Universidad Nacional Autónoma de México

Facultad de Ingeniería

Computación

Identificación de ironía en textos cortos

T E S I S

que para obtener el grado de:

Ingeniero en Computación

presenta:

Gabriela Jasso López

director de tesis:

Iván Vladimir Meza Ruiz

Distrito Federal Abril, 2016

Ciudad Universitaria, Cd. Mx.

Identificación de ironía en textos cortos

por

Gabriela Jasso López

Tesis presentada para obtener el grado de

Ingeniero en Computación

en la

Facultad de Ingeniería

Universidad Nacional Autónoma de México

Distrito Federal. Abril, 2016

Dedicatoria
A mis padres y a todos los que me han apoyado

Índice general

Capítulo 1

Introducción

En las últimas décadas, el amplio uso de las computadoras y telecomunicaciones han permitido que el mundo tenga fácil acceso a internet. Esto provoca también el establecimiento de varios sitios dedicados a ofrecer una plataforma para que el usuario pueda expresarse e interactuar con otros usuarios, entre ellos el cúmulo de sitios que conocemos como redes sociales. Entre las más populares, se encuentran Facebook (968 millones de usuarios[1]) y Twitter (316 millones de usuarios[2]). En ellas, los usuarios escriben acerca de su vida, opiniones y sentimientos; acumulando millones de publicaciones al día, en diversos idiomas, públicamente accesibles. Los datos extraídos de ambos sitios son valiosos para el análisis mercadológico del usuario, ya que son testimonios de primera mano acerca de una variedad de temas de interés. Twitter y Facebook, por ejemplo, ofrecen la posibilidad de publicitar una marca o personalidad en sus sitios, adecuándose al gusto del usuario; mismo que averiguan mediante sus publicaciones.

Existen diversas herramientas computacionales cuya utilidad se basa en el análisis y manipulación de textos, tanto comercial como académicamente. Por ejemplo, TextCompactor (Edyburn, 2015) es una herramienta que resume textos automáticamente, cuya ejecución está basada en principios académicos (Lloret y Palomar, 2012). Sentiment140 (Go, Bhayani, y Huang, 2015) es un sitio que analiza la polaridad del sentimiento en una publicación de Twitter(“positivo” o “negativo”) (Zhu, Kiritchenko, y Mohammad, 2014). No obstante, aún no hay herramientas masivamente utilizadas que identifiquen el tono de una oración. El problema se ha analizado

[1] http://newsroom.fb.com/company-info/
[2] https://about.twitter.com/company

recientemente mediante temas como el humor y la ironía, y con especial énfasis en los textos de redes sociales, ya que son generados rápidamente y son de fácil obtención. Sin embargo, la detección del tono es una tarea sumamente difícil, principalmente porque la respuesta es subjetiva por naturaleza, ya que depende del contexto del interlocutor. Esto se evidencia con una búsqueda ejemplo en Twitter de la etiqueta *ironía*. El mensaje de la figura 1-1 es irónico de una forma que es relativamente sencilla de entender. El autor expresa que el día fue muy productivo, pero al agregar la etiqueta de ironía está indicando que fue todo lo contario: Sin

Figura 1-1: Ejemplo de etiqueta *ironía* en Twitter para el que no se necesita contexto (Twitter id: 605030329368358912)

embargo, se requiere saber un poco acerca del contexto del siguiente para entenderlo, como se puede observar en el *tweet* en la figura 1-2. El tema del que se expresa la ironía puede ser desconocido para algunos, en cuyo caso no entenderían que es un comentario irónico si no estuviese la etiqueta:

Figura 1-2: Ejemplo de etiqueta *ironía* en Twitter para el que se necesita contexto (Twitter id:605116788750356481)

Además de las ambigüedades descritos anteriormente, los textos escritos por usuarios de redes sociales tienen la particularidad de estar escritos en un lenguaje poco formal, con faltas de ortografía intencionales y no intencionales, modismos, emoticones, puntuación mal utilizada, entre otras características. A pesar de todos estos contratiempos, se han logrado varios estudios hacia la identificación de la ironía, con resultados prometedores.

1.1. Objetivo

El objetivo de esta investigación es identificar ironía en un texto corto en español. Se trata este problema como uno de clasificación, donde un clasificador es entrenado con ejemplos conocidos y posteriormente, con ejemplos desconocidos se prueba el desempeño de la clasificación.

1.2. Motivación

Con base en lo anterior, esta tesis busca crear un sistema que distinga entre un texto irónico y uno no irónico. El estado del arte tiene resultados favorables en inglés, con una variedad de métodos (explicados en el segundo capítulo).

Además de contribuir académicamente, realizar y afinar un sistema así podría ser ventajoso a varios campos. Un beneficio sería facilitar el análisis de opinión de los textos de usuarios o consumidores, ya sea en forma de reseñas de consumidor o comentarios personales acerca de la marca, producto u organización. Dicho análisis es más sencillo si se eliminan previamente los comentarios burlones y la crítica no constructiva.

Además, un sistema de detección de ironía permite analizar a los detractores de un producto, organización o usuario filtrando los comentarios irónicos del resto. Se puede observar la naturaleza de las burlas, qué porcentaje de la base de usuarios/consumidores habla irónicamente del producto u organización. También se podría detectar a los usuarios que abusan verbalmente de otros en redes sociales por medio del filtrado a los usuarios con mayor cantidad de comentarios irónicos dirigidos a otros, entre otras tareas posibles asociadas al perfilado de usuarios (Rangel y Rosso, 2013).

Finalmente, un sistema de detección de ironía puede ayudar a mejorar la generación de texto irónico. Esto le daría más profundidad al contenido de los sistemas de generación de lenguaje

natural, haciéndolo más parecido al lenguaje humano.

1.3. Consideraciones

En esta sección se revisan algunas consideraciones acerca de los conceptos de *Twitter* e *ironía* que el lector puede requerir para tener un mejor entendimiento del resto de la tesis.

1.3.1. Twitter

Twitter[3] es una plataforma de microbloggeo lanzada en 2006. Las publicaciones en Twitter no pueden superar los 140 caracteres. Cada publicación de Twitter se denomina *tweet*.

Los usuarios se identifican con un nombre de usuario único, mediante el cual se pueden comunicar en las publicaciones, lo cual es conocido como mencionar o mención. Para mencionar a un usuario basta con escribir su nombre, precedido de una arroba. Por ejemplo, *Hola @jack* o *@jack Hola*. Una mención envía una notificación al usuario mencionado, la cual puede contestar y que resulta en una conversación. Un ejemplo de mención se puede localizar en la figura 1-4.

Existe la posibilidad de embeber imágenes y vídeo en un *tweet*. El hipervínculo al vídeo o imagen embebido/a es parte del contenido (texto) del *tweet*. Es decir, los caracteres que forman la dirección del hipervínculo se restan al máximo de 140. Lo mismo ocurre si se incluyen vínculos en el *tweet*. Por defecto, el sitio acorta los vínculos, para permitir a los usuarios incluir vínculos grandes sin perder muchos caracteres. Un ejemplo de vínculos en Twitter se puede ver en la figura 1-5.

[3] http://twitter.com

Figura 1-3: Tweet ejemplo (Twitter id: 649362068857036800)

Figura 1-5: Hipervínculos en Twitter

Figura 1-4: Menciones en Twitter. En este caso el usuario mencionado es **nochedemuseos**

Adicionalmente, cualquier tweet puede tener cualquier número de etiquetas (o *hashtags* en inglés) que estén dentro de la marca de los 140 caracteres. Etiquetar permite que el tweet sea rastreado cuando alguien busque dicha etiqueta. Sin embargo, el uso de las etiquetas en Twitter es muy amplio; no siempre se limitan a describir el texto que las acompaña. Las palabras que son *hashtags* deben ser precedidas por el caracter #. Algunos ejemplos de *hashtags* se pueden encontrar en la figura 1-6.

Figura 1-6: Ejemplos de etiquetas en Twitter.

1.4. Ironía

La Real Academia Española define ironía como[4]:

1. Burla fina y disimulada.

2. Tono burlón con que se dice.

3. Figura retórica que consiste en dar a entender lo contrario de lo que se dice.

Una definición popularmente aceptada es la del retórico Quintiliano: ironía es decir lo opuesto a lo que se quiere decir. Como menciona Peña (2012), esta definición ha sido refutada por varios, entre ellos se extrae esta cita:

> La ironía no expresa lo contrario de lo que se piensa, sino algo mucho menos preciso, una gama de significación que, marcando un fuerte contraste, se extiende desde lo contradictorio a lo contrario, algo que supone la negación del sentido literal, pero no exactamente lo contrario de este. (Álvarez, 1986)

De acuerdo con Zavala (1992), la ironía es una yuxtaposición de perspectivas opuestas en un enunciado. "La contradicción semántica no existe en el enunciado (como en la ambigüedad o la paradoja) ni entre lo que se sabe y lo que se dice (como en la mentira o el suspenso), sino entre la proposición y su referente", agrega.

Muchos trabajan bajo el concepto de varios tipos de ironías. La ironía situacional, trágica, cómica, dramática, entre varios. Por otro lado, Haverkate (1990) propone sólo tres tipos: ironía verbal, dramática y del sino. Este documento se enfoca en la ironía verbal, que de acuerdo con Haverkate es la que basa su contradicción en una representación de carácter lingüístico.

García Barrientos define la ironía o antífrasis, como una *expresión en tono de burla de una significación contraria (o diferente) a la del enunciado que se pone de manifiesto por el contexto o la pronunciación, el gesto, etc* (Barrientos, 1998). Un ejemplo de ello es la frase *"¡Qué hermoso día!"*, para expresar que el día no es hermoso. Barrientos define al sarcasmo y asteísmo como subclases de ironía: *Sarcasmo es la clase de ironía que se caracteriza por la intención cruel, hostil o maliciosa que expresa* (donde un ejemplo es *"Él siempre tan trabajador"*, denotando

[4] http://buscon.rae.es/drae/srv/search?val=iron%EDa

que el individuo no trabaja), mientras que asteísmo *es la expresión de una alabanza en forma de represión, o viceversa* (un ejemplo sería decirle a un viajero *"te falta mundo"*). Adicionalmente, Monlau (1842) menciona el carientismo, dyasirmo y mimesis como variantes de ironía.

Existen muchas otras definiciones; sin embargo, todas coinciden en que la ironía puede implicar una burla (bajo alguna de las figuras retóricas relacionadas), y también puede dar a entender lo opuesto a lo que expone. Se puede observar que cuando se habla de sarcasmo, también se habla de ironía. Ironía es la forma más general de sarcasmo.

1.4.1. Ironía en Twitter

En este trabajo, se busca ironía por medio del *hashtag* **#ironía**. Un ligero análisis a los *tweets* con esa etiqueta muestra que ironía a veces se confunde con sarcasmo, dada la definición anterior. Una búsqueda al *hashtag* **#sarcasmo** arroja resultados muy similares al de ironía, en algunos casos que encajan mejor en la definición de ironía que en la de sarcasmo. Es decir, para el habla contidiana de un usuario de Twitter no hay gran diferencia entre ambos términos. Por ejemplo, éstos son algunos *tweets* etiquetados con el *hashtag* o etiqueta **ironía**:

Figura 1-7: Tweets etiquetados con ironía (Twitter ids: 604990620407607298, 605853826847940609, 604810721424252928, 604796926329004033)

Éstos son algunos *tweets* del *hashtag* **sarcasmo**:

Figura 1-8: Tweets etiquetados como sarcamo (Twitter ids: 605155462523154432, 602522964131762176, 615212712948330496, 615371369572868096)

No existe una diferencia discernible en ambos. Hay algunos casos que, de ser juzgados con alguna de las definiciones anteriormente descritas, están mal etiquetados.

Éstos son algunos *tweets* sin etiquetas **sarcasmo** e **ironía**:

Figura 1-9: Tweets no etiquetados como ironía ni sarcasmo (Twitter ids: 605499566297858048, 605234449362481154, 605233709336260609, 605684688724873216)

Existen varios casos donde los *tweets* sin etiquetas **sarcasmo** e **ironía** rayan en alguna de éstas, pero a simple vista se observa que no ocurre en la mayoría de los casos. En contraste, ironía y sarcasmo a veces se utilizan indistintamente, posiblemente debido a que están relacionados y el vocabulario popular tiende a agruparlos bajo el mismo concepto.

Resalta de inmediato que los usuarios de la plataforma Twitter no necesariamente escriben "sarcasmo" o "ironía" cuando el mensaje concuerda estrictamente con una definición formal. Además, por lo observado en la sección anterior acerca de los estudios acerca de la ironía, no existe una única definición de ironía (o de sarcasmo) que sea el estándar de todos los ámbitos. Es por ello que en este trabajo se considera un concepto de ironía que también envuelve al sarcasmo. De esta forma se consideran más ejemplos (*tweets*) y se representa la holgura que tienen los usuarios al citar alguno de los dos conceptos, lo cual refleja de forma más certera el concepto de ironía de Twitter en general. No se consideran otros fenómenos relacionados a la ironía como la sátira o la crítica ya que su uso es menos frecuente entre la mayoría de los usuarios.

1.5. Estructura de la tesis

Este documento se organizará de la siguiente forma: el segundo capítulo relata los esfuerzos anteriores y el estado del arte en la detección automática de ironía y/o sarcasmo. El tercer capítulo trata del sistema de clasificación -la forma de introducir los documentos a un clasificador, el funcionamiento del mismo y la arquitectura del sistema propuesto. El cuarto capítulo explica la motivación detrás del corpus utilizado y el proceso de extracción, etiquetado, preprocesado y análisis del mismo. El capítulo quinto documenta el experimento: las métricas usadas para evaluarlo, las condiciones del mismo, los resultados y su análisis. Finalmente, el capítulo seis recopila las conclusiones del experimento y de la tesis en general.

Capítulo 2

Antecedentes

En este capítulo, se listan y comentan los esfuerzos anteriores y recientes en relación a la detección automática de ironía y/o sarcasmo. Además, se presenta una comparación entre los mismos y entre el trabajo de esta tesis.

2.1. Introdución

La ironía ha sido estudiada ampliamente en varios campos, como la literatura y la filosofía. Sin embargo, no es hasta recientemente que se trató de entender por medio de la computación. Entre los esfuerzos más tempranos se encuentra el de Utsumi (1996), que idea una serie de alusiones irónicas por medio de las cuales crea un modelo para representar el grado de ironía en una oración. Sin embargo, este modelo era muy abstracto para representar ironía más allá de los confines de una interacción de locutor-interlocutor. En una idea similar, Veale y Hao (2008) crean un sistema de generación de metáforas creativas, en el que analizan metáforas usadas para describir opiniones irónicas.

