMo	Reg. Logística	0.7246	0.7011	0.6888	0.6932
Léxicas + ELMo	Random Forest	0.7176	0.7190	0.6403	0.6431
Léxicas + ELMo	AdaBoost	0.7202	0.7374	0.6362	0.6363

A manera de resumen, los resultados obtenidos con el mejor clasificador para cada enfoque se presentan en la Tabla 6.5. Además, se reportan los resultados de microTC (nuestro baseline) y los resultados reportados por Van Hee et al., 2018 como los ganadores del concurso. Este último utiliza un enfoque de Deep Learning (Baziotis et al., 2018), basado en dos redes neuronales BiLSTM: una a nivel de palabra y otra a nivel caracter, para capturar la información semántica y sintáctica en los tuits.

Tab. 6.5: Comparativo de mejor resultado obtenido por enfoque para los tuits en inglés.

Modelo	Clasificador	Accuracy	Precision	Recall	F1 score
Léxicas	Random Forest	0.7228	0.7063	0.6651	0.6723
doc2vec	Reg. Logística	0.7246	0.7076	0.6686	0.6759
ELMo	Reg. Logística	0.7298	0.7132	0.6763	0.6840
Híbrido (léxicas-doc2vec)	SVM	0.7524	0.7484	0.6950	0.7053
MicroTC	SVM	0.8575	0.8606	0.8576	0.8572
Baziotis et al., 2018	Red LSTM	0.735	0.630	0.801	0.705

En general, para los datos en inglés se obtuvieron resultados muy parecidos en los 3 modelos propuestos: el modelo de características léxicas, el modelo semántico basado en doc2vec y el modelo semántico utilizando los vectores de ELMo. Los modelos semánticos obtuvieron un desempeño ligeramente mayor que el modelo léxico propuesto, siendo el modelo ELMo el que presenta las mejores métricas.

Asimismo, se encontró que agregar las características léxicas al modelo semántico de doc2vec ayuda a mejorar el nivel de predicción tanto de los tuits irónicos como los no irónicos, lo que se refleja en un aumento en todas las métricas. Siendo el enfoque híbrido el único que sobrepasa el 0.7 en la medida F1, equiparándose con el resultado obtenido por los ganadores del SemEval 2018.

Por otro lado, es importante recalcar que en ninguno de los modelos propuestos se logra superar el baseline (microTC), el cual considera únicamente información léxica de los tuits.

6.2 Español - Corpus Balanceado

Para los textos en español se realizaron los mismos 4 ejercicios que con los datos en inglés, descritos en la sección 6.1. Sin embargo, como se realizó un proceso de muestreo sobre el corpus original, los resultados presentados en esta sección se obtuvieron del promedio de los resultados obtenidos en cada uno de los 5 corpus creados. Esto a fin de reducir el efecto de la muestra que se tuvo.

Tomando en cuenta solo las características léxicas, el desempeño de los clasificadores es similar, siendo AdaBoost el algoritmo que mejor logra clasificar los datos. Para cada clasificador, la relación entre la precisión y el recall es parecida, es decir que los errores de clasificación como falsos positivos y falsos negativos fueron, en cada clasificador, más o menos iguales, lo que se observa en la Tabla 6.6.

Tab. 6.6: Desempeño de los clasificadores con las características léxicas en textos en español.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.6204	0.6214	0.6211	0.6203
Reg. Logística	0.5998	0.6017	0.6009	0.5994
Random Forest	0.6357	0.6367	0.6364	0.6356
AdaBoost	0.6449	0.6452	0.6452	0.6449

La Tabla 6.7 muestra los resultados obtenidos cuando utilizamos el primer enfoque semántico, basado en doc2vec. En comparación con el enfoque léxico los resultados son bastante parecidos, aunque el enfoque de doc2vec logra un mejor desempeño en todos los clasificadores. En este ejercicio, el mejor resultado se tiene con Random Forest, el cual logra identificar casi dos terceras partes de los textos de manera correcta. El peor resultado se tiene con regresión logística, en todas las métricas se encuentra muy por debajo del resto de los clasificadores.

Para los embeddings obtenidos mediante ELMo, la Tabla 6.8 muestra el desempeño de los 4 clasificadores. Los resultados son muy superiores a los alcanzados en los dos ejercicios anteriores.

