UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Detección de ironía en textos cortos usando redes neuronales

TESIS

Que para obtener el título de

Licenciado en ingeniería en computación

P R E S E N T A

Max Armando Sánchez Hernández

DIRECTOR DE TESIS

Dr. Iván Vladimir Meza Ruiz

Ciudad Universitaria, Cd. Mx., 2019

Agradecimientos

Agradezco a los involucrados en la realización de esta tesis, a mi asesor Dr. Ivan Vladimir Meza Ruiz, a la Facultad de Ingeniería por la preparación que me dio, a la UNAM por darme la oportunidad de demostrar que puedo realizar mis sueños a base de esfuerzo y dedicación. A mi familia que me apoyo incondicionalmente. A mi hermana Ade que siempre supo que decir para animarme. A mi mamá por criarme y dotarme de criterio propio, por su preocupación que siempre me motivó a no dar el mínimo. A mi padre por ser un bastión de apoyo siempre. A mi hermano Juan por ser un modelo a seguir. A mi novia Diana por ser siempre la compañera incondicional que cualquier persona podría desear. A los errores y caídas que sufrí hasta el momento ya que sin ellas no hubiera aprendido que lo valioso cuesta sudor, lágrimas y sangre, que no hay camino recto al éxito.

Índice general

Capítulo 1

Introducción

La ironía es una figura retórica que consiste en decir lo contrario de lo que se quiere dar a entender, esta definición es aveces difícil de entender para una persona y por lo tanto es de esperarse más difícil de hacer entender a un sistema de cómputo. Hasta este momento podemos darle a una computadora un conjunto de instrucciones específicas, y si logramos definir de algún modo nuestras intenciones, podemos hacer que una computadora entienda de forma concreta un concepto. Sin embargo, la ironía reta a la lógica y es difícil para una persona explicar si una oración es irónica o es algo literal, muchas veces esto depende del contexto. Es cuando entra la inteligencia artificial, que nos ayuda a no explicar cosas que no entendemos bien o que dependen de muchas condiciones, y hace que nuestro sistema computacional salte la barrera de la lógica dotándola de procesos que simulan el razonamiento humano. La inteligencia artificial se ha usado en otras tareas como conducir un automóvil, clasificar objetos, darle significado a las palabras, entre otras.

El objetivo de esta tesis es proponer un modelo de red neuronal que pueda identificar ironía en textos cortos procedentes de Twitter. Dicha solución podría ayudar a los estudios de mercado que buscan la aceptación de un producto mediante el monitoreo de las redes sociales para minar opiniones, incluso en campañas políticas, comerciales o movimientos sociales ya que predeciría la opi-

nión real de una persona sobre cierto tema. A su vez tiene conexión con otros problemas como la búsqueda de significados, minería de opiniones, modelos para detectar contradicciones, entre otros.

El problema antes descrito ha sido explorado por comunidades de científicos e investigadores alrededor del mundo, en la Tabla 1.1 se pueden ver algunos de los trabajos que se han hecho anteriormente:

Investigador/es	Artículo	Métodos
Mihalcea, Strapparava & Pulman	Learning to laugh (automatically): Computational models for humor recognition. Computational intelligence	Naive Bayes SVM
Tsur & Davidov	Semisupervised recognition of sarcastic sentences in twitter and amazon	k-NN
Sounjanya Poria, Erik Cambria, Devamanyu Hazarika	A depper look into sarcastic tweets using deep convolutional neural networks	CNN

Tabla 1.1: Algunos trabajos sobre ironía y/o sarcasmo

Como se ha podido ver, las redes neuronales se han ocupado para esta tarea, sin embargo muchos sugieren que estas implementaciones pueden mejorarse, las redes neuronales se han utilizado en la universidad de Standford para crear el pie de foto de imágenes, Google las ocupa para reconocer los números de las casa en las fotos que toman sus automóviles y ubicarlas en el mapa, en Mountain View las ocupan para mejorar el reconocimiento de voz de Android, ahorrar electricidad en sus servidores, y esto es solo una pequeña parte de sus aplicaciones. Por lo que es una motivación para probar como se desempeñan las redes neuronales en esta tarea.

La descripción del método es la siguiente:

- Obtención del corpus

- Preprocesamiento de los documentos que componen el corpus (embedding, tokenización, normalización, lematización, conversión a vectores), explorar las diversas herramientas que ya existen y destacar la mejor de todas.

- Análisis de la red neuronal que mejor se adapta al problemas, crear un conjunto de caminos viables para elegir los que podrían dar mejores resultados

- Diseño de los experimentos, diseñar las redes neuronales que resolverán la tarea

- Evaluación, elegir las métricas que mejor describan el desempeño del modelo

- Conclusiones, se dará una explicación de los resultados, se analizará las oportunidades de crecimiento, lo que se hizo bien y lo que se hizo mal.

Debido a que las redes neuronales han tenido un desempeño excelente en una gran variedad de aplicaciones es de esperarse buenos resultados, en caso de que los resultados sean negativos sería muy importante revisar los diferentes parámetros de la red neuronal que se pueden cambiar, por ejemplo el número de capas ocultas, el número de nodos de cada capa, la normalización de los datos, la iniciación pseudoaleatoria de los pesos.