2.2. Corpus usados

La facilidad de obtener información de primera mano desde redes sociales tuvo un gran impacto sobre el procesamiento de textos computacional. Diversos investigadores usan corpus provenientes de redes sociales, como publicaciones de Twitter (Bosco, Patti, y Bolioli, 2013;

Davidov, Tsur, y Rappoport, 2010; González-Ibáñez, Muresan, y Wacholder, 2011; Liebrecht, Kunneman, y van den Bosch, 2013; Ptácek, Habernal, y Hong, 2014; Reyes, Rosso, y Veale, 2013; Vanin, Freitas, Vieira, y Bochernitsan, 2013), Yahoo (Tang y Chen, 2014), reseñas de productos del sitio AmazonTsur y Davidov (2010), comentarios en sitios de noticias(Carvalho, Sarmento, Silva, y de Oliveira, 2009), entre otros. Una de las ventajas que suponen estos corpus, muchas veces aprovechada y considerada en sus publicaciones, es que están previamente etiquetados con una calificación (en caso de las reseñas) o tema (hashtags en Twitter). Un resumen de los corpus usados se puede ver en la tabla 2-1.

2.2.1. Enfoques de estudio

Los enfoques de estos estudios se concentran en la identificación de ironía (como Reyes y Carvalho), sarcasmo (como Davidov y Liebrecht), o el grado del mismo (Utsumi y Ghosh y cols. (2015)), así como en el análisis de los elementos que identifican estos fenómenos. También se observa que se aislan los textos de un cierto idioma, dependiendo del autor. La mayoría de los autores anteriormente mencionados aborda el problema en inglés. Carvalho et al., Vanin et al., Liebrecht et al., Tang et al., Bosco et al. y Ptácek et al. usan textos (*tweets*) en portugués, neerlandés, chino, italiano y checo, respectivamente.

2.3. Clases para la clasificación

La mayoría de estos trabajos tratan la identificación de ironía como un problema de clasificación, donde existe la clase irónica y una o más contrastantes. Por ejemplo, en el trabajo de González-Ibáñez et al. se consideran dos clases además de la sarcástica (sarcástica, positiva y negativa), y se etiqueta por medio de *hashtags*: #sarcasmo, etiquetas con polaridad positiva (como *#happy*, feliz en inglés), y etiquetas con polaridad negativa (como *#sad*, triste en inglés). Por otro lado, en el trabajo de Reyes et al., se consideran tres clases además de la irónica, identificadas por los *hashtags*: #irony, #politics, #humor y #education, o ironía, política, humor y educación. En este caso política, humor y educación fungen como las clases contrastantes a la irónica. Liebrecht et al. usa sólo dos clases: sarcástico (denotado por #sarcasme en neerlandés) y trasfondo (todo lo que no esté etiquetado explícitamente como sarcástico). En los casos an-

teriores, se asume el etiquetado de los usuarios, por medio de *hashtags*. Existe problema de que pueda estar mal etiquetado o puedan existir textos repetidos. Algunos autores ofrecen un análisis del corpus (en su mayoría superficial) para tratar de justificar esta decisión, pero no abandonan el etiquetado. Davidov et al. y Bosco et al., entre otros, ante este problema cambian el etiquetado a uno manual. En el caso de Carvalho et al. y de los trabajos dedicados a construir un corpus y analizar los elementos que identifican la ironía, el etiquetado suele hacerse manualmente. Carvalho et al. plantean cuatro clases: irónica, no irónica, indecisa y ambigua.

2.4. Vectorización

Para realizar la tarea de clasificación, se vectoriza el corpus de acuerdo con alguna serie de medidas o características del texto. Los autores difieren en sus criterios, usando diferentes características para representar sus documentos. Muchas de ellas son ampliamente usadas y otras fueron creadas por el autor para esta tarea en específico. En términos generales, se pueden dividir en características de estilo, de diccionarios gramaticales o lingüísticos y de diccionarios emocionales o psicológicas. Un resumen de estas características se puede encontrar en la tabla 2-2.

2.4.1. Características de estilo

Las características de estilo se refieren a los símbolos (letras, mayor uso de mayúsculas o de minúsculas, frecuencia de ciertas palabras sobre otras, etc) y puntuación (puntos, signos de exclamación e interrogación, acentos, comas, punto y comas, etc) que se usan en los textos, que pueden estar personalizados para representar mejor un idioma. Un ejemplo de características de estilo se observa en el trabajo de Carvalho et al., en que se usa la puntuación, citas (entrecomillado) y signos de interrogación entre las características que representan un texto. También representa con "expresiones de risa" fijas, o formas en que la gente expresa que se ríe o divierte en redes sociales, como emoticones (*:)*, por ejemplo) y acrónimos (*lol*, por ejemplo). Por otro lado, Reyes et.al usan n-gramas (secuencias de caracteres o palabras), skip-gramas (palabras separadas entre si por una cierta brecha de otras palabras), y signos de puntuación, entre otros. Liebrecht et al., por su parte, usaron unigramas, bigramas y trigramas de palabra

para representar su texto. Es decir, una palabra, una secuencia de dos palabras, y una de tres palabras respectivamente. Tungthamthiti, Kiyoaki, y Mohd (2014) también los usan.

2.4.2. Características lingüísticas y gramaticales

Las características de diccionarios gramaticales o lingüísticos se refieren a lo relacionado con la estructura lingüística/gramatical de una frase. Un ejemplo de ello es la característica *temporal compression* o compresión de tiempo en el documento de Reyes et al., en que describe la cantidad de adverbios temporales (como *súbitamente, ahora, abruptamente, de repente*, etc) en una frase, y debe reflejar si el evento relatado está ocurriendo en otra línea de tiempo o no. Similar a esto, está *temporal imbalance* en el mismo documento, que describe las diferencias en los tiempos verbales de los verbos que componen una oración. Varios otros trabajos, entre ellos el de Barbieri y Saggion (2014) y González-Ibáñez et al. usan *Part-Of-Speech Taggers* (parte del discurso, por sus siglas en inglés) para identificar la cantidad de elementos de cada parte del discurso, como son verbos, sustantivos, pronombres, etc. Diversos autores proponen sus propias características, bajo un análisis de su importancia en el discurso irónico. Algunos analizan su utilidad en la tarea de clasificación.

2.4.3. Características emocionales

Las características emocionales o psicológicas se refieren al contexto de las palabras de un texto, tomando en consideración la reacción que causan en el lector. Entre las características emocionales para describir texto se encuentra el análisis de polaridad en el *tweet*. Algunos reflejan cómo cambia la polaridad de palabra en palabra (Reyes y cols., 2013), otros las suman para obtener una polaridad final (Tungthamthiti y cols., 2014), mientras que algunos analizan el efecto de la negación sobre la ironía del mensaje (Maynard y Greenwood, 2014; Xu, Santus, Laszlo, y Huang, 2015). Para ayudar en esta tarea se usan regularmente diccionarios (SentiNet, Depeche Mood, American National Corpus entre otros) de sentimiento de palabras, asociando una escala de sentimientos (donde los extremos son positivo y negativo) a cada una de ellas. González-Ibáñez et al. usan una medida de distancia entre términos positivos y negativos de una frase. Reyes et al. usan una serie de "escenarios emocionales" extraídos de diccionarios anotados con las dimensiones activación, metáforas y agradabilidad (del inglés *activation, imagery* y

pleasantess) de cada palabra, además de skip-gramas de polaridad (cómo cambia la polaridad de entre dos palabras separadas entre sí por una cierta brecha de palabras). Estos últimos se basan en la intuición de que uno generalmente emplea términos positivos para comunicar un significado negativo cuando se usa ironía. Como este ejemplo hay varios de autores que proponen características propias dependiendo del problema que quieren atacar.

2.5. Clasificación

Salvo por los enfoques lingüísticos, la mayoría de los enfoques tratan la detección de ironía/ sarcasmo como un problema de clasificación, por lo que usan un algoritmo clasificador. La mayoría justifica su decisión basado en el enfoque que adoptan o en esfuerzos previos que tuvieron buenos resultados. Se le introducen al clasificador los textos vectorizados (de acuerdo con las características anteriores) etiquetados como irónico/sarcástico y las clases de comparación, de las cuales aprende patrones y asigna una etiqueta a un texto desconocido. Para ello, la mayoría introduce una igual cantidad de textos de cada clase, a lo que llaman una distribución balanceada. Reyes et al. propone una distribución adicional, más realista de 70 % de *tweets* no irónicos y 30 % de *tweets* irónicos. Liebrecht et al. usa una distribución balanceada para entrenar, y una muestra realista (donde los *tweets* irónicos no son ni el 10 % del total) para probar. La mayoría de los autores usa una muestra balanceada para probar

Los resultados más sobresalientes (en cifras) que abordan el problema de clasificación son los de Ptácek, con 93 % de puntaje F para el inglés, seguido de Reyes et al., Tsur et al. y Tungthamthiti et al., obteniendo resultados por arriba del 70 %. Adicionalmente, se han probado otros enfoques recientemente. Varios enfoques se observaron en SemEval 2015, en una tarea en que se debe definir un grado de ironía en lugar de una clasificación contundente (Ghosh y cols., 2015). Otro esfuerzo reciente que reporta resultados exitosos es el de Bamman y Smith (2015), que propone encontrar ironía a partir del contexto de un texto de Twitter como la conversación que antecede y sucede al *tweet*, características del autor, entre otros.

Autor	**Enfoque**	**Idioma**	**Etiquetado**	**Tipo**	**Cantidad**
Carvalho	ironía	portugués	manual	comentarios de usuario para un diario	250,000, del cual menos del 10 % es sarcástico
Davidov	sarcasmo	inglés	Amazon Mechanical Turk	reseñas de usuario (Amazon) y *tweets* (Twitter)	66,000 reseñas de usuario para extracción de patrones (Amazon) y 5.6 millones de *tweets* (Twitter), de entre los cuales se entrena y evalúa con conjuntos pequeños de menos de 2000 *tweets* sarcásticos
González-Ibáñez	sarcasmo	inglés	automático para la clase no sarcástica (hashtags), manual para la clase sarcástica	*tweets* (Twitter)	900 *tweets* por clase
Liebrecht	sarcasmo	neerlandés	automático (hashtags)	*tweets* (Twitter)	para entrenar usa 77,948 *tweets* sarcásticos, para probar usa otro conjunto de 135 sarcásticos vs. 3.3 millones de *tweets* no sarcásticos
Reyes	ironía	inglés	automático (hashtags)	*tweets* (Twitter)	10,000 *tweets* irónicos, y 10,000 para cada clase contrastante
Barbieri	ironía	inglés	automático (hashtags)	*tweets* (Twitter)	10,000 *tweets* irónicos, y 10,000 para cada clase contrastante
Tungthamthiti	sarcasmo	inglés	manual	*tweets* (Twitter)	25,000 *tweets* sarcásticos y 25,000 *tweets* normales
Bosco	ironía	italiano	manual	*tweets* (Twitter)	3,288 *tweets* irónicos
Varin	ironía	portugués	automático (patrones)	*tweets* (Twitter)	2,780 *tweets* registrados irónicos
Tang	ironía	chino	semi-automático (patrones)	publicaciones (Yahoo) y otros	2,825 publicaciones
Ptácek	sarcasmo	inglés	automático (hashtags)	*tweets* (Twitter)	130,000 *tweets* sarcásticos y 650,000 *tweets* normales
Este trabajo	ironía y sarcasmo	español	automático (hashtags)	*tweets* (Twitter)	14000 *tweets* irónicos para entrenar, 1050 para probar, y 600,000 *tweets* no irónicos para ambas

Tabla 2-1: Corpus varios

Autor	**Estilo**	**Lingüísticas**	**Emocionales**
Carvalho	puntuación, signos de interrogación, expresiones de gracia, terminaciones en diminutivos	palabras de interjección, determinantes de demostración, reconocimiento de entidades nombradas por diccionario	requiere presencia de adj./sust. positivos sin negativos alrededor
Davidov	signos de puntuación, patrones "sarcásticos" encontrados por el sistema hecho con reseñas de Amazon	ninguna	ninguna
González-Ibáñez	unigramas de palabra, emoticones, menciones de usuario	procesos lingüísticos (adjetivos, pronombres, etc.)	procesos psicológicos (positivo, negativo), WordNet afectivo
Liebrecht	unigramas, bigramas y trigramas de palabra, signos de exclamación	intensificadores (del neerlandés), palabras derivadas de *sarcasmo*	ninguna
Reyes	signos de puntuación, c-grams, skip-grams	diversos conteos de verbos, adverbios temporales, términos de oposición, campos semánticos	s-grams de polaridad, diccionarios de activación, imagen, agradabilidad
Barbieri	puntuación, longitud de palabra, emoticones, puntuación	cantidad de etiquetas del etiquetador POS, sinónimos	intensidad de palabras
Tungthamthiti	puntuación, unigramas, bigramas y trigramas de palabra, emoticones, jerga popular	coherencia gramatical de la oración	connotación de palabras con su polaridad, contradicciones en el sentimiento de las palabras de la oración
Ptácek	n-gramas de palabra y carácter, emoticones, puntuación, patrones	características, forma de POS, n-gramas de POS	ninguno
Este trabajo	unigramas, bigramas y trigramas de palabra, bigramas, trigramas y tetragramas de caracter	ninguna	ninguna

Tabla 2-2: Características al vectorizar

Autor	**Enfoque**	**Clases de clasificación**	**Método**
Carvalho	ironía	irónico, no irónico, dudoso, ambiguo	se usan 500 comentarios del total, se detectan patrones sobre la totalidad del corpus, los comentarios detectados se categorizan en las clases manualmente
Davidov	sarcasmo	escala del 1 al 5 donde 1-2 representan lo no sarcástico y 3-5 lo sarcástico	K-vecinos más cercanos para clasificar
González-Ibáñez	sarcasmo	positivo, negativo, sarcástico	regresión logística y *Support Vector Machines* con SMO
Liebrecht	sarcasmo	sarcástico y *background*, es decir, no etiquetado explícitamente como sarcástico	clasificador Winnow
Reyes	ironía	ironía, política, educación y humor	clasificación con Decision Trees y Naive Bayes
Barbieri	ironía	ironía, política, educación y humor	clasificación con Decision Trees
Tungthamthiti	sarcasmo	sarcástico y normal	dos clasificadores SVM entrenados con dos conjuntos de características distintos
Ptácek	sarcasmo	sarcástico y normal	MaxEnt y SVM
Este trabajo	ironía y sarcasmo	sarcástico/irónico y *background*, es decir, no etiquetado explícitamente como sarcástico/irónico	clasificación con *Random Forests* y *Support Vector Machines*

Tabla 2-3: Métodos de antecedentes

A pesar de que todos los autores reportan al final sólo sus mejores resultados y la combinación de características para vectorizar que los llevan ahí, algunos reportaron los resultados que obtienen cuando usan diferentes combinaciones de características. Cavlaho et al. reporta que sus características más efectivas fueron los signos de puntuación y las expresiones de risa/gracia, donde los patrones de conocimiento lingüístico resultaron poco efectivos para detectar ironía (en portugués). Reyes et al. reporta la utilidad de los factores de estilo (puntuación, n-gramas de caracter, skip-grams) como los más valiosos para su experimento, en el que muchas características emocionales no aportan mucho al experimento (en cifras).

Capítulo 3

Método

En este capítulo se desglosa el sistema de clasificación, incluyendo la forma de introducir los documentos a un clasificador, detalles de su funcionamiento y la arquitectura del sistema que esta tesis propone.