Tab. 6.7: Desempeño de los clasificadores con las características semánticas con Doc2vec en textos en español.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.6918	0.6917	0.6912	0.6913
Reg. Logística	0.6689	0.6709	0.6668	0.6659
Random Forest	0.6983	0.6984	0.6985	0.6983
AdaBoost	0.6944	0.6944	0.6945	0.6944

El peor desempeño lo tiene Random Forest, que es casi un punto porcentual mayor que el mejor modelo presentado utilizando los vectores de doc2vec. El mejor modelo es el que tiene como clasificador la Regresión logística, el 83 % de los casos son bien idetificados, y hay un balance entre los falsos positivos como los falsos negativos. Este resultado coincide con el mejor modelo para los textos en inglés usando el enfoque de ELMo.

Tab. 6.8: Desempeño de los clasificadores con las características semánticas con ELMo en textos en español.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.8271	0.8274	0.8275	0.8271
Reg. Logística	0.8336	0.8341	0.8341	0.8336
Random Forest	0.7688	0.7689	0.7691	0.7688
AdaBoost	0.7688	0.7709	0.7698	0.7687

Como en el caso del idioma inglés, se realizó un ejercicio de combinación de enfoques, utilizando las características léxicas con los vectores de doc2vec, y por otro lado combinando las características léxicas con los vectores de ELMo. En la Tabla 6.9 se observan los resultados.

Los resultados por tipo de clasificador varían en cuando incorporamos la información léxica a los vectores obtenidos mediante doc2vec: para SVM y Regresión Logística se mejora el desempeño al incluir la información léxica como variables adicionales (logrando que el modelo con SVM aumente más de un punto porcentual en la medida F1), sin embargo, para Random Forest y AdaBoost, la incorporación de la información léxica no se ve reflejado en una mejora los resultados para las métricas presentadas.

Por su parte, para los vectores obtenidos a través de ELMo, todos los clasificadores presentan mejoras al incluir las variables léxicas, aunque en diferente medida. AdaBoost es el clasificador al que más beneficia la incorporación de esta información, aunque el mejor modelo sigue siendo el que utiliza Regresión Logística como algoritmo de clasificación.

Tab. 6.9: Desempeño de los clasificadores considerando tanto las características léxicas como las semánticas en textos en español.

Modelo	Clasificador	Accuracy	Precision	Recall	F1 score
Léxicas + doc2vec	SVM	0.7053	0.7052	0.7052	0.7052
Léxicas + doc2vec	Reg. Logística	0.6852	0.6854	0.6855	0.6852
Léxicas + doc2vec	Random Forest	0.6966	0.6967	0.6968	0.6966
Léxicas + doc2vec	AdaBoost	0.6913	0.6916	0.6916	0.6913
Léxicas + ELMo	SVM	0.8363	0.8373	0.8369	0.8362
Léxicas + ELMo	Reg. Logística	0.8376	0.8385	0.8382	0.8376
Léxicas + ELMo	Random Forest	0.7723	0.7736	0.7731	0.7723
Léxicas + ELMo	AdaBoost	0.7898	0.7920	0.7908	0.7897

De todos los ejercicios realizados, este es el único caso en el que los algoritmos de clasificación varían en diferente dirección entre ellos. Lo que nos permite observar la capacidad que tienen métodos como SVM y Regresión Logística para trabajar con conjuntos de datos no tan grandes.

A manera de resumen, los resultados obtenidos con el mejor clasificador para cada enfoque se presentan en la Tabla 6.10. Además, se reportan los resultados de microTC (nuestro baseline) y los resultados obtenidos por Jasso y Meza, 2016.

Tab. 6.10: Comparativo de mejor resultado obtenido por enfoque para los tuits en español.

Modelo	Clasificador	Accuracy	Precision	Recall	F1 score
Léxicas	AdaBoost	0.6449	0.6452	0.6452	0.6449
doc2vec	Random Forest	0.6983	0.6984	0.6985	0.6983
ELMo	Reg. Logística	0.8336	0.8341	0.8341	0.8336
Híbrido (léxicas-ELMo)	Reg. Logística	0.8376	0.8385	0.8382	0.8376
MicroTC	SVM	0.9208	0.9254	0.9221	0.9207
Jasso & Meza	Random Forest				0.87

El modelo que contiene solo características léxicas es el que presenta los resultados más bajos. Los resultados que se obtienen al utilizar los vectores entrenados con doc2vec no mejoran sustancialmente al enfoque léxico. Además, al considerar un modelo híbrido observamos que agregar las características léxicas al modelo semántico mejora ligeramente el desempeño del modelo basado en las características semánticas de ELMo.