Para finalizar la estructura de esta tesis es la siguiente: primero se presentaran algunos antecedentes y el estado del arte de la tarea que se propone, para después explicar un poco de la teoría detrás del método que se usará para resolverla, por último se obtendrán resultados que se compararán con los obtenidos en la sección de antecedentes para terminar con una conclusión y cuales podrían ser los trabajos a futuro.

En este trabajo se usó el recurso de Google, llamado Colab Research, desde el cual se puede usar un entorno de desarrollo en Python 2 o 3, usando como interfaz Jupyter. La ventaja de usar esta herramienta es que se puede acceder al hardware de procesamiento de tensores de Google, Tensor Processing Unit (TPU), los cuales por experiencia propia llegan a ser hasta 20 veces más rápido que usar una GPU. Los experimentos de esta tesis se encuentran en la siguiente liga https://drive.google.com/open?id=1oV5X1ZlOxXT-3nxp89BRwWcmSuSbOm43

Capítulo 2

Antecedentes

La ironía como un recurso lingüístico se destaca entre otras por su facilidad para satisfacer al lector, la ironía da un valor agregado a la comunicación, si aparece como sátira provoca una risa casi instantánea, si aparece como sarcasmo suele ser más hiriente, incluso suele hacernos pensar en las implicaciones de lo que nos atrevemos a decir, como en el siguiente ejemplo:

"(...)En otra ocasión Borges firmaba ejemplares en una librería del Centro. Un joven se acercó con Ficciones y le dijo: 'Maestro, usted es inmortal'... Borges le contestó: 'Vamos, hombre, no hay por qué ser tan pesimista'." Romero (2012)

Este ejemplo ilustra muy bien el impacto de la ironía, de cierta forma maximiza la idea que el autor desea. Este impacto suele aportar una mayor carga de información y por lo tanto más interés en conseguirla. Ésta es una de las motivaciones que han tenido diferentes grupos de estudio para abordar la clasificación de la ironía.

Para hablar de las motivaciones que este problema ha tenido para analizarse podemos empezar por ver el trabajo de Maynard & Greenwood (2014) quienes se orientan por el análisis de sentimientos en redes sociales cuando este recurso es

utilizado; para Utsumi (1995) fue uno de los primeros trabajos que se hicieron en este tema y su motivo para realizar un sistema fue para proveer de herramientas a los sistemas NLP(*Natural Language Processing*) y que pudieran manejar la ironía de manera correcta, este sistema consistía en un conjunto de reglas de emociones; Davidov *et al.* (2010) crean un sistema semi supervisado el cual tiene buen desempeño y resuelve la clasificación en inglés, algunos de sus motivos fue la monitorización de marcas, resumen de reseñas, sistemas de diálogo y proponen un marco de referencia para crear estos sistemas con detección de sarcasmo.

2.1. Estado del arte

El problema de la clasificación de la ironía reside en su característica de comprimir ideas en un pequeño conjunto de palabras, como en el siguiente ejemplo:

Me encanta discutir por las mañanas por dios es algo tan :)

Figura 2.1: Twitter id 631446828127506432

De esta pequeña frase manualmente podemos extraer las siguientes ideas: La persona no está encantada y a la persona no le gusta discutir. En este ejemplo pudimos extraer dos ideas aparte de la idea que por si sola expresa la frase, y en un contexto general ésta no tiene importancia. El siguiente ejemplo expone un caso mucho más complejo:

Empieza un bonito día... Selectividad2015

Figura 2.2: Twitter id 608133042348130304

Esta frase al contrario de la anterior requiere de un contexto para saber que es irónica, ya que la *Selectividad* es un examen que se aplica en España para acceder a la educación universitaria. De esta frase podemos extraer las siguientes suposiciones: La persona se siente nerviosa por el examen, la persona preferiría empezar su día con otra actividad, la selectividad es un examen difícil, etc. En este ejemplo necesitamos de un contexto que nos ayude a intuir hacia donde

va dirigida la idea que se quiere comunicar y a la vez esta idea genera más ideas relacionadas con un contexto general, por ejemplo: *A nadie le gustan las pruebas*, *Las personas le tienen miedo al fracaso*, etc.

La extracción de información es un problema muy complejo y consiste en el estudio de un texto y la inferencia de la idea. En general la ironía interfiere en esta extracción introduciendo ruido a la inferencia, ya que aunque la ironía en forma de sarcasmo representa un pequeño porcentaje según Liu *et al.* (2007) modifica la inferencia por ser una figura lingüística que no puede ser tomada literalmente, aquí como sugerencia puede omitirse o invertirse la polaridad de la idea (si es que esto se quiere obtener). Por ejemplo en la extracción de reseñas podemos tomarlas como negativas, debido que la mayoría de las veces que se usa el sarcasmo (tipo de ironía que consiste en herir) es para dañar o para dar una mala opinión, esto dependerá del área de estudio donde se desee aplicar un modelo de extracción de información. Sin embargo la parte central de esto es la detección de la ironía.

2.1.1. Análisis de los métodos usados

Los métodos