3.1. Introducción

Para identificar ironía y no ironía se requiere de los textos base, una forma de vectorizarlos y algún algoritmo clasificador. En este capítulo se hablará de la vectorización y los clasificadores por separado. En la sección de vectorización se describe la teoría, funcionamiento y algunos ejemplos de la herramienta *Word2vec* y los modelos en los que está basada. Para la clasificación, se proponen dos algoritmos para los distintos enfoques manejados: *Random Forests* (Bosques Aleatorios, del inglés) y *Support Vector Machines* (Máquinas de Soporte Vectorial, del inglés). Se aborda su funcionamiento, relación al experimento y resultados esperados.

3.2. Vectorización

De acuerdo con la sección de antecedentes, se observa que existen varias características candidatas a vectorizar. Para el alcance de este documento, se consideran dos grupos: n-gramas de palabra y n-gramas de caracter, ya que fueron factores constantemente efectivos en los estudios anteriores (Ptácek y cols., 2014; Reyes y cols., 2013).

3.2.1. N-gramas

Se proponen rangos de 1 a 3 para los n-gramas de palabra. Es un rango corto ya que el rango de los textos es corto. Para los n-gramas de caracter se propone un rango de 2 a 4, para explotar la corta longitud del *tweet*, y no pasar de la longitud de palabra promedio.

3.2.2. Tf-idf

Diversas medidas pueden caracterizar los n-gramas. Una medida propuesta es la de $tf-idf$, o frecuencia de término - frecuencia inversa de documento. Ésta refleja la relevancia de un término para un documento parte de un conjunto. Mientras que un conteo de términos refleja los términos más comunes, $tf-idf$ compensa los términos comunes con su frecuencia en la colección completa de documentos. La compensación permite diferenciar los términos más comunes y darles un peso bajo.

En la ecuación 3-1 se define la frecuencia de término, en relación al término en cuestión t y el documento al que pertenece d. La frecuencia de término es el resultado del cociente de la frecuencia de un término en un documento $f(t,d)$ y el término w con la frecuencia más alta en el documento d, $maxf(w,d)$.

$$tf(t,d) = \frac{f(t,d)}{max\left\{f(w,d) : w \in d\right\}} \tag{3-1}$$

La frecuencia inversa de documento, definida en la ecuación 3-2, es una medida de la información que provee el término t a lo largo del conjunto de todos los documentos D. Es el cociente escalado logarítmicamente del número de documentos totales $|D|$ y el número de documentos que contienen el término t.

$$idf(t,D) = \log \frac{|D|}{|\left\{d \in D : t \in d\right\}|} \tag{3-2}$$

Finalmente, $tf-idf$ se define en la ecuación 3-3, donde se multiplican los valores de tf e idf de un término t, en relación al documento al que pertenece d y el conjunto total de documentos D.

$$tf-idf(t,d,D) = tf(t,d) \times idf(t,D) \tag{3-3}$$

Un sondeo rápido a los *tweets* recolectados muestra que se repiten una gran mayoría de palabras muy comunes; $tf-idf$ debe puntuar más bajo éstas que palabras menos frecuentemente usadas y con más contenido. Se elige esta medida en espera de que ayude a filtrar términos innecesarios y haga más eficiente la descripción vectorial de cada *tweet*.

3.2.3. *Word2vec*

Word2vec y su aplicaciones por parte de Mikolov, Sutskever, Chen, Corrado, y Dean (2013) han atraído la atención de muchos en los últimos años. *Word2vec* es el nombre dado a una red neuronal de dos capas que, dada una entrada (un corpus) no etiquetada de entrenamiento, produce un vector para cada palabra en la misma, que codifica su información semántica o contextual. El propósito y utilidad de *Word2vec* es agrupar los vectores de palabras similares en un espacio vectorial, con lo que se puede medir la similitud semántica entre dos palabras, calculando la distancia entre sus vectores de palabra correspondientes.

Dados suficientes datos, *Word2vec* puede hacer suposiciones muy precisas (Goldberg y Levy, 2014) acerca del significado de una palabra basado en apariciones pasadas. Esas suposiciones se pueden usar para establecer la asociación de una palabra con otras, por medio de operaciones traslaciones lineales entre los vectores de dichas palabras. Por ejemplo, el resultado de un cálculo de vectores vec("Madrid") - vec("España") + vec("Francia") es más cercano a vec("Paris") que a cualquier otro vector de palabra.

Los vectores de palabra de *Word2vec* se pueden usar como características para varias tareas de procesamiento de lenguaje natural supervisadas, como clasificación de documentos, reconocimiento de entidades nombradas y análisis de sentimiento. La información semántica contenida en estos vectores los vuelve importantes para estas tareas.

Los modelos propuestos detrás de esta herramienta se describen en (Mikolov, Chen, Corrado, y Dean, 2013) y (Mikolov, Sutskever, y cols., 2013), así como algunas técnicas de optimización. En términos generales, *Word2vec* se basa en los modelos CBOW y Skip-gram.

Modelo de bolsa de palabras continuas (Continuous Bag-Of-Word Model, o CBOW)

En términos generales, CBOW es entrenado para predecir la palabra objetivo w_I a partir de las palabras contextuales que la rodean, w_j. Es decir, dado un conjunto de palabras, CBOW

predice la palabra actual. En el ejemplo que se observa en la figura 3-1, CBOW podría predecir palabras como “vuela” o “canta”.

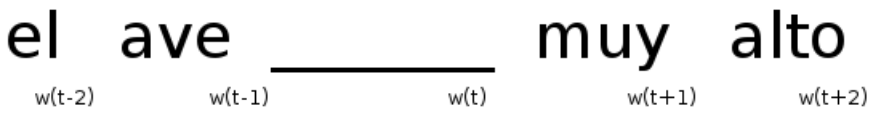

Figura 3-1: Ejemplo de contexto CBOW.

El objetivo de CBOW es maximizar $p(w_j|w_I)$ a través del conjunto de entrenamiento. Esta probabilidad resulta proporcional a la distancia entre los vectores actuales asignados a w_j y w_I. Este modelo se entrena un ejemplo a la vez, por lo que el objetivo en cada iteración es minimizar la distancia entre los vectores actuales para w_j y w_I, incrementando la probabilidad $p(w_j|w_I)$. Al repetir este proceso en el conjunto de entrenamiento se observa que los vectores para las palabras que regularmente co-ocurren tienden a estar más cerca entre sí, y este proceso gradualmente converge a un estado final de los vectores.

Este modelo fue propuesto por Mikolov et al. Se asume que sólo una palabra se considera por contexto; es decir, el modelo va a predecir una palabra objetivo dada una palabra contexto, similar a un modelo de bigramas.

En el diagrama a continuación se observa la red del modelo bajo la definición anterior. En esta configuración, el tamaño de vocabulario es V, y el tamaño de la capa oculta es N. Los nodos en las capas adyacentes están completamente conectados. El vector de entrada es un vector codificado con *one-hot*; es decir, para cada palabra de contexto de entrada, sólo un nodo de $\{x_1, ..., x_V\}$ será 1, y el resto de los nodos será 0.

Los pesos entre la capa de entrada y la capa de salida pueden ser representados por una matriz V x N **W**. Cada fila de **W** es la representación vectorial N-dimensional v_w de la palabra asociada de la capa de entrada. Dado un contexto (una palabra), asumiendo que $x_k = 1$ y

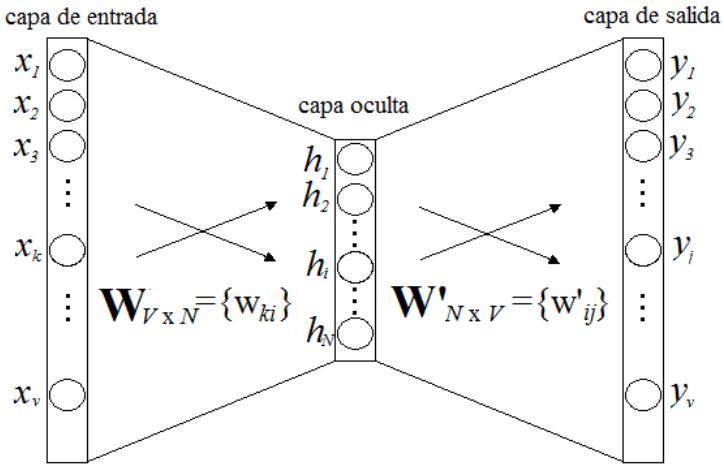

Figura 3-2: Modelo CBOW para *Word2vec* [basado en (Rong, 2014a)].

$x_{k'} = 0$ para $k' \neq k$, entonces

$$\mathbf{h} = \mathbf{x}^T \mathbf{W} = W_{k,\cdot} := \mathbf{v}_{w_I}, \tag{3-4}$$

que representa copiar la k-ésima fila de $\mathbf{W}$ a $\mathbf{h}$. $\mathbf{v}_{\mathbf{w_I}}$ es la representación vectorial de la palabra de entrada w_I. $\mathbf{v}_w$ es la representación de la palabra de entrada w_I.

Desde la capa de entrada a la capa de salida, existe una matriz de pesos diferente $\mathbf{W}' = \{w'_{ij}\}$, que es una matriz N x V. Usando estos pesos, se puede computar una puntuación u_j para cada palabra en el vocabulario

$$u_j = {\mathbf{v}'_{w_j}}^T \dot{\mathbf{h}}, \tag{3-5}$$

donde ${\mathbf{v}'}_{w_j}$ es la columna j-ésima de la matriz **W'**. Entonces se puede usar soft-max. un modelo de clasificación loglineal, para obtener la distribución de palabras posterior.

$$p(w_j|w_I) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{3-6}$$

Este modelo después se retropropaga para aprender las matrices de pesos W y W', se actualizan los pesos de salida y entrada de la capa oculta. Esto se puede generalizar para una o más palabras continuas.

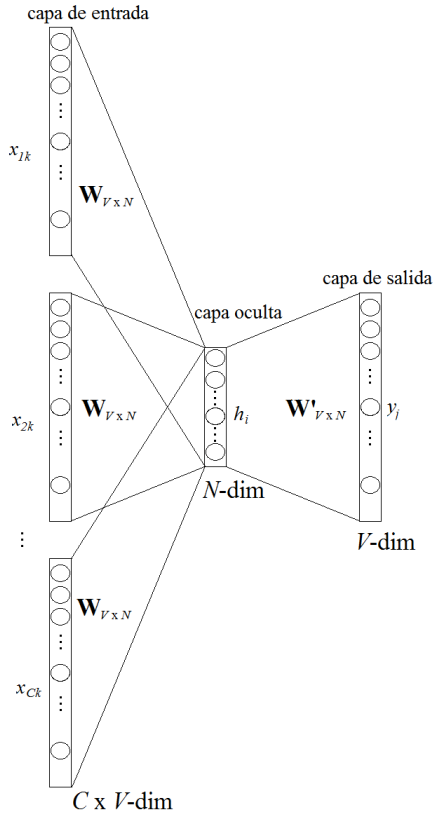

Figura 3-3: Modelo CBOW con un contexto multi-palabra para *Word2vec* [basado en Rong (2014b)].

Modelo de Skip-Gram

En términos generales, skip-gram es entrenado para predecir el contexto $w_{O,C}$ a partir de la palabra objetivo w_I. En un ejemplo similar al descrito en la figura 3-1, la figura 3-4 muestra la palabra objetivo, en la cual pueden corresponder varios contextos.

La entrada del modelo de *skip-gram* (salto-grama, del inglés) es una palabra w_I y la salida son las palabras en el contexto de w_I $\{w_{O1}, ..., w_O, C\}$ definidas por una ventana de palabras tamaño C (donde la ventana consiste de un rango). El vector de entrada es un vector codificado con *one-hot*; es decir, para cada palabra de contexto de entrada, sólo un nodo de $\{x_1, ..., x_V\}$ será 1, y el resto de los nodos será 0. El objetivo de skip-gram es maximizar $p(w_{O,C}|w_I)$ a través del conjunto de entrenamiento, de forma similar a CBOW, pero a la inversa.

Figura 3-4: Ejemplo de modelo Skip-gram.

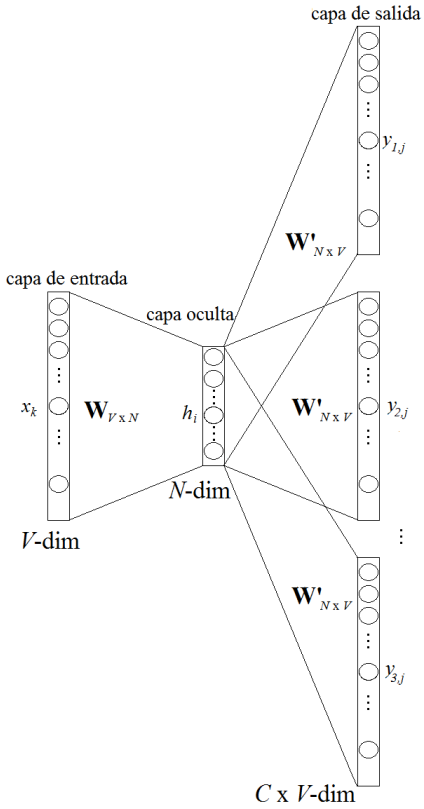

Figura 3-5: Modelo Skip-Gram para *Word2vec* [basado en (Rong, 2014c)].

En el modelo anterior, $\mathbf{x}$ representa el vector codificado en *one-hot* correspondiente a la palabra de entrada en la instancia de entrenamiento, mientras que $\{\mathbf{y_1}, ..., \mathbf{y_C}\}$ son los vectores

codificados en *one-hot* correspondientes a las palabras de salida en la instancia de entrenamiento.

La matriz $V \times N$ $\mathbf{W}$ es la matriz de pesos entre la capa de entrada y la capa oculta, cuya i-ésima fila representa los pesos correspondientes a la i-ésima palabra del vocabulario. La matriz de pesos $\mathbf{W}$ es lo que interesa saber ya que contiene las codificaciones de vectores de todas las palabras del vocabulario (una por fila). Cada vector de palabra de salida también tiene asociada una matriz de salida $N \times v$ $\mathbf{W'}$. Existe además una capa oculta constituida por N nodos. La entrada a una unidad en la capa oculta h_i es la suma pesada de todas sus entradas, por lo que en el vector de entrada $\mathbf{x}$ los elementos no-cero son los únicos que contribuyen a la capa oculta. Por lo tanto, para la entrada $\mathbf{x}$ con $x_k = 1$ y $x_{k'} = 0$ para toda $k' \neq k$, las salidas de la capa oculta serán equivalentes a la k-ésima fila de $\mathbf{W}$. Matemáticamente:

$$\mathbf{h} = \mathbf{x}^T \mathbf{W} = \mathbf{W}_{(k,.)} := \mathbf{v}_{w_I} \tag{3-7}$$

Las entradas para cada uno de los nodos de salida $C \times V$ se computa con la suma pesada de sus entradas. Por lo que la entrada al j-ésimo nodo de la c-ésima palabra de salida es

$$u_{c,j} = \mathbf{v'^{T}}_{w_j} \mathbf{h} \tag{3-8}$$

Sin embargo, también se observa que las capas de salida para cada palabra de salida comparten los mismos pesos, por lo que $u_{c,j} = u_j$. Se puede computar la salida del j-ésimo nodo de la c-ésima palabra de salida por medio de la función softmax, que produce una distribución multinomial

$$p(w_{c,j} = w_{O,c} | w_I) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{3-9}$$

es decir, este valor es la probabilidad de que la salida del j-ésimo nodo de la c-ésima palabra de salida sea igual que el valor actual del j-ésimo índice de la c-ésimo vector de salida (codificado en *one-hot*). Esto nos permite saber cómo se propagan las entradas a traves de la red para producir salidas. Se puede derivar la gradiente de error necesaria para usar el algoritmo de retropropagación para aprender **W** y **W'**.