En cuanto al comparativo con los resultados obtenidos por Jasso y Meza, es difícil hacer una comparación, por no ser precisamente el mismo conjunto de datos, y por no contar más que con el resultado de la medida F1. Sin embargo, nos sirve de referencia, y nos permite inferir que se tiene un buen resultado con el modelo híbrido.

A pesar de ello, el mejor resultado lo tiene nuevamente microTC, al igual que se encontró con el corpus en inglés. El porcentaje de accuray es muy grande comparado con los otros modelos, pero también con relación a los trabajos anteriores tanto en español como en otros idiomas, incluido el inglés. Todas las métricas muestran resultados muy altos.

6.3 Español - Corpus No Balanceado

En esta sección se exponen los resultados obtenidos al utilizar el corpus de datos en español completo. Es importante señalar que debido al desbalance entre las categorías, el *accuracy* se ve sesgado por la poca presencia de datos de la categoría de ironía; el observar valores altos en esta métrica no puede ser considerado como un buen desempeño del modelo. Por lo tanto, en esta sección nos enfocaremos en el análisis de la precisión y recall y su impacto en la medida F1.

Como se puede observar en la Tabla 6.11, las características léxicas no pueden discriminar los textos irónicos de los no irónicos. El peor modelo es el que utiliza como clasificador Regresión Logística, pues prácticamente clasifica todo como textos no irónicos, lo que se refleja en que tanto la precisión como el recall sean muy bajos. El mejor clasificador en este ejercicio fue AdaBoost, que logra identificar más casos de ironía, aunque el valor de la medida F1 sigue siendo muy baja.

Tab. 6.11: Desempeño de los clasificadores con las características léxicas en textos en español. Corpus No Balanceado.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.9033	0.8575	0.5111	0.4968
Reg. Logística	0.9016	0.4508	0.5000	0.4741
Random Forest	0.9039	0.8302	0.5167	0.5077
AdaBoost	0.8990	0.6665	0.5324	0.5377

La Tabla 6.12 muestra los resultados para el enfoque semántico utilizando los vectores obtenidos a través de doc2vec. En comparación del enfoque léxico, con doc2vec se tiene una mejora sustancial en las métricas para todos los clasificadores. En particular, el modelo con Regresión Logística logra una precisión muy alta, la cual se debe a que tiene muy pocos falsos positivos, pero al tener muchos falsos negativos el recall, y por consiguiente la medida F1, es muy bajo. Al utilizar como clasificador Random Forest se obtiene el mejor resultado, no obstante, en contraste con los resultados obtenidos con el corpus balanceado tenemos un poder de clasificación muy bajo.

En el segundo enfoque semántico, el peor desempeño se tiene cuando utilizamos Random Forest como clasificador, en este ejercicio el algoritmo tiende a subestimar la categoría de ironía lo que se puede apreciar al tener tanto el recall como la precision muy bajas; este resultado es consistente

Tab. 6.12: Desempeño de los clasificadores con características semánticas Doc2vec en textos en español. Corpus No Balanceado.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.9028	0.7176	0.5610	0.5817
Reg. Logística	0.9049	0.9523	0.5165	0.5068
Random Forest	0.9004	0.7003	0.5817	0.6071
AdaBoost	0.9014	0.7063	0.5761	0.6009

con los obtenidos utilizando el corpus en español balanceado y el corpus en inglés. En este ejercicio, el clasificador que mejor desempeño tiene es SVM, el cual sobrepasa considerablemente al resultado de la Regresión Logística, que con el Corpus Balanceado y el Corpus en Inglés había obtenido el mejor resultado. Haciendo la comparación entre los resultados obtenidos con el corpus balanceado (Table 6.8), podemos notar una disminución importante en el poder predictivo para todos los clasificadores utilizando el corpus no balanceado (Tabla 6.13), siendo SVM el que menos disminuye, pues en el corpus balanceado reportaba una medida F1 de 0.8271, mientras que en el corpus no balanceado pasa a ser de 0.8112.

En general, con los resultados de las secciones 6.2 y 6.3 podemos ver que, para los textos en español, el enfoque semántico basado en la representación vectorial de ELMo se logra clasificar mejor que con los vectores entrenados a través de doc2vec.

Tab. 6.13: Desempeño de los clasificadores con características semánticas ELMo en textos en español. Corpus No Balanceado.

Clasificador	Accuracy	Precision	Recall	F1 score
SVM	0.9430	0.8822	0.7665	0.8112
Reg. Logística	0.9399	0.8862	0.7414	0.7928
Random Forest	0.9081	0.8537	0.5429	0.5551
AdaBoost	0.9144	0.7706	0.6704	0.7054

Por último, hacemos el ejercicio de mezclar las características léxicas con cada uno de los enfoques semánticos, los resultados se presentan en la Tabla 6.14. Para el caso de los vectores obtenidos con doc2vec, en general, el incorporar las características léxicas no aporta información relevante para la detección de ironía. Para la representación vectorial de ELMo con los clasificadores lineales, sí mejora el desempeño de los modelos al incorporar la información léxica.

Para resumir todos los resultados, en la Tabla 6.15 se muestran solo el mejor modelo para cada ejercicio. Se muestra también el resultados de microTC (nuestro baseline), el mejor resultado obtenido con el corpus Balanceado y los resultados obtenidos por Jasso y Meza, 2016.

Tab. 6.14: Desempeño de los clasificadores considerando modelo híbrido en textos en español. Corpus No Balanceado.

Modelo	Clasificador	Accuracy	Precision	Recall	F1 score
Léxicas + doc2vec	SVM	0.9057	0.8587	0.5267	0.5264
Léxicas + doc2vec	Reg. Logística	0.9049	0.9374	0.5172	0.5383
Léxicas + doc2vec	Random Forest	0.9006	0.7016	0.5811	0.6065
Léxicas + doc2vec	AdaBoost	0.9022	0.7125	0.5747	0.5996
Léxicas + ELMo	SVM	0.9444	0.8959	0.7630	0.8126
Léxicas + ELMo	Reg. Logística	0.9434	0.8885	0.7629	0.8104
Léxicas + ELMo	Random Forest	0.9071	0.8606	0.5350	0.5414
Léxicas + ELMo	AdaBoost	0.9053	0.9525	0.5186	0.5109

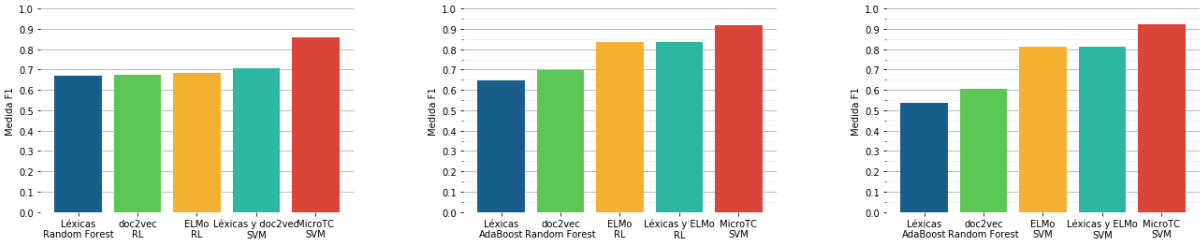
Tab. 6.15: Comparativo de mejor resultado obtenido por enfoque para los tuits en español. Corpus No Balanceado.

Modelo	Clasificador	Accuracy	Precision	Recall	F1 score
Léxicas	AdaBoost	0.8990	0.6665	0.5324	0.5377
doc2vec	Random Forest	0.9004	0.7003	0.5817	0.6071
ELMo	SVM	0.9430	0.8822	0.7665	0.8112
Híbrido (léxicas-ELMo)	SVM	0.9444	0.8959	0.7630	0.8126
MicroTC	SVM	0.9748	0.9611	0.8925	0.9233
Corpus Balanceado (Hib.)	Reg. Logística	0.8376	0.8385	0.8382	0.8376
Jasso y Meza, 2016	Random Forest				0.80

El peor desempeño se tiene cuando solo se consideran las variables léxicas. Cuando consideramos las características semánticas con los vectores de doc2vec se tiene una mejora con respecto al enfoque léxico, pero se observa una disminución de casi 10 puntos porcentuales con respecto al desempeño que tiene el mismo modelo pero con el corpus balanceado. Al considerar un modelo híbrido observamos que agregar las características léxicas al modelo semántico mejora ligeramente el desempeño del modelo basado en las características semánticas de ELMo. Nuevamente, el mejor resultado lo tiene microTC. En comparación con el corpus balanceado en donde microTC reporta una medida F1 de 0.9207, con el corpus no balanceado se tiene un F1 de 0.9233, que es incluso un poco mejor.