Eficiencia computacional

Se usa soft-max jerárquico para eficientizar el cómputo de soft-max. Adicionalmente, se realiza muestreo negativo (Negative Sampling, en inglés) para eficientizar el soft-max jerárquicoRong (2014d).

3.3. Clasificación

3.3.1. *Random Forests*

Random Forest o bosque aleatorio es el nombre dado a un conjunto de árboles de decisión, cada uno capaz de producir una respuesta ante un conjunto de valores. Para un problema de clasificación, esta respuesta es una clase a la que se es miembro, lo cual asocia o clasifica un conjunto de valores predictores independientes con una de las categorías presentes en la variable dependiente. En problemas de clasificación, el conjunto de árboles simples votan por la clase más popular. En un problema de regresión, las respuestas se promedian para obtener un estimado de la variable dependiente. Usar conjuntos de árboles puede llevar a mejoras significativas en la exactitud de la predicción.

El bosque aleatorio inicia con un árbol de decisión. En éste, una entrada se ingresa en la parte superior y mientras viaja bajo el árbol, los datos se concentran en conjuntos más y más pequeños. Un árbol de decisión es un árbol en el que cada nodo interno (es decir, no hoja) está etiquetado con una característica de entrada. Los arcos que vienen de un nodo etiquetado con una característica son nombrados con cada uno de los valores posibles de la característica. Cada hoja del árbol es etiquetada con una clase o una distribución de probabilidad a través de las clases. La figura 3-6 muestra un ejemplo de árbol de decisión.

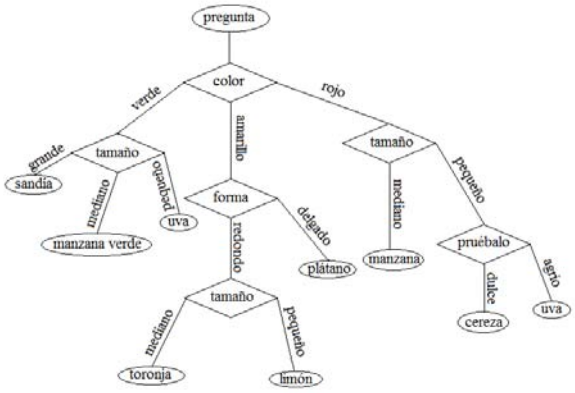

Figura 3-6: Ejemplo de árbol de decisión [basado en (Mukerjee, 2013)]

Un árbol puede ser aprendido separando el conjunto fuente en subconjuntos basados en el valor de algún atributo. El proceso se repite en cada subconjunto derivado hasta que todo el subconjunto en algún nodo tiene el mismo valor que la variable objetivo, o cuando partir el conjunto ya no agrega valor a las predicciones.

Los árboles de decisión son fáciles de calcular, pero no clasifican particularmente bien, por lo que se les llama clasificadores débiles. Sin embargo, se puede buscar construir un clasificador fuerte a partir de varios clasificadores débiles, llamados ensamble. Para ello se adoptan diversas estrategias:

Bootstrapping

Significa generar nuevos conjuntos del mismo tamaño por medio de un muestreo con reemplazo. Una muestra bootstrap debe tener aproximadamente el 62.3 % de las muestras originales.

Bagging

*B*ootstrap *Agg*regat*ing* o Bagging es una técnica para entrenar cada clasificador débil con un conjunto de datos diferente, generado por la estrategia de *bootstrapping*.

Por medio de las muestras generadas por *bagging* se crean árboles de decisión. Con éstos se gana diversidad al examinar diversos ejemplos en los datos. Por convención los árboles se sobreentrenan, ya que tienen menos datos y están hechos a su medida (sin embargo el bosque completo no se sobreentrena). El algoritmo para construir los árboles se describe a continuación:

- Para $b = 1$ a B

- Sacar una muestra *bootstrap* Z de tamaño N de los datos de entrenamiento.
- Construir un árbol *random-forest* T_b con los datos *bootstrap*, repitiendo los pasos siguientes para cada nodo terminal del árbol, hasta que se logra un tamaño de nodo mínimo previamente designado
  - Seleccionar m variables al azar de entre las p variables
  - Elegir el mejor punto de corte o variable entre las m variables
  - Partir el nodo en dos nodos hijos

- Mostrar ensamble de árboles $T_{b_1}^B$

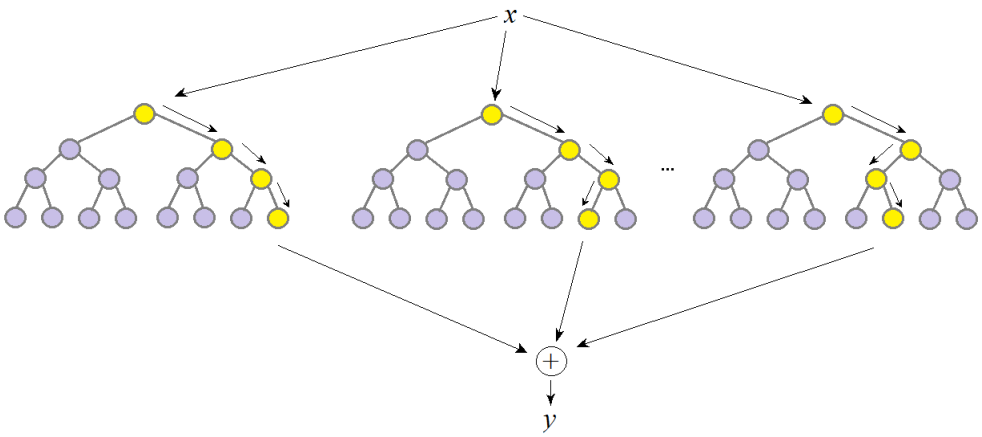

Figura 3-7: Random Forest [basado en (Kazoo, 2013)]

Cada decisión del árbol (es decir, cada variable en que el árbol se parte en dos posibles grupos de valores) se crea examinando cada variable y eligiendo el "mejor corte" de entre todas las variables. *Random Forests* selecciona un conjunto aleatorio de variables para cada corte. Por convención, el número de variables a considerar es $\sqrt{p}$, donde p es el número de entradas candidatas. Por ejemplo, si se tienen 100 variables de entrada, se seleccionan al azar 10 de ellas para cada corte. Esto fuerza los árboles a encontrar formas alternativas de predecir la variable objetivo, además de evitar correlaciones entre los árboles. Al final se mantiene el elemento de aleatoriedad en los conjuntos de entrenamiento, así como al realizar los cortes en los árboles. Obtenido el ensamble, se introducen variables desconocidas a través de cada árbol, y se obtiene una clase o nodo final en el árbol. La clase predicha es la moda (o el voto de la mayoría) de

entre las predicciones de los árboles de decisión, como se puede observar en el ejemplo de la figura 3-8.

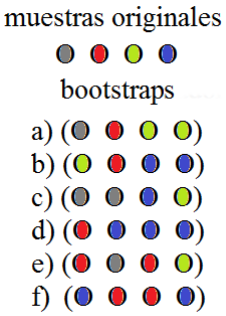

Figura 3-8: Ejemplo de bootstrapping (simplificado)

Dado que el *bootstrapping* deja algunos datos de prueba, se pueden usar los datos restantes para evaluar el error de la clasificación:

- Para cada muestra identifica los árboles que no usaron la muestra en el entrenamiento
- Obtiene el promedio de la exactitud prediciendo esta muestra
- Obtiene el promedio sobre todas las muestras

Esta estimación se denomina *out-of-bag error estimate* u ***oob estimate***.

Cabe mencionar que la implementación de *Random Forests* elegida para realizar la clasificación (scikit-learn) no usa el voto de mayoría para asignar las clases. En su lugar, calcula las probabilidades de las hojas de los B árboles l_j

$$P(c_i) = \frac{1}{B}\sum_j P(c_i|l_j)$$

En el documento que propone esta técnica (Breiman, 2001), se muestra que la tasa de error del bosque depende de la correlación entre cualesquiera dos árboles, donde incrementar la correlación incrementa la tasa de error del bosque. Además, también se muestra que depende de la fuerza de cada árbol individual en el bosque. Un árbol con una tasa baja de error es un clasificador fuerte. Incrementar la fuerza de los árboles individuales disminuye la tasa de error del bosque.

Considero que Bosques Aleatorios es una técnica adecuada para esta tarea por varias razones. Primeramente porque hay casos de éxito en otros idiomas en clasificación por este medio. Además, el método usado por Bosques Aleatorios para clasificar es muy adecuado para una tema tan grande y difícil de generalizar como ironía. El uso de muchos clasificadores débiles sobreentrenados le muestra al sistema diversos aspectos de las publicaciones irónicas, de forma que la decisión final es la más informada. Es apropiado también por su capacidad de ejecutarse en paralelo, ahorrando tiempo de procesamiento, y la forma en que realiza el muestreo, teniendo buenos resultados aún con pocas muestras.

Típicamente, *Random Forests* puede flexiblemente incorporar datos faltantes en los valores predictores. Cuando los datos faltantes se encuentran para una observación (o caso) particular durante la construcción del modelo, la predicción hecha para ese caso se basa en el último nodo (no terminal) precedente en el árbol respectivo. Por ejemplo, si en un punto en particular en la secuencia de árboles un valor predictor se selecciona en la raíz (o cualquier otro nodo no terminal) para algunos casos sin datos válidos, entonces la predicción para esos casos simplemente se basa en la media general en la raíz (o cualquier otro nodo no terminal). Por lo tanto, no hay necesidad de eliminar casos del análisis si hacen falta datos para algunos de los predictores. *Random Forests* también presenta una exactitud de predicción alta entre los algoritmos contemporáneos, además de que trabaja eficientemente con bases de datos masivas, y no sobreentrena.

3.3.2. *Support Vector Machines*

Support Vector Machines (SVMs, o Máquinas de Soporte Vectorial en español) son un conjunto de métodos relacionados para aprendizaje supervisado, aplicables a problemas de clasificación y regresión. Dado un conjunto de datos de entrenamiento, marcado como perteneciente a una de dos categorías, una máquina de soporte vectorial construye un modelo que, dado un ejemplo nuevo, lo asignará a una u otra categoría. Desde la presentación del clasificador SVM hace dos décadas (Vapnik, 1995), SVM se popularizó debido a sus sólidos fundamentos teóricos. El desarrollo de implementaciones eficientes ha llevado a numerosas aplicaciones.

El conjunto con el que se entrenará la máquina de soporte vectorial es representado como puntos en un espacio. Idealmente, los ejemplos de entrenamiento de una clase estarán geométri-

camente cerca entre sí. Por ejemplo, en la figura 3-9, se observa la separación entre hombres y mujeres por su relación altura/peso.

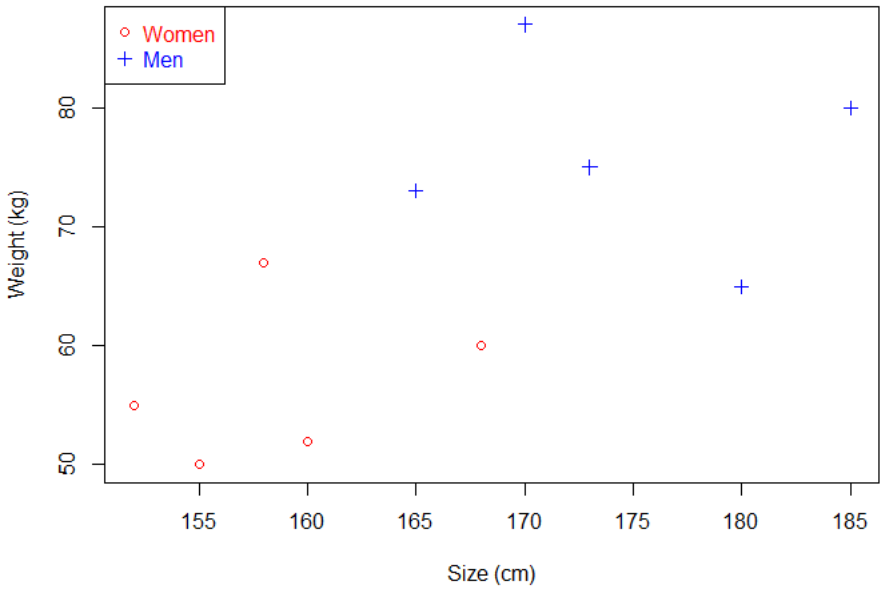

Figura 3-9: Datos en una representación vectorial [tomado de (Kowalczyk, 2014a)]

Se busca construir un clasificador que, dada la altura y peso de una persona nueva, determine si es hombre o mujer. En este ejemplo, se puede encontrar una separación dibujando una línea entre ellos.

Generalizando geométricamente, esta línea es un hiperplano, que como se puede observar en la figura 3-10, separa los elementos de ambas clases. Las Máquinas de Soporte Vectorial usan el hiperplano a modo de clasificador: la clase de un punto nuevo depende de su posición respecto al hiperplano.

La línea marcada en la figura 3-10, (también conocida como frontera de decisión) es uno de los posibles hiperplanos separadores; sin embargo, no es necesariamente el mejor. Intuitivamente, se tiene menos dudas de la clase de los puntos más lejanos a la frontera de decisión que de los más cercanos. Si el punto está muy cerca a la frontera de decisión, es probable que un cambio ligero a la misma cause que la predicción sea contraria. Un indicador de la "confianza" del modelo es qué tan amplio es el margen entre la frontera y los puntos más cercanos. El

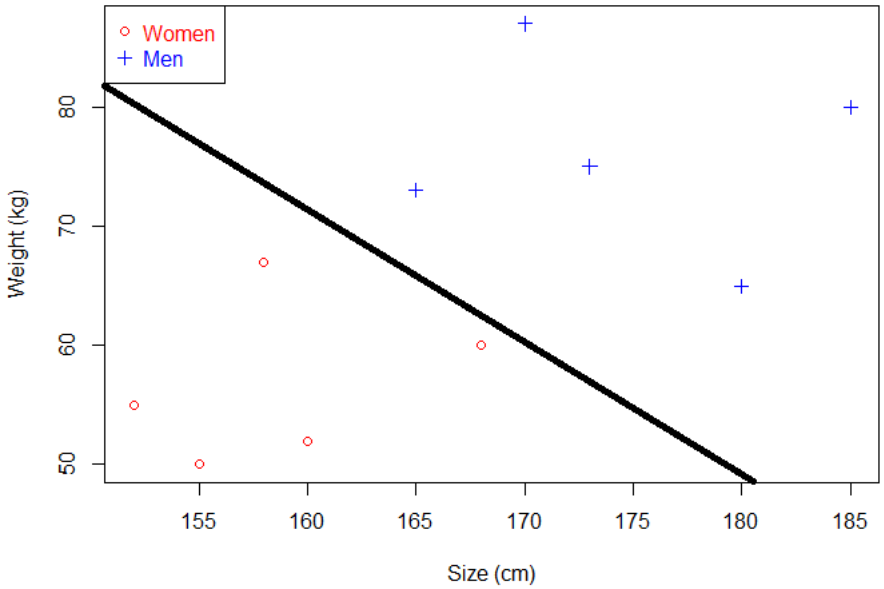

Figura 3-10: Separando una representación vectorial [tomado de (Kowalczyk, 2014b)]

objetivo de una máquina de soporte vectorial es encontrar el hiperplano separador óptimo que maximice el margen de los datos de entrenamiento. Para el caso lineal:

Se tienen L puntos de entrenamiento, donde cada entrada x_i tiene D atributos (es decir, de dimensión D) y está en una de dos clases $y_i = -1$ o $+1$. El conjunto de entrenamiento es de la forma:

$$\{x_i, y_i\} \text{ donde } \text{ i=1...L}, y_i \in -1, 1, x \in \mathbb{R}^D$$

Para este ejemplo se asume que los datos son linealmente separables, es decir, que se puede dibujar una línea en una gráfica de $x_1 vs x_2$ separando las dos clases cuando $D = 2$, y un hiperplano en gráficas de $x_1, x_2, ..., x_D$ para $D > 2$. Este hiperplano se puede describir por $\mathbf{x} \cdot \mathbf{w} + b = 0$ donde w es el vector ortogonal al hiperplano y $\frac{b}{\|w\|}$ es la distancia perpendicular del hiperplano al origen.

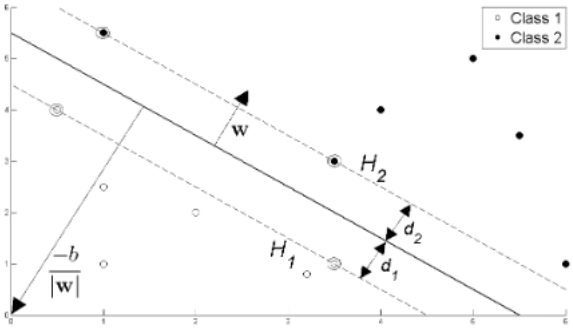

Figura 3-11: Hiperplano en dos clases linealmente separables [tomado de (Fletcher, 2009a)]

Para implementar una máquina de soporte vectorial se deben buscar las variables $\mathbf{x}$ y b para que nuestros datos de entrenamiento se puedan describir como: $\mathbf{x}_i \cdot \mathbf{w} + b \geq +1$ para $y_i = +1$ $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$ para $y_i = -1$, mismas que se pueden combinar en

$$y_i(\mathbf{x}_i \cdot \mathbf{w} - 1 \geq 0 \forall i) \tag{3-10}$$

Considerando únicamente los puntos por los que son atravesados, podemos describir H_1 y H_2 con:

$$\mathbf{x}_i \cdot \mathbf{w} + b = +1 \text{ para } H1$$

$$\mathbf{x}_i \cdot \mathbf{w} + b = -1 \text{ para } H2$$

En el mismo diagrama, definimos d_1 como la distancia de H_1 al hiperplano y d_2 desde H_2 al mismo. En este caso, el hiperplano es equidistante de H_1 y H_2, por lo que $d_1 = d_2$; el margen del hiperplano. También podemos observar que los puntos en los que recaigan H_1 y H_2 definen este margen. A estos puntos los llamamos los vectores de soporte (*support vectors* del inglés), ya que establecen las fronteras de cada clase. Para posicionar el hiperplano lo más lejos posible de los vectores de soporte, debemos maximizar el margen entre los mismos. Geométricamente, se observa que el margen equivale a $\frac{1}{\|w\|}$, y maximizarlo de acuerdo con las restricciones planteadas en la ecuación 3-10 es equivalente a encontrar:

$$min\|\mathbf{w}\| \text{ tal que } y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \forall i$$

Dado que minimizar $\|\mathbf{w}\|$ es equivalente a minimizar $1/2\|\mathbf{w}\|^2$, se usa este término para usar optimización con programación cuadrática posteriormente. Ahora se debe encontrar

$$min 1/2\|\mathbf{w}\|^2 \text{ tal que } y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \forall i$$

Para minimizar, se encuentran los multiplicadores de Lagrange α, donde $\alpha_i \geq 0 \forall i$:

$$L_P \equiv 1/2\|\mathbf{w}\|^2 - \alpha[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1) \forall i]$$

$$L_P \equiv 1/2\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$$

$$L_P \equiv 1/2\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{L} \alpha_i$$

Se busca encontrar $\mathbf{w}$ y b que minimicen, y α que maximice la ecuación anterior, manteniendo $\alpha_i \geq -\forall i$. Lo hacemos diferenciando L_P con respecto a $\mathbf{w}$ y b y fijando las derivadas a cero.

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i \tag{3-11}$$

$$\frac{\partial L_P}{\partial b} = 0 \implies \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{3-12}$$

$$L_P \equiv \sum_{i=1j}^{L} \alpha_i - 1/2 \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \text{ t.q. } \alpha_i \geq 0 \forall i, \sum_{i=1}^{L} \alpha y_i = 0$$

$$\equiv \sum_{i=1j}^{L} \alpha_i - 1/2 \sum_{i,j} \alpha_i H_{ij} \alpha_j \text{ donde } H_{ij} \equiv y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\equiv \sum_{i=1}^{L} \alpha_i - 1/2 \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \text{ t.q. } \alpha_i \geq 0 \forall i, \sum_{i=1}^{L} \alpha y_i = 0$$

$$max\alpha[\sum_{i=1}^{L}\alpha_i - 1/2\boldsymbol{\alpha}^T\mathbf{H}\boldsymbol{\alpha}] \text{ t.q. } \alpha_i \geq 0\forall i \text{ y } \sum_{i=1}^{L}\alpha y_i = 0$$

El anterior es un problema de optimización cuadrática, que podemos introducir a un programa y retornará α y usando la ecuación 3-11 podemos obtener $\mathbf{w}$. b se puede resolver usando las ecuaciones anteriores, con lo que se puede obtener el hiperplano óptimo.

En este ejemplo, el conjunto es linealmente separable. Para casos en los que no es completamente linealmente separable, se añade una holgura (o margen suave) desde la concepción de los hiperplanos paralelos al hiperplano separador. Para el margen suave, los puntos del lado incorrecto de la frontera del margen reciben una penalización que incrementa con la distancia desde él.

Además, se tienen los casos no-lineales. En el caso lineal anterior, se sugiere una matriz $\mathbf{H} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i{}^T\mathbf{x}_j$. Este es un ejemplo de funciones llamadas Funciones de Kernel, que definen una transformación a un vector de entrada a un espacio distinto (también llamado *espacio de características*). En este caso, $\mathbf{H}$ es un kernel lineal. Distintas funciones de kernel pueden ayudarnos a transformar los datos no linealmente separables a una dimensionalidad en que logren serlo. Para ello, se debe elegir el kernel que mejor convenga al conjunto de entrenamiento (Fletcher, 2009b; Ng, 2015-2016).

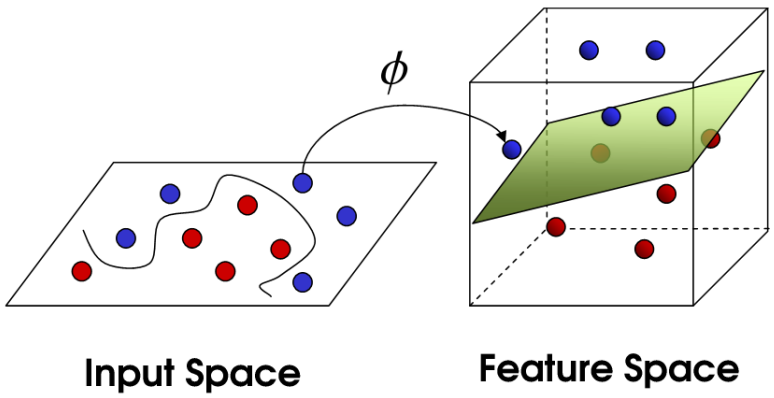

Figura 3-12: Ejemplo de hiperplano en dos clases no linealmente separables [tomado de (Joshi, 2012)]

Para este trabajo, se utilizó un kernel lineal, bajo la intuición de que los *tweets* de ambas

etiquetas difieren considerablemente, como la mayoría de los corpus textuales (Joachims, 1998); además de que se consideró el kernel lineal como un enfoque inicial, tras el cual se consideraría un cambio de estrategia de ser necesario.

Adicionalmente, se eligen las Máquinas de Soporte Vectorial por su popularidad entre las fuentes de este trabajo que también se dedican a clasificar *tweets* e ironía, con las cuales tuvo resultados satisfactorios.

Por facilidad se usa la paquetería Scikit-Learn[1], así como Gensim[2], NumPy[3] y SciPy[4], para el lenguaje de programación Python.

3.4. Arquitectura del sistema

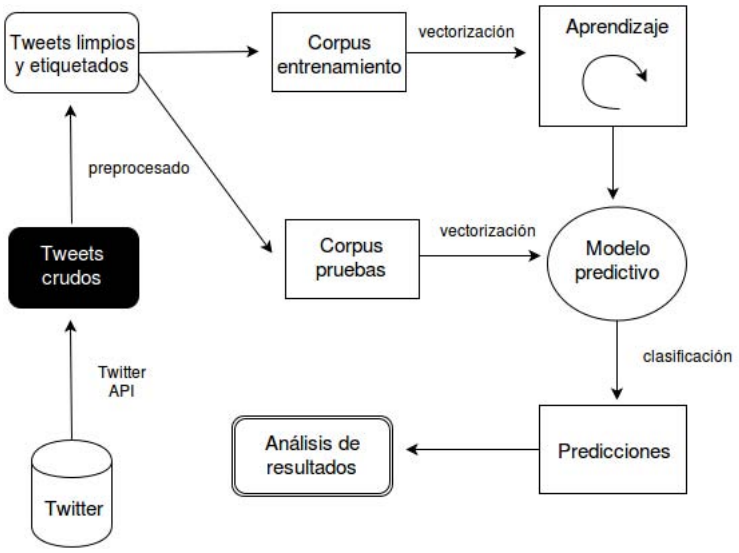

Figura 3-13: Arquitectura del sistema

Se diseñó un proceso para la ejecución de los experimentos y pruebas de este trabajo, descrito en la figura 3-13. Los *tweets* se recopilan desde el servidor de Twitter por medio de su API. Se almacena el crudo de los *tweets*. Posteriormente, pasan por el procesado descrito en el capítulo de Corpus. Cuando el corpus está limpio y etiquetado, se vectoriza y utiliza para entrenar un

[1] http://scikit-learn.org/
[2] https://radimrehurek.com/gensim/models/doc2vec.html
[3] http://www.numpy.org/
[4] http://www.scipy.org/

clasificador. Un grupo independiente de *tweets* extraídos por el mismo medio se usa para probar el clasificador recién entrenado, prediciendo las etiquetas de dichos *tweets*. Éstas se analizan, y si los resultados pueden ser mejorados, se realizan los ajustes necesarios para volver a entrenar.

3.5. Consideraciones

Para mejorar el desempeño del algoritmo, se buscan y eliminan las palabras vacías, es decir, palabras sin contenido; que suelen ser artículos, pronombres o preposiciones. Algunos ejemplos: *la, el, una, un, su, ellos, entre, hacia*, etc.

Adicionalmente, se propone una medida mínima para que un término sea considerado en el clasificador, es decir, las palabras con la cuenta $tf - idf$ más baja no se consideran. Esto pensando en acelerar el procesamiento, ya que incluir todos los términos es muy pesado computacionalmente.

Capítulo 4

Creación del corpus

En este capítulo se relata la motivación detrás de la propuesta de corpus presentada; así como el proceso de extracción, etiquetado, preprocesado y análisis del mismo.

4.1. Propuesta

Para este experimento, se requiere un corpus de textos cortos irónicos y no irónicos. Se propone que este corpus se busque en idioma español, debido a la falta de estudios en este idioma. Adicionalmente, se desea que el corpus provenga del sitio Twitter, debido a la facilidad de extracción y etiquetado, así como el límite por defecto de 140 caracteres. Además, se desea que el corpus sea lo más amplio posible, en contraste con la mayoría de los corpus usados en trabajos anteriores.

Se propone que los *tweets* que se usen para el corpus sean los etiquetados previamente por los usuarios como irónicos, como se obseva en los trabajos de Reyes et al. y Liebrecht et al. En la sección 1.3, se mostró una definición de ironía que incorpora sarcasmo como una especialización de la misma. Basándose en una inspección manual de los *tweets* etiquetados "ironía" y en la definición, muchas se salen por poco y entran a la de sarcasmo. Se observa que el fenómeno ocurre de ambos lados. Para los usuarios de Twitter, ambos términos se usan prácticamente de forma indistinta, probablemente debido a que no piensan concienzudamente en la definición de las palabras que usan cotidianamente. La propuesta de este enfoque es fusionar ambos *tweets*, para ofrecer una perspectiva más general de ironía, además de que sería la más apegada al

consenso de los usuarios.

También se propone que la clase contrastante se base en el enfoque de Liebrecht et al., en el que todo lo no explícitamente etiquetado como irónico se considera no irónico; enfoque que se observa en la figura 4-1, entre los de otros autores. Esto está sujeto a discusión, ya que no todos los usuarios etiquetan explícitamente sus textos irónicos. Se espera que el análisis al corpus esclarezca las similitudes y contrastes entre los textos de ambas clases.

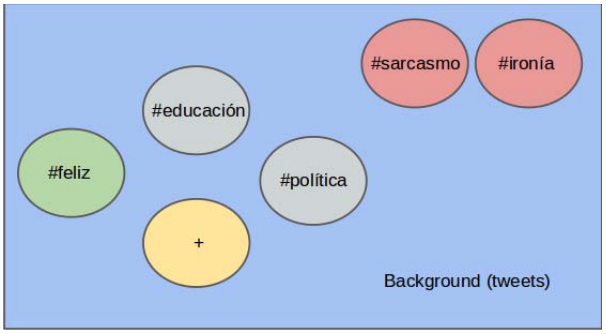

Figura 4-1: Enfoques para clase irónica y clases contrastantes

4.2. Extracción

Extraer *tweets* requiere de un usuario de Twitter, que otorga acceso a su API. Cada *tweet* es un objeto con ciertas características, como usuario (que lo publica), texto (el texto que se publica), geolocalización (de qué país/ciudad se originó el *tweet*), idioma, etc. Para la recuperación de *tweets* irónicos, se buscaron los *tweets* en español que incluyeran los hashtags #ironía y #sarcasmo, y se realizó una transmisión (*streaming*, en inglés) en vivo de los *tweets* nuevos. Esto se realizó en un período de alrededor de mes y medio. Simultáneamente, se extrajeron los *tweets* no irónicos. Para localizarlos, se realizó un *streaming* de los *tweets* con las palabras *que, qué, como, cómo, cuando, cuándo, donde, dónde, porque, por qué, este, esta* y *tiene*. Se eligieron estas palabras buscando que los *tweets* fueran descriptivos y generales, ya que buscar una palabra en particular haría que los resultados se orientaran hacia un tema en específico. Al tiempo que se recuperaban alrededor de catorce mil *tweets* irónicos, se recuperaron alrededor de seiscientos setenta y cinco mil *tweets* no irónicos. Se observa que en esta búsqueda los *tweets* irónicos representan menos del 5 % del total.