6.4 Resumen de resultados

Para concluir el Capítulo, se presenta en la Figura 6.1 un resumen del desempeño de cada enfoque con el mejor clasificador, basados en la medida F1, para los 3 corpus de textos utilizados.



(a) Inglés. **(b)** Español balanceado. **(c)** Español no balanceado.

Fig. 6.1: Resumen comparativo mejor resultado por enfoque para cada corpus.

Como es posible observar, de entre las propuestas de modelos, los mejores resultados se obtuvieron utilizando el enfoque semántico a través de los vectores de ELMo; siendo para los datos en español (Figuras 6.1b y 6.1c) donde se observa una mejora considerable con respecto al enfoque semántico basado en doc2vec.

De los 3 enfoques presentados, el peor desempeño se tiene cuando solo usamos las características léxicas. Este tipo de variables resulta más útil en el idioma inglés que en el español, como se aprecia en la Figura 6.1a, y fue el modelo más sensible a un desbalance de las categorías en el idioma español, en donde la medida F1 se situa debajo del 0.6 (Figura 6.1c).

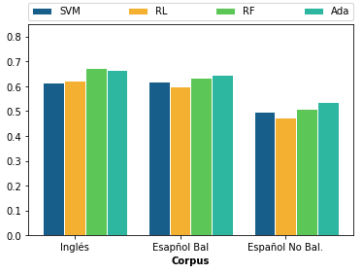
No obstante, al incluir las variables léxicas a los modelos semánticos se logra mejorar ligeramente el desempeño de los modelos semánticos, y es en el idioma inglés donde mejor se aprecia este resultado, Figura 6.1a.

En lo que corresponde al desempeño de microTC, la Figura 6.1 sustenta lo que se observó en las secciones 6.1, 6.2 y 6.3: el desempeño de microTC es superior a los enfoques presentados. Además, para los datos en español no se observa una disminución de ajuste cuando tomamos el corpus no balanceado.

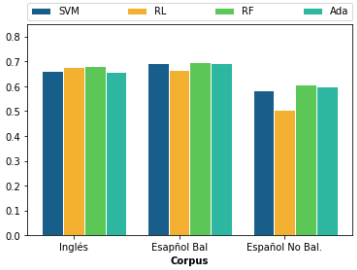
6.5 Desempeño de los clasificadores

Como un ejercicio adicional, analizamos el desempeño que tuvieron los clasificadores considerados en esta tesis en cada uno de los ejercicios que se llevaron a acabo. Debido a que cada enfoque presenta un conjunto de características diferentes, el objetivo de esta sección es examinar cómo afectan factores como tamaño de corpus, desbalanceo de clases y tipo de características en el desempeño de los algoritmos de clasificación. La Figura 6.2 muestra el comparativo del desempeño que tuvieron los 4 clasificadores para cada ejercicio, considerando únicamente los resultados de la medida F1.

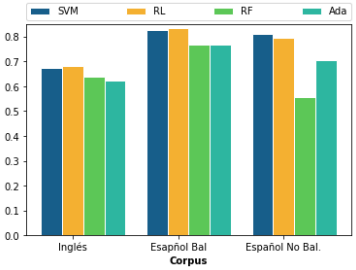
Los clasificadores utilizados en este trabajo son de dos tipos: SVM y Regresión Logística son clasificadores lineales, mientras que Random Forest y AdaBoost son algoritmos basados en árboles de decisión. En cuanto a tipo de clasificadores podemos ver en las Figuras 6.2c y 6.2e que los clasificadores lineales son más efectivos cuando se trata de clasificar las representaciones vectoriales de ELMo. Para los modelos basados en características léxicas, los clasificadores basados en árboles de decisión tuvieron un mejor desempeño (Figura 6.2a). Mientras que en el enfoque basado en doc2vec, los resultados no se inclinan hacia un tipo de clasificador en particular, como se aprecia en las Figuras 6.2b y 6.2d.