El proceso de extracción se realizó por medio de la biblioteca *twitter* de Erik Michaels-Ober[1] y un *script* propio hecho en el lenguaje de programación Ruby.

4.3. Etiquetado

La premisa de la extracción es que todos los *tweets* están automáticamente etiquetados, por lo que el único etiquetado posterior que se realiza es el de denominar lo etiquetado #ironía y #sarcasmo como **ironía**.

4.4. Preprocesado

Antes de ser analizado, el corpus es preprocesado para evitar algunos problemas:

4.4.1. Eliminar repeticiones en ambas clases

A simple vista se detectó un período en el que muchos *tweets* irónicos (y no irónicos) aparecían en la transmisión repetidamente, ya que eran parte de una estrategia política o publicitaria. Se optó por eliminar todas las repeticiones exactas (con exactamente el mismo texto, sin importar que vinieran de usuarios distintos), fueran de una cuenta o distintas, ya que sesgan el corpus hacia el contenido de los mismos. El *tweet* no se elimina por completo, simplemente sólo se incluye una de todas las repeticiones. Un ejemplo de un *tweet* eliminado es el siguiente: ***¡Cómo que la SCT y SEDESOL entregaron pantallas en para beneficiar al PRI!***, que aparece en el corpus recolectado muchas veces escrito exactamente igual.[2]

4.4.2. Eliminar irónicos dentro de no irónicos

Debido a lo amplio de los términos de búsqueda del conjunto no irónico, existe la posibilidad de que su transmisión recoja *tweets* marcados con #ironía o #sarcasmo. Posterior a la recopilación y a eliminar repeticiones, se separaron también los resultados entre los *tweets* no irónicos etiquetados con los *hashtags* irónicos. Esto se realizó por medio de una expresión regular, en la que basta que se incluya un *hashtag* que incorpore en algún punto las palabras, con o sin

[1] https://github.com/sferik/twitter

[2] Este *tweet* se cita sólo en texto y no en imagen ya que estos tweets han sido eliminados.

Figura 4-2: Ejemplos de elementos irónicos dentro de *tweets* no irónicos (ids: 606388650465755136, 605518376744583169)

acentos (en caso de ironía) o mayúsculas. De todos los *tweets* no irónicos, 28 fueron eliminados en este filtro. Un ejemplo de estos *tweets* se encuentra en la figura 4-2.

4.4.3. Formato de etiquetas

En Twitter se puede etiquetar cualquier palabra o texto, provocando que muchos usuarios no mantengan la formalidad en la etiqueta. Se etiquetan muchas palabras en el mismo texto (por ejemplo, *#tuitDelDía*), palabras en otros idiomas (por ejemplo, *#random*), entre otros usos. Esto puede tentar a eliminar las etiquetas del todo. Sin embargo se perdería mucha información ya que precisamente por el liberal uso de etiquetas, muchos usuarios escriben todo lo que quieren decir en las mismas y eliminarlas nos dejaría sin mensaje del todo.

Cabe mencionar también que dejar las etiquetas haría que la clasificación sea predispuesta por las etiquetas comunes a todos los *tweets* irónicos. Se opta por eliminar completamente las etiquetas que incluyan la palabra "ironía" o "sarcasmo" (en todas las variantes explicadas en el apartado anterior), y el resto reescribirlas sin lo que las hace etiquetas, es decir, el caracter #. De esta forma son parte del texto del *tweet*.

En el ejemplo de la figura 4-3, el texto recuperado del *tweet* es

Eres lo peor como ser humano solo violencia es ***@EnriqueVargasdV #VargasViolento http://t.co/q4LvqTzaWB***

El texto resultado de aplicar el formato de etiquetas es

Figura 4-3: Ejemplo de *tweet* con menciones, etiquetas e imagen (id: 605520629161410560)

Eres lo peor como ser humano solo violencia es @EnriqueVargasdV
VargasViolento http://t.co/q4LvqTzaWB

4.4.4. Formato de menciones

Se tiene un problema similar al de las etiquetas, ya que no se quiere excluir completamente la *idea* de una mención, pero se quiere evitar la diversidad de palabras que generarían los millones de nombres de usuarios de Twitter. Se reescriben todas las menciones como un mismo caracter (@), para preservar la idea de la mención sin caer en todos los nombres de usuarios. El *tweet* de la figura 4-3 queda ahora así:
Eres lo peor como ser humano solo violencia es @ VargasViolento http://t.co/q4LvqTzaWB

4.4.5. Formato de imágenes, vídeos e hipervínculos

Las imágenes, vídeos aparecen en el texto en forma de hipervínculos (así como los hipervínculos). Una vez más, para preservar el concepto del vínculo se reescriben todos los vínculos como un mismo símbolo (*http://link*). Ahora el *tweet* de la figura 4-3 resulta lo siguiente:
Eres lo peor como ser humano solo violencia es @ VargasViolento http://link

La extracción de etiquetas, menciones e hipervínculos se hace con la biblioteca *twitter-text* provista por Twitter para el lenguaje de programación Ruby.

4.4.6. Tokenización

Para un procesado más sencillo se separan las palabras por espacios, y se considera palabras a los signos de puntuación. Esto ayuda a evitar espacios excesivos, dar uniformidad a los textos y darle peso a los signos de puntuación. Para esta tarea se utiliza el tokenizador del paquete NLTK para el lenguaje de programación Python[3].

4.4.7. Palabras función

Se considera una lista de palabras función para retirar de todos los textos. Un ejemplo del *tweet* de la figura 4-3 sin palabras función:
eres peor ser humano violencia @ VargasViolento http://link

4.5. Análisis

Se extrajo una porción de *tweets* no irónicos del mismo tamaño que la cuenta total de *tweets* irónicos.

4.5.1. Datos generales

Se analizaron un total de 14511 *tweets* irónicos y no irónicos. Se extrajeron algunos datos generales, como cantidad total de palabras, palabras únicas (eliminando repeticiones), diversidad léxicaJohansson (2009) y longitud promedio de palabra, se pueden apreciar en la tabla 4-1.

Conjunto	Palabras	Palabras únicas	Diversidad léxica	Long. promedio palabra
Irónico	219,964	28,033	0.12	4
No irónico	228,922	23,920	0.10	3
Irónico sin p.f.	128,413	27,695	0.21	4
No irónico sin p.f.	124,559	23,586	0.19	4

Tabla 4-1: Datos del corpus de *tweets*.

[3]http://www.nltk.org/

La cuenta de elementos en las tablas 4-2 y 4-3 (sin palabras función) refleja que en ambos casos se usan pesadamente los símbolos, y en los irónicos se realizan más menciones. Adicionalmente, se realiza un listado basado en $tf - idf$Paltoglou y Thelwall (2010) (*Term Frequency-Inverse Document Frequency* o Frecuencia de Término-Frecuencia Inversa de Término, del inglés). Una palabra muy frecuente es determinada como no tan relevante para este algoritmo de pesado, de esta forma veremos qué tan distintas son las palabras con más alto y más bajo TF-IDF.

Palabra	**Cuenta**
@	5598
,	5485
.	5326
!	4092
:	3832
http	2934
//link:	2850
...	2850
?	2195
”	853

Tabla 4-2: Elementos en *tweets* irónicos (sin p.f.)

Palabra	**Cuenta**
,	6243
@	5348
.	4808
:	3862
?	3053
http	2763
//link:	2762
!	2411
te	2081
...	1380

Tabla 4-3: Elementos en *tweets* no irónicos (sin p.f.)

Palabra	
esfumando	retirarian
priista	acentos
corporacion	primermundista
debieron	muerete
citar	fomentar

Tabla 4-4: Palabras en irónicos con un alto tf-idf

Palabra	
loquisimo	colgo
coleccion	fantaseo
nubladon	felicitas
demuestras	avises
felicitan	metiche

Tabla 4-5: Palabras en no-irónicos con un alto tf-idf

Eliminando palabras función la prioridad de los elementos cambia, como se puede ver en las tablas 4-4 y 4-5. TF-IDF permite ver que los valores más altos de cada etiqueta son más diferentes de lo que dejan ver las cuentas de palabras, por lo que se incorpora TF-IDF a la vectorización final.

A pesar de las precauciones y medidas tomadas, hay ciertas complicaciones con el corpus que no se pueden reparar, ya que son inherentes al diseño del experimento. Un ejemplo de ello es la variación de la escritura en las redes sociales, donde se generan términos nuevos día a día, se cambia la ortografía de las palabras, entre otras ocurrencias. Se espera que este enfoque lo incorpore, en lugar de tratar de evitarlo.

Adicionalmente, no hay forma de saber si lo etiquetado irónico o sarcástico en realidad lo es. Y aunque se etiquete formalmente, se encontrará que no habrá un consenso en todos los casos, ya que lo irónico varía con el contexto del interlocutor. La premisa de este texto no es identificar la ironía como concepto literario y retórico, sino identificar lo que se considera ironía por la mayoría de usuarios en Twitter.

Capítulo 5

Experimentos

La sección de experimentos de este documento consiste en una serie de comparaciones entre métodos (clasificadores y vectorización), donde se busca obtener la combinación que produzca mejores resultados. Posteriormente, se evalúa por medio de algunas medidas estándar para la industria.

5.1. Evaluación

Para la evaluación de los resultados se revisan algunas métricas usadas ampliamente en los trabajos anteriores. Las dos más frecuentes y básicas medidas de efectividad en recuperación de información son precisión y cobertura (*recall* en inglés) (Manning, Raghavan, y Schütze, 2008).

5.1.1. Precisión y cobertura

Se definen para un caso simple en que un sistema que recupera información retorna un conjunto de documentos dada una consulta, que se puede extender a las otras situaciones, como clasificación.

Precision (P) es la fracción de documentos recuperados que son relevantes

$$Precision = \frac{\text{num elementos relevantes recuperados}}{\text{num elementos recuperados}} = P(\text{relevante} \mid \text{recuperado}) \qquad (5\text{-}1)$$

Recall (R) es la fracción de documentos relevantes recuperados

$$Recall = \frac{\text{num elementos relevantes recuperados}}{\text{num elementos relevantes}} = P(\text{recuperado} \mid \text{relevante}) \tag{5-2}$$

Éstas nociones son más claras examinando la tabla 5-1:

	Relevante	No relevante
Recuperado	positivos verdaderos (tp)	positivos falsos (fp)
No recuperado	negativos falsos (fn)	negativos verdaderos (tn)

Tabla 5-1: Relación entre elementos recuperados

Por lo que:

$$P = \frac{tp}{tp + fp} \tag{5-3}$$

$$R = \frac{tp}{tp + fn} \tag{5-4}$$

Es sencillo de entender la extrapolación de la clase relevante y no relevante cualquier problema de clasificación binaria.

5.1.2. Exactitud

Exactitud se refiere a las recuperaciones correctas en un sistema. A partir de la tabla anterior, se puede definir exactitud, o *accuracy* en inglés como la fracción de clasificaciones que son correctas, es decir:

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn} \tag{5-5}$$

Sin embargo, para tareas de recuperación de documentos (como ésta, en cierta medida), una exactitud alta no basta. Un sistema ajustado para tener la máxima exactitud puede aparentar tener buen desempeño etiquetando todo igual en una tarea donde la distribución de las clases está muy desbalanceada (ya que estadísticamente, la mayoría de los resultados están de un solo

lado). Se puede tener una alta exactitud, pero también con una tasa alta de falsos positivos. Es decir, un sistema muy exacto puede tener muchos resultados no relevantes.

5.1.3. Puntaje F

Las medidas de precisión y recuerdo concentran la evaluación en los positivos verdaderos que se recuperen, preguntando qué porcentaje de los documentos relevantes han sido encontrados y la tasa de falsos positivos.

Estas cualidades se balancean entre sí, ya que se puede tener un recuerdo de 1 con un baja precisión, recuperando todos los documentos para cualquier consulta, por ejemplo. Una medida de contraste entre precisión y recuerdo es el Puntaje F o *F-measure*, que es la media armónica pesada de la presición y el recuerdo.

$$F\text{-}score = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \tag{5-6}$$

donde

$$\beta^2 = \frac{1-\alpha}{\alpha}, \alpha \in [0,1] \tag{5-7}$$

Para $\beta = 1$

$$F_{\beta=1} = \frac{2PR}{P+R} \tag{5-8}$$

5.1.4. Matriz de confusión

Una matriz de confusión[1] contiene información acerca de las clasificaciones realizadas por un sistema de clasificación. El desempeño de estos sistemas es comúnmente evaluado usando los datos en dicha matriz. La siguiente tabla muestra la matriz de confusión para un clasificador de dos clases. Se puede observar en la tabla 5-2 que es una forma más enfocada de ver la tabla anterior.

Tp es el número de predicciones correctas de que una instancia es positiva, *fp* es el número de predicciones incorrectas de que una instancia será positiva, *fn* es el número de predicciones incorrectas de que una instancia es negativa y *tn* es el número de veces en que se predice correc-

[1] Kohavi y Provost, 1998

	Predicción positiva	Predicción negativa
Realmente positivo	tp	fn
Realmente negativo	fp	tn

Tabla 5-2: Elementos de una matriz de confusión

tamente que una instancia es negativa. A partir de la matriz de confusión se pueden obtener las medidas mencionadas anteriormente. Se adjunta la matriz de confusión a los resultados obtenidos en los experimentos, a modo de prueba.

5.2. Vectorización

Como se habló en la sección de arquitectura y de nuevo en la sección de corpus, los *tweets* se vectorizan antes de entrar al clasificador. Además del preprocesado al *tweet*, se hace un procesado posterior: se convierte todo a minúsculas y se eliminan los acentos *unicode*, para que no causen conflicto y diversidades innecesarias. Se tienen dos propuestas para definir lo que constituirá cada elementos a contar (también llamados *tokens*): por palabra y por n-grama.

5.2.1. Por palabra

Dentro de la vectorización por palabra, se realizaron dos líneas de experimentos. Siguiendo enfoques anteriores en clasificación de ironía, se usa una representación dispersa de los *tweets* para entrenar los clasificadores descritos anteriormente (*Support Vector Machines* y *Random Forests*). Esta representación dispersa está formada por la cuenta y puntuación tf-idf de unigramas, bigramas y trigramas de palabra. Por ejemplo, dentro de la frase *Esta es una frase* hay cuatro unigramas (*esta, es, una, frase*), tres bigramas (*esta es, es una, una frase*) y dos trigramas (*esta es una, es una frase*).