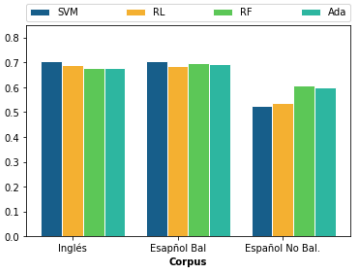
(a) Características léxicas



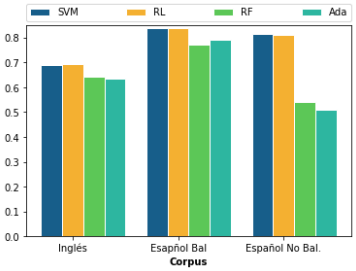
(b) Semántico - d2v



(c) Semántico - ELMo



(d) Híbrido - d2v



(e) Híbrido - ELMo

Fig. 6.2: Resultado de la medida F1 para cada clasificador, en cada corpus y modelo.

De los 4 clasificadores, tanto Random Forest como SVM fueron los que, en promedio, mejor desempeño tuvieron.

7 Conclusiones y trabajos Futuros

La finalidad de este trabajo era estudiar la ironía como un problema de aprendizaje supervisado, explorar los modelos utilizados en el estado del arte particularmente para textos generados por usuarios, a través de Twitter, en el idioma español y evaluar si las caracterización de la ironía varia conforme al idioma. A través de los modelos y las métricas presentados en el Capítulo 5, se evaluó el desempeño de los modelos utilizados, tanto en el idioma español como en inglés.

Con base en los resultados obtenidos en el Capítulo 6, en este Capítulo hacemos un compendio de los principales hallazgos de este trabajo y se enlistan algunos futuros trabajos.

7.1 Conclusiones

Lo primero que se puede reconocer es que el modelo de características basado únicamente en cómo escriben los usuarios, es decir, el modelo léxico, no funcionó muy bien. Nuestro supuesto es que debido a que los textos son generados por usuarios y en una red social, éstos suelen contener más errores de ortografía y gramatical que en textos formales. Lo que dificulta la extracción de características, pues palabras como “hola”, “ola” y “hoooooola” no se vinculan todas como un mismo concepto. Asimismo, características basadas en la gramática, como las variables POS tagging, se ven particularmente afectadas por la falta de acentos, puntos y por los errores de ortografía.

Con el experimento del Corpus No Balanceado, pudimos observar que el modelo léxico propuesto es el más sensible a la proporción de textos irónicos. Lo cual creemos se debe a que el uso de recursos como emoticones o signos de puntuación se utilizan en las redes sociales para dar entonación, expresar emociones o separar personas en un diálogo. Estas características no son únicas de la ironía, sino más bien del tipo de fuente de información, de manera que cuando incorporamos más variedad de tuits, estas características se hicieron presentes en textos que no eran irónicos, lo que disminuyó el poder predictivo del modelo.

Las características semánticas mejoran los resultados, pero no logran explicar del todo el fenómeno. Al igual que sucede en el enfoque basado en características léxicas, el tener palabras mal escritas genera algo de ruido en la identificación de “tokens” que se refieren a lo mismo, y aunque

semánticamente las representaciones vectoriales de palabras como “hola”, “ola” y “hooooola” se encuentren cercanas en el espacio de características, la representación no es exactamente la misma. Adicionalmente, debido a que en las redes sociales existe un cierto código de expresión, los vectores semánticos entrenados en Corpus de textos de Wikipedia no son tan eficientes, pues no logran capturar la semántica particular de las redes sociales.

Desde el enfoque semántico utilizando el método de doc2vec para la extracción de características fue bastante consistente entre idiomas, el desempeño fue más o menos el mismo. Mientras que el desempeño disminuyó un poco al utilizar el Corpus No balanceado en comparación con el Corpus Balanceado, aunque la disminución del poder predictivo no fue tan marcado como en el caso de las características léxicas.

Por su parte, en el enfoque basado en la representación semántica obtenida con ELMo, los resultados mejoraron mucho en los textos en español. Pensamos que esto debe a que en ELMo una palabra puede estar asignado a más de un vector; de manera que la palabra “ola” puede tener un vector cuando está rodeado de palabras como “mar”, “sal” o “playa” y tener otro para cuando está rodeado de palabras como “buenos” y “días”. Esta misma característica, puede ser de utilidad para asignar dos vectores alejados entre si para palabras como “bueno” dependiendo de si las palabras a su alrededor son positivas o negativas. En cuanto a cómo cambiaron los resultados cuando utilizamos un corpus en español o uno en inglés, detectamos que el rendimiento de todos los clasificadores mejoró mucho más en el idioma español que en inglés. Esto debido quizá a que la estructura gramatical del español es más compleja que en inglés, por ejemplo, en las palabras “perdida” y “pérdida” el uso de un acento cambia el significado semántico de la palabra, y los vectores de ELMo asignarán dos vectores semánticos diferentes aún con la falta de acento. Observamos también que el problema de detección de ironía utilizando ELMo fue el único en el que claramente se observó que los algoritmos lineales de clasificación eran mejor que los árboles de decisión.