La segunda línea por palabra usa una representación distribuida de los *tweets* basada en la herramienta *word2vec* mencionada en el capítulo de método. El objetivo es determinar la relación de cada palabra con su contexto por medio de operaciones vectoriales.

5.2.2. Por carácter

Para el enfoque basado en caracteres se usan n-gramas de caracter, una característica que probó ser representativa en los trabajos de Reyes et al. y Ptácek et al.. Se usa el análisis previo del corpus para elegir la cantidad de caracteres mínimo y máximo de n-gramas. La cantidad promedio de caracteres en lo etiquetado como irónico es de 4, mientras que en lo etiquetado como no-irónico es de 3, por lo que se usan 4 caracteres como máximo para considerar palabras completas, empezando desde 2. Esto significa que se usarán como medida bigramas, trigramas y tetragramas de caracter. Este enfoque también es capaz de recuperar otras características existentes, como signos de puntuación, *emoji* y emoticones.

El contenido generado por usuarios está plagado de irregularidades, como errores ortográficos o de dedo. Los n-gramas de caracter tienen la ventaja de adaptarse a los vicios de usuarios, gracias a que identifican n-gramas que simbolizan alguna característica lingüística, como un lema, prefijo sufijo. Por ejemplo, dadas algunas escrituras encontradas de la palabra "este": *este, estee, eeeste, eestee*. Todas contienen el trigrama *est*, aún si tres de ellas no están escritas correctamente. Con los enfoques basados en palabras, este tipo de diversidad resulta en muchas *features* o características con baja frecuencia que podrían estar relacionadas entre sí.

5.2.3. Consideraciones de la implementación

La experimentación fue realizada usando Máquinas de Soporte Vectorial (SVM de sus siglas en inglés) y Bosques Aleatorios (RF de sus siglas en inglés) como clasificadores. Para ambos, se usa la implementación de *scikit-learn*. SVM usa un kernel lineal y su configuración no fue alterada. En el caso de RF, se usaron 1,000 estimadores. Se decide usar estos clasificadores siguiendo los trabajos previos de González-Ibáñez y cols. (2011); Ptácek y cols. (2014) para Máquinas de Soporte Vectorial, así como Reyes y cols. (2013) para Bosques Aleatorios.

Adicionalmente, se usa $tf - idf$ (frecuencia de término - frecuencia inversa de documento, de sus siglas en inglés) para la representación de palabras y caracteres, ya que tiende a favorecer los términos relevantes a través de todos los documentos. Palabras vacías comunes se excluyen con una lista negra de palabras. Para crear el modelo distribucional, se entrena un espacio vectorial usando *doc2vec* con aproximadamente $660,000$ *tweets* no irónicos implementando un modelo *C-BOW*. Se toman las medidas necesarias para asegurar que el modelo distribucional

no contenga los *tweets* no irónicos de prueba, de forma que le sean desconocidos hasta que se realice dicha prueba.

Distribución del corpus

Se realizarán experimentos sobre ciertas distribuciones de *tweets* irónicos y no irónicos. Como se mencionó en los antecedentes, para el entrenamiento de la mayoría de ellos se usa una cantidad igual por clase. En este caso se tienen solamente dos clases. Se realizarán los experimentos con la distribución balanceada, pero además con la distribución 70 % no irónicos - 30 % propuesta por Reyes et al. y con una propuesta en este documento, que trata de ser más realista, de 90 % no irónicos y 10 % irónicos. La tabla 5-3 muestra el número de *tweets* usados en cada caso.

	Irónicos	No irónicos
	Balanceado	
Entrenamiento	14,511	14,511
Prueba	1,483	1,483
	No balanceado	
Entrenamiento	14,511	33,859
Prueba	1,483	3,458
	Propuesta	
Entrenamiento	14,511	130,599
Prueba	1,483	13,347

Tabla 5-3: Distribuciones de datos usadas en este trabajo

5.3. Experimentos y resultados

Además de cambiar las proporciones en los datos de prueba, se probaron tres líneas de estudio: *palabra-grama* (llamado *word-gram* en este documento), basado en representar el *tweet* como un vector disperso de los pesos $tf-idf$ de sus palabras; *word2vec*, basado en representar el *tweet* como un vector distribucional basado en la herramienta *word2vec*; y finalmente, *caracter-grama* (llamado *char-gram* en este documento), basado en la representación del *tweet* como un vector disperso de los pesos $tf-idf$ de los caracteres. En las secciones siguientes, se presentan los resultados principales para cada línea. Los resultados se presentan en inglés, recordando

que exactitud se traduce a *accuracy*, precisión a *precision*, cobertura a *recall* y el puntaje F a *F-score*.

5.3.1. Distribución balanceada

	word-gram		*word2vec*		char-gram	
	SVM	RF	SVM	RF	SVM	RF
accuracy	0.68	0.66	0.78	0.76	0.87	0.88
precision	0.67	0.66	0.77	0.76	0.87	0.93
recall	0.69	0.68	0.79	0.78	0.86	0.81
F-score	**0.68**	0.65	**0.78**	0.76	0.86	**0.87**

Tabla 5-4: Resultados promedio por base y clasificador (50-50)

En las tabla 5-4 se observan los resultados promediados de ambas clases en cuanto a exactitud, precisión, cobertura y puntaje F, de acuerdo con cada base y clasificador. En general, los resultados para cada base-clasificador rondan alrededor de las mismas cifras, salvo el resultado de *char-gram* para Bosques Aleatorios, que tiene una precisión ligeramente más alta que el resto de sus resultados. A primera vista se aprecia que *char-gram* tiene el mejor desempeño en general.

Inspeccionando el resultado no promediado podemos ver la puntación obtenida por cada clase (es decir, los resultados de la clasificación de los elementos irónicos/no irónicos, respectivamente). Se presentan los datos de la tabla 5-5 únicamente para *Support Vector Machines* en todas las bases. Análogamente, se tiene la tabla 5-6 para *Random Forests*. Entre las bases por palabra, *word2vec* parece tener las mejores estadísticas para ironía, y en *char-gram*, *Random Forests* es más preciso que *Support Vector Machines*, a pesar de que éste último recupera más elementos irónicos que el anterior.

SVM						
	word-gram		*word2vec*		char-gram	
	ironía	no ironía	ironía	no ironía	ironía	no ironía
precision	0.67	0.68	0.77	0.79	0.87	0.87
recall	0.67	0.66	0.79	0.76	0.87	0.87
F-score	0.68	0.67	0.78	0.78	0.87	0.87

Tabla 5-5: Resultados por clase para todas las bases con SVM (50-50)

RF						
	word		*word2vec*		char	
	ironía	no ironía	ironía	no ironía	ironía	no ironía
precision	0.66	0.67	0.77	0.76	0.94	0.84
recall	0.68	0.65	0.76	0.77	0.81	0.94
F-score	0.67	0.66	0.76	0.76	0.87	0.89

Tabla 5-6: Resultados por clase para todas las bases con RF (50-50)

La tabla 5-7 muestra los puntajes f para cada experimento en una distribución balanceada. También resume el desempeño de otros trabajos en otros idiomas donde sean comparables. Se observa que a nivel de palabra, la base *word2vec* supera la de *word-gram*. Esto es entendible, ya que los vectores de *word2vec* consideran una descripción muy amplia del lenguaje que se habla en *Twitter* para calcular el modelo distribucional. El mejor resultado obtenido en este nivel, un puntaje F de **0.78** con SVM, es cercano al de Tungthamthiti et al. que fue de 0.79, con un corpus balanceado en inglés y un conjunto de características para vectorizar muy amplio. Por otro lado, se nota que el mejor resultado es a nivel de caracteres, **0.87** con el clasificador de Bosques Aleatorios. En este nivel queda sólo segundo ante el mejor resultado hasta ahora en todos los idiomas por Ptácek et al., más alto que la puntuación para el idioma inglés por Reyes et al., y ampliamente mejor que intentos previos en checo y portugués. Es decir, es un resultado comparable a los otros lenguajes en una distribución balanceada.

Base	RF	SVM
Nivel palabra		
word-gram	0.68	0.67
word2vec	0.76	0.78
Para otros idiomas		
Davidov et al. (2010)	N/A	0.83
Tungthamthiti et al. (2014)	N/A	0.79
Nivel caracter		
char-gram	**0.87**	**0.86**
Para otros idiomas		
Reyes et al. (2013)	N/A	0.71
Ptácek et al. (2014)(inglés)	N/A	**0.93**

Tabla 5-7: Puntaje F para todas las bases bajo una distribución balanceada de *tweets* (50-50)

5.3.2. Distribución no balanceada

	word-gram		*word2vec*		char-gram	
	SVM	RF	SVM	RF	SVM	RF
accuracy	0.70	0.56	0.78	0.75	0.88	0.90
precision	0.50	0.37	0.66	0.74	0.81	0.95
recall	0.29	0.68	0.57	0.26	0.79	0.70
F-score	0.37	**0.48**	**0.61**	0.38	**0.80**	**0.80**

Tabla 5-8: Resultados promedio por base y clasificador (70-30)

La tabla 5-8 muestra los resultados promedio por base y clasificador de la segunda distribución (30 % *tweets* irónicos, 70 % *tweets* no irónicos). Inmediatamente se aprecia que el desempeño bajó considerablemente. A pesar de que se esperaba esto, no se esperaba la dureza de la misma. Para *word-gram* con SVM se tiene una alta exactitud pero precisión pobre, así como para *word2vec* con SVM. Con *Random Forests*, por otro lado, se tiene una alta exactitud y precisión, pero un recuerdo muy bajo; lo que significa que se recuperó tan sólo un 54 % de los elementos, pero fueron correctamente clasificados. Los puntajes F de las dos bases por palabra son menores a 70. Por otro lado, se observa que la base por caracter se mantiene estable en ambos clasificadores.

En la tabla 5-9 se tiene el desglose de la clasificación por clase. Bosques Aleatorios es más preciso que Máquinas de Soporte Vectorial, pero de nuevo con un recuerdo más bajo, que al final promedia el mismo puntaje F. Sin embargo, en este caso Bosques Aleatorios es mejor en reconocer no ironía.

SVM						
	word		*word2vec*		char	
	ironía	no ironía	ironía	no ironía	ironía	no ironía
precision	0.50	0.74	0.66	0.83	0.81	0.91
recall	0.29	0.87	0.57	0.87	0.80	0.92
F-score	0.37	0.80	0.85	0.61	0.80	0.92

Tabla 5-9: Resultados por clase para todas las bases con SVM (70-30)

La tabla 5-11 presenta los puntajes F en relación a otros autores. Reyes et al. y Ptácek et al. proponen configuraciones no balanceadas similares (30-70 y 25-75, respectivamente), donde el primero reporta una caída en el desempeño.

RF						
	word		*word2vec*		char	
	ironía	no ironía	ironía	no ironía	ironía	no ironía
precision	0.37	0.79	0.74	0.75	0.96	0.88
recall	0.68	0.51	0.26	0.96	0.69	0.99
F-score	0.48	0.62	0.38	0.84	0.80	0.93

Tabla 5-10: Resultados por clase para todas las bases con RF (70-30)

Base	RF	SVM
Nivel palabra		
word-gram	0.48	0.37
word2vec	0.38	0.61
Para otros idiomas		
Reyes et al. (2013)	N/A	0.53
Ptácek et al. (2014) *(inglés 1:3)*	N/A	0.92
Nivel caracter		
char-gram	**0.80**	**0.80**

Tabla 5-11: Puntajes F para todas las bases bajo una distribución no balanceada de *tweets* (30-70)

5.3.3. Distribución propuesta

Dados los resultados con la distribución no balanceada, se busca probar la flexibilidad de la representación nivel caracter con una distribución más realista. Para esto se propone una configuración formada por 10 % elementos irónicos y 90 % elementos no irónicos.

char-gram				
	SVM		RF	
	ironía	no ironía	ironía	no ironía
precision	0.88	0.96	0.99	0.94
recall	0.64	0.99	0.44	1.00
F-score	0.74	0.98	0.61	0.97

Tabla 5-12: Resultados por clase para char-gram (90-10)

Observando los casos separados por clase en la tabla 5-12, resalta que siempre hay un resultado superior para no ironía que para ironía, lo cual es entendible por la cantidad de elementos. En la tabla 5-13 se puede observar los resultados generales de esta distribución a

nivel caracter. Ambos presentan una alta exactitud pero un recuerdo muy bajo. Para ambos clasificadores el desempeño decae; sin embargo, en el caso de SVM el desempeño continúa siendo competitivo con un puntaje F de 0.74. La tabla 5-14 presenta la matriz de confusión de la última distribución, donde se aprecia la mayor cantidad de elementos correctamente clasificados de SVM.

char-gram		
	SVM	RF
accuracy	0.95	0.94
precision	0.88	0.99
recall	0.64	0.44
F-score	**0.74**	0.61

Tabla 5-13: Resultados generales por de char-gram para todos los clasificadores (90-10)

SVM		
	positivo	negativo
realmente positivo	955	528
realmente negativo	134	13213
RF		
	positivo	negativo
realmente positivo	657	826
realmente negativo	7	13340

Tabla 5-14: Matriz de confusión para clasificaciones (90-10)

5.4. Análisis

Se obtuvieron resultados favorables en todas las líneas base, donde los mejores vinieron de la base a nivel caracter para las tres configuraciones del corpus, indicando que los n-gramas de caracter son buenos indicadores para ironía en el idioma español y textos cortos. Éstos pueden resultar útiles para representar el lenguaje coloquial de los usuarios y la alta expresividad de las frases de pocos caracteres. Además, parecen trabajar a favor de la diversidad en el vocabulario (ocasionada por errores ortográficos y abreviaturas, escritos en ocasión a propósito), siempre presente en el contenido generado por usuarios.

Es posible que para textos muy cortos como son los *tweets*, las características de vectorización basadas en palabras no logren asimilar suficiente información por *tweet* para representarlo

correctamente, mientras que un modelo basado en caracteres divide un enunciado en muchas más características, teniendo una imagen más clara de cada *tweet*. Un resultado favorable para los n-gramas de caracter se refleja también en los trabajos de Ptácek et al. y Reyes et al.

Tras estos resultados, se exploraron las cuentas y pesos $tf - idf$ de las características más comunes y se encontró que los *emoji* y emoticones (por ejemplo, *:), :(, :D, :'(*, entre otras) tienen puntuaciones altas. Las "expresiones de gracia" como *jajaja* y *lol* existen en ambos conjuntos de datos, pero son más representativas del conjunto irónico; conforme a lo reportado por Carvalho y cols. (2009); Liebrecht y cols. (2013); Reyes y cols. (2013); Tsur y Davidov (2010).

También se observa una cuenta muy alta de morfemas comunes en español. Es posible que los n-gramas de caracter contengan mucha información morfológica, y sean capaces de extraer morfemas comunes para usar como características altas o bajas para una cierta clase. Además, por lo descrito anteriormente, los morfemas recuperados por n-gramas de caracter pueden en parte representar la diversidad y variaciones en el lenguaje coloquial del contenido generado por usuarios.