En cuanto al experimento de juntar las características léxicas con las semánticas, detectamos que no se puede generalizar si esto produce una mejora en los modelos, ya que depende del tipo de características que analicemos (si son las de doc2vec o ELMo), si el corpus es balanceado o no y del tipo de clasificador que se considere. Además, en los casos donde se observó una mejoría, esta mejoría no fue muy alta.

Por último, el mejor resultado en cada modelo fue el que se obtuvo en el baseline de microTC, para cualquier idioma y sin importar la proporción de datos irónicos en la muestra. Es importante remarcar que los resultados de microTC son tan buenos como los que se han obtenido en otros trabajos utilizando un enfoque de aprendizaje profundo, lo cual nos habla de la potencia que tiene aunque no es un modelo de detección de ironía per se. Los buenos resultados de microTC

indican que una buena representación léxica puede dar resultados competitivos con respecto a modelos semánticos o modelos complejos de aprendizaje profundo.

7.2 Trabajos futuros

Como trabajo futuro, creemos que es posible mejorar los resultados realizando alguna o varias de las siguientes acciones:

- Entrenar los vectores de ELMo en un corpus de tuits en español. Con esto, en la representación semántica de las palabras se plasmaría no solo el significado semántico sino el lenguaje propio de las redes sociales.

- Utilizar representaciones sobre q-gramas. Para mitigar el ruido que genera que las palabras puedan estar mal escritas o con repetición de algunas letras, se plantea el uso de estructuras más pequeñas que las palabras, es decir los q-gramas. Con ello, se puede identificar que "hola", "ola" y "hooooola" y comparten el q-grama "ola".

- Utilizar una arquitectura de aprendizaje profundo. Los trabajos más recientes del estado del arte, no solo en la detección de ironía sino en muchos problemas de Procesamiento del Lenguaje Natural han sido abordados mediante modelos de lenguaje (redes recurrentes, convolucionales), consiguiendo mejorar significativamente el desempeño de los modelos anteriores. Es posible partir de alguna representación semántica de los documentos de las aquí presentadas como insumo para abordar el problema desde este enfoque.

Bibliografía

Amit, Yali y Donald Geman (1997). «Shape quantization and recognition with randomized trees». En: *Neural computation* 9.7, págs. 1545-1588 (zitiert auf Seite 39).

Barbieri, Francesco, Francesco Ronzano y Horacio Saggion (2014). «Italian irony detection in twitter: a first approach». En: *The First Italian Conference on Computational Linguistics CLiC-it*, pág. 28 (zitiert auf den Seiten 12, 13, 15).

Baziotis, Christos, Nikos Athanasiou, Pinelopi Papalampidi et al. (2018). «Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns». En: *arXiv preprint arXiv:1804.06659* (zitiert auf Seite 45).

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent y Christian Jauvin (2003). «A neural probabilistic language model». En: *Journal of machine learning research* 3.Feb, págs. 1137-1155 (zitiert auf den Seiten 31-34).

Breiman, Leo (2001). «Random forests». En: *Machine learning* 45.1, págs. 5-32 (zitiert auf Seite 39).

Carvalho, Paula, Luıs Sarmento, Mário J Silva y Eugénio De Oliveira (2009). «Clues for detecting irony in user-generated contents: oh...!! it's so easy;-». En: *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*. ACM, págs. 53-56 (zitiert auf Seite 13).

Charalampakis, Basilis, Dimitris Spathis, Elias Kouslis y Katia Kermanidis (2016). «A comparison between semi-supervised and supervised text mining techniques on detecting irony in greek political tweets». En: *Engineering Applications of Artificial Intelligence* 51, págs. 50-57 (zitiert auf Seite 12).

Che, Wanxiang, Yijia Liu, Yuxuan Wang, Bo Zheng y Ting Liu (2018). «Towards Better UD Parsing: Deep Contextualized Word Embeddings, Ensemble, and Treebank Concatenation». En: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, págs. 55-64. URL: http://www.aclweb.org/anthology/K18-2005 (zitiert auf Seite 36).

Cortes, Corinna y Vladimir Vapnik (1995). «Support-vector networks». En: *Machine learning* 20.3, págs. 273-297 (zitiert auf Seite 37).

Dietterich, Thomas G (2000). «An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization». En: *Machine learning* 40.2, págs. 139-157 (zitiert auf Seite 39).

Feurer, Matthias, Aaron Klein, Katharina Eggensperger et al. (2015). «Efficient and Robust Automated Machine Learning». En: *Advances in Neural Information Processing Systems 28*. Ed. por C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama y R. Garnett. Curran Associates, Inc., págs. 2962-2970. URL: `http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf` (zitiert auf Seite 40).

Forslid, Erik y Niklas Wikén (2015). *Automatic irony-and sarcasm detection in Social media* (zitiert auf Seite 11).

Freund, Yoav y Robert E Schapire (1997). «A decision-theoretic generalization of on-line learning and an application to boosting». En: *Journal of computer and system sciences* 55.1, págs. 119-139 (zitiert auf Seite 39).

Ho, Tin Kam (1995). «Random decision forests». En: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE, págs. 278-282 (zitiert auf Seite 39).

James, Gareth, Daniela Witten, Trevor Hastie y Robert Tibshirani (2013). *An introduction to statistical learning*. Vol. 112. Springer.

Jasso, Gabriela e Ivan Meza (2016). «Character and Word Baselines for Irony Detection in Spanish Short Texts». En: *Procesamiento del Lenguaje Natural* 56, págs. 41-48 (zitiert auf den Seiten 12, 14, 15, 18, 43, 48, 50, 51).

Joshi, Aditya, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya y Mark Carman (2016). «Are word embedding-based features useful for sarcasm detection?» En: *arXiv preprint arXiv:1610.00883* (zitiert auf Seite 14).

Jurafsky, Dan (2000). *Speech & language processing*. Pearson Education India (zitiert auf Seite 5).

Karoui, Jihen, Farah Banamara Zitoune y Veronique Moriceau (2017). «Soukhria: Towards an irony detection system for arabic in social media». En: *Procedia Computer Science* 117, págs. 161-168 (zitiert auf den Seiten 13, 15).

Le, Quoc y Tomas Mikolov (2014). «Distributed representations of sentences and documents». En: *International conference on machine learning*, págs. 1188-1196 (zitiert auf den Seiten 33, 34).

Mikolov, Tomas, Kai Chen, Greg Corrado y Jeffrey Dean (2013). «Efficient estimation of word representations in vector space». En: *arXiv preprint arXiv:1301.3781* (zitiert auf Seite 33).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado y Jeff Dean (2013). «Distributed representations of words and phrases and their compositionality». En: *Advances in neural information processing systems*, págs. 3111-3119 (zitiert auf den Seiten 33, 34).

Pedregosa, F., G. Varoquaux, A. Gramfort et al. (2011). «Scikit-learn: Machine Learning in Python». En: *Journal of Machine Learning Research* 12, págs. 2825-2830.

Peters, Matthew E, Mark Neumann, Mohit Iyyer et al. (2018). «Deep contextualized word representations». En: *arXiv preprint arXiv:1802.05365* (zitiert auf den Seiten 33, 35).

Ptáček, Tomáš, Ivan Habernal y Jun Hong (2014). «Sarcasm detection on czech and english twitter». En: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, págs. 213-223 (zitiert auf Seite 12).

Reyes, Antonio, Paolo Rosso y Tony Veale (2013). «A multidimensional approach for detecting irony in twitter». En: *Language resources and evaluation* 47.1, págs. 239-268 (zitiert auf den Seiten 12, 15).

Tellez, Eric S, Sabino Miranda-Jiménez, Mario Graff et al. (2017). «A simple approach to multilingual polarity classification in twitter». En: *Pattern Recognition Letters* 94, págs. 68-74 (zitiert auf Seite 27).

Tellez, Eric S., Daniela Moctezuma, Sabino Miranda-Jiménez y Mario Graff (2018). «An automated text categorization framework based on hyperparameter optimization». En: *Knowledge-Based Systems* 149, págs. 110-123. URL: `https://www.sciencedirect.com/science/article/pii/S0950705118301217` (zitiert auf Seite 23).

Van Hee, Cynthia, Els Lefever y Véronique Hoste (2018). «Semeval-2018 task 3: Irony detection in english tweets». En: *Proceedings of The 12th International Workshop on Semantic Evaluation*, págs. 39-50 (zitiert auf den Seiten 17, 45).