Bosques Aleatorios demostró los mejores resultados en un conjunto de datos balanceado, mientras que Máquinas de Soporte Vectorial lo hizo en un conjunto no balanceado (y más amplio). En el último caso el desempeño de Bosques Aleatorios decae, se vuelve más tardado y pesado computacionalmente, posiblemente por la forma en que se cargan y manejan los datos en cada algoritmo.

El mejor resultado a la fecha, por Ptácek et al., usa una combinación de características, donde las predominantes son estilísticas. Su sistema reporta los mejores resultados para el inglés. En el resumen de antecedentes resaltan este y los otros diferentes enfoques hacia la vectorización en el problema de ironía, donde la mayoría de los autores idean sus propias métricas basadas en una interpretación del sarcasmo o ironía. Sin embargo, la mayoría de los autores reportan que estas características no contribuyen de forma significativa a sus sistemas. Las características basadas en una cierta intuición tienden a recaer en palabras claves o indicadores específicos al dominio y resultan inflexibles al cambio, funcionando mejor en entornos controlados. Los mejores resultados en la literatura previa (aún sobre idiomas distintos al español) se obtienen usando una mayoría de características de estilo, por encima de las características con contenido emocional o gramatical. Este experimento es consistente con esas conclusiones, produciendo

resultados comparables al estado del arte usando características de estilo sumamente sencillas.

Capítulo 6

Conclusiones

Los resultados obtenidos en este experimento sugieren que en su mayoría se logró identificar ironía en textos cortos en español. Esto se logró por medio de un enfoque de clasificación binaria para textos cortos, distinguiendo entre irónico y no irónico. La investigación se enfoca en ironía verbal, en la cual el concepto de ironía incluye el del sarcasmo.

Siguiendo enfoques anteriores, se aprovechan textos cortos (*tweets*) etiquetados manualmente por medio de las etiquetas *#ironía* y *#sarcasmo*. Se amasa un corpus de los mismos, además de otro de *tweets* no marcados con dichas etiquetas, a modo de comparación.

Se proponen tres bases para sistemas de clasificación enfocados en características simples de palabra y caracter para *tweets* en español. La primer base consiste en representar los *tweets* con los pesos $tf - idf$ de los n-gramas de palabra. La segunda consiste en una representación distribucional de los *tweets*. Finalmente, la tercera base representa a los *tweets* con los pesos $tf - idf$ de los n-gramas de caracter.

Los enfoques llegan a resultados comparables a los trabajos relacionados en otros idiomas. La base a nivel caracter supera algunos enfoques en otros idiomas. Se alcanzaron puntajes F de 0.87 en un conjunto balanceado usando un clasificador de Bosques Aleatorios, 0.80 en una configuración no balanceada (70/30) y 0.74 en una configuración aún más imbalanceada pero más realista (90/10), en ambos casos usando un clasificador de Máquinas de Soporte Vectorial.

Se observa que las características basadas en caracteres son buenos indicadores para detección de ironía, y en general ofrecen una buena base para el idioma español.

En el desarrollo de este trabajo se recopiló un corpus de tamaño considerable en idioma

español de *tweets* irónicos y no irónicos. Este corpus originó un proyecto de etiquetado, apoyado por la Red Temática en Tecnologías del Lenguaje[1], que ahora está públicamente disponible en **http://turing.iimas.unam.mx/ironia/**.

Adicionalmente, se envió un artículo a la Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural[2], que fue aceptado para su publicación en su 56avo ejemplar con reseñas favorables[3].

6.1. Trabajo futuro

El enfoque de esta investigación es muy preciso, por lo que se omiten diversas variaciones o configuraciones que podrían ofrecer nueva información o incrementar el desempeño de la clasificación. Un ejemplo es ampliar el enfoque para que pueda distinguir entre sarcasmo e ironía, además de lo que no sea ninguna de ellas.

Para este trabajo se consideran características de vectorización muy simples ya que es una primera aproximación hacia el problema. Estas bases aún se pueden mezclar entre sí, o ampliar agregando más características de estilo y características con un enfoque lingüístico.

Se obtuvieron resultados favorables a nivel palabra también, en la línea de *word2vec*. Este experimento es relativamente novedoso, ya que la herramienta fue creada muy recientemente. Para el presente trabajo, se usó la configuración más básica de la herramienta. Sus variantes y usos se siguen explorando y mejorando día con día. Por ello, el experimento con *word2vec* aún puede ser alterado para mejorar su desempeño y resultados. La configuración de los clasificadores usados también fue muy sencilla. Es posible modificar los parámetros de ambos para obtener un desempeño superior, además de probar otros clasificadores.

Finalmente, el corpus debe ser ampliado para diversificar los *tweets*, en particular las olas de *tweets* referentes a un evento o acontecimiento (una campaña electoral, el lanzamiento de un disco, un partido deportivo, etc).

[1] http://ltl.inaoep.mx/RedTTL/
[2] www.sepln.org
[3] http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5285

Referencias

Bamman, D., y Smith, N. A. (2015). Contextualized sarcasm detection on twitter. En *Proceedings of the ninth international conference on web and social media, ICWSM 2015, university of oxford, oxford, uk, may 26-29, 2015* (pp. 574–577). Descargado de `http://www.aaai.org/ocs/index.php/ICWSM/ICWSM15/paper/view/10538`

Barbieri, F., y Saggion, H. (2014, may). Modelling irony in twitter: Feature analysis and evaluation. En N. C. C. Chair) y cols. (Eds.), *Proceedings of the ninth international conference on language resources and evaluation (lrec'14).* Reykjavik, Iceland: European Language Resources Association (ELRA).

Barrientos, J. (1998). *El lenguaje literario: Las figuras retóricas.* Arco Libros. Descargado de `https://books.google.com.mx/books?id=clagRf3sv58C`

Bosco, C., Patti, V., y Bolioli, A. (2013). Developing corpora for sentiment analysis: The case of irony and senti-tut. *IEEE Intelligent Systems*, *28*(2), 55-63. doi: http://doi.ieeecomputersociety.org/10.1109/MIS.2013.28

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Carvalho, P., Sarmento, L., Silva, M. J., y de Oliveira, E. (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* (pp. 53–56). New York, NY, USA: ACM. Descargado de `http://doi.acm.org/10.1145/1651461.1651471` doi: 10.1145/1651461.1651471

Davidov, D., Tsur, O., y Rappoport, A. (2010, julio). Semi-Supervised recognition of sarcastic sentences in twitter and amazon. En *Proceeding of the 23rd international conference on computational linguistics (coling).*

Edyburn, K. (2015). *Text compactor: Free online automatic text summarization tool.* Descargado de http://textcompactor.com/ (En línea, visitado Abril 5 2016 en http://textcompactor.com/)

Fletcher, T. (2009a). *Figure 1: Hyperplane through two linearly separable classes.* University College London. Descargado de http://www.tristanfletcher.co.uk/SVM%20Explained.pdf ([En línea, visitado Abril 5, 2016 en http://www.tristanfletcher.co.uk/SVM%20Explained.pdf])

Fletcher, T. (2009b). *Support vector machines explained*. University College London. Descargado de http://www.tristanfletcher.co.uk/SVM%20Explained.pdf (En línea, visitado Abril 5 2016 en http://www.tristanfletcher.co.uk/SVM%20Explained.pdf)

Ghosh, A., Li, G., Veale, T., Rosso, P., Shutova, E., Barnden, J., y Reyes, A. (2015, June). Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. En *Proceedings of the 9th international workshop on semantic evaluation (semeval 2015)* (pp. 470–478). Denver, Colorado: Association for Computational Linguistics. Descargado de http://www.aclweb.org/anthology/S15-2080

Go, A., Bhayani, R., y Huang, L. (2015). *Sentiment140- a twitter sentiment analysis tool.* Descargado de http://www.sentiment140.com/ (En línea, visitado Abril 5 2016 en http://www.sentiment140.com/)

Goldberg, Y., y Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

González-Ibáñez, R., Muresan, S., y Wacholder, N. (2011). Identifying sarcasm in twitter: A closer look. En *Acl (short papers)* (p. 581-586). The Association for Computer Linguistics.

Haverkate, H. (1990). *A speech act analysis of irony* (Vol. 14) (n.º 1). Elsevier.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *European Conference on Machine Learning (ECML)*, 137–142.

Johansson, V. (2009). Lexical diversity and lexical density in speech and writing: a developmental perspective. *Working Papers in Linguistics*, *53*, 61–79.

Joshi, P. (2012). *2d to 3d projection.* Descargado de https://prateekvjoshi.files.wordpress.com/2012/08/2d-to-3d-projection.jpeg ([En línea, visitado Abril 5, 2016 en https://prateekvjoshi.files.wordpress.com/2012/08/2d-

to-3d-projection.jpeg])

Kazoo, M. (2013). *Random forest and its derivatives algorithm.* Descargado de http://cdn-ak.f.st-hatena.com/images/fotolife/k/kazoo04/20131204/20131204173330.png ([En línea, visitado Abril 5, 2016 en http://cdn-ak.f.st-hatena.com/images/fotolife/k/kazoo04/20131204/20131204173330.png])

Kowalczyk, A. (2014a). *Understanding the math - part 1 - the margin.* Descargado de http://i2.wp.com/www.svm-tutorial.com/wp-content/uploads/2014/11/01_svm-dataset1.png ([En línea, visitado Abril 5, 2016])

Kowalczyk, A. (2014b). *Understanding the math - part 1 - the margin.* Descargado de http://i2.wp.com/www.svm-tutorial.com/wp-content/uploads/2014/11/01_svm-dataset1-sep ([En línea, visitado Abril 5, 2016 en http://i2.wp.com/www.svm-tutorial.com/wp-content/uploads/2014/11/01_svm-dataset1-separated.png])

Liebrecht, C., Kunneman, F., y van den Bosch, A. (2013, Jun). The perfect solution for detecting sarcasm in tweets #not. En *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis.* Association for Computational Linguistics. Descargado de http://www.aclweb.org/anthology/W13-1605

Lloret, E., y Palomar, M. (2012, enero). Text summarisation in progress: A literature review. *Artif. Intell. Rev.*, *37*(1), 1–41. Descargado de http://dx.doi.org/10.1007/s10462-011-9216-z doi: 10.1007/s10462-011-9216-z

Manning, C. D., Raghavan, P., y Schütze, H. (2008). *Introduction to information retrieval.* Cambridge: Cambridge University Press. Descargado de http://nlp.stanford.edu/IR-book/

Maynard, D., y Greenwood, M. A. (2014). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. En *Proceedings of the ninth international conference on language resources and evaluation (lrec-2014), reykjavik, iceland, may 26-31, 2014.* (pp. 4238–4243). Descargado de http://www.lrec-conf.org/proceedings/lrec2014/summaries/67.html

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. Descargado de

http://arxiv.org/abs/1301.3781

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

Monlau, P. (1842). *Elementos de literatura, ó, arte de componer en prosa y verso: para uso de las universidades é institutos.* Impr. y Libr. de Pablo Riera. Descargado de https://books.google.com.mx/books?id=MPlQAAAAcAAJ

Mukerjee, S. (2013). *Decision tree old kiwi.* Descargado de http://3.bp.blogspot.com/-uHx1X7KXAdU/Udo_SdpidxI/AAAAAAAAHPY/_3mf7zhATLk/s1600/Deci ([En línea, visitado Abril 5, 2016 en http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf])

Ng, A. (2015-2016). *Cs229 lecture notes: Part v support vector machines.* Stanford University. Descargado de http://cs229.stanford.edu/notes/cs229-notes3.pdf (En línea, visitado Abril 5 2016 en http://cs229.stanford.edu/notes/cs229-notes3.pdf)

Paltoglou, G., y Thelwall, M. (2010). A study of information retrieval weighting schemes for sentiment analysis. En *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 1386–1395). Stroudsburg, PA, USA: Association for Computational Linguistics. Descargado de http://dl.acm.org/citation.cfm?id=1858681.1858822

Peña, E. B. (2012). El discurso de los parlamentarios y parlamentarias andaluces. análisis pragmalingüístico de la ironía verbal. *Discurso & Sociedad*, *6*(1), 79–97.

Ptácek, T., Habernal, I., y Hong, J. (2014). Sarcasm detection on czech and english twitter. En *Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers* (pp. 213–223).

Rangel, F., y Rosso, P. (2013). Use of language and author profiling: Identification of gender and age. En *Proceedings of the 10th workshop on natural language processing and cognitive science (nlpcs-2013).*

Reyes, A., Rosso, P., y Veale, T. (2013). A multidimensional approach for detecting irony in twitter. *Language Resources and Evaluation*, *47*(1), 239–268. Descargado de http://dx.doi.org/10.1007/s10579-012-9196-x doi: 10.1007/s10579-012-9196-x

Rong, X. (2014a). *Figure 1.* Descargado de http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf ([En línea, visitado Abril 5, 2016 en http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf])

Rong, X. (2014b). *Figure 2.* Descargado de http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf ([En línea, visitado Abril 5, 2016 en http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf])

Rong, X. (2014c). *Figure 3.* Descargado de http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf ([En línea, visitado Abril 5, 2016 en http://www-personal.umich.edu/ ronxin/pdf/w2vexp.pdf])

Rong, X. (2014d). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Tang, Y.-j., y Chen, H.-H. (2014). Chinese irony corpus construction and ironic structure analysis. En *Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers* (pp. 1269–1278).

Tsur, O., y Davidov, D. (2010). Icwsm – a great catchy name: Semi-supervised recognition of sarcastic sentences in product reviews. En *In international aaai conference on weblogs and social.*

Tungthamthiti, P., Kiyoaki, S., y Mohd, M. (2014, December). Recognition of sarcasms in tweets based on concept level sentiment analysis and supervised learning approaches. En *Proceedings of the 28th pacific asia conference on language, information, and computation* (pp. 404–413). Phuket,Thailand: Department of Linguistics, Chulalongkorn University. Descargado de http://www.aclweb.org/anthology/Y14-1047

Utsumi, A. (1996). A unified theory of irony and its computational formalization. En *Coling* (p. 962-967).

Vanin, A. A., Freitas, L. A., Vieira, R., y Bochernitsan, M. (2013). Some clues on irony detection in tweets. En *Proceedings of the 22nd international conference on world wide web companion* (pp. 635–636).

Veale, T., y Hao, Y. (2008). A fluid knowledge representation for understanding and generating creative metaphors. En D. Scott y H. Uszkoreit (Eds.), *Coling* (p. 945-952).

Xu, H., Santus, E., Laszlo, A., y Huang, C.-R. (2015, June). Llt-polyu: Identifying sentiment intensity in ironic tweets. En *Proceedings of the 9th international workshop on semantic*

evaluation (semeval 2015) (pp. 673–678). Denver, Colorado: Association for Computational Linguistics. Descargado de `http://www.aclweb.org/anthology/S15-2113`

Zavala, L. (1992). Para nombrar las formas de la ironía. *Discurso*, *13*, 59–83.

Zhu, X., Kiritchenko, S., y Mohammad, S. M. (2014). Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. *SemEval 2014*, *443*.

Álvarez, A. (1986). Enunciación e ironía. *Archivum: Revista de la Facultad de Filología*, *36*, 77-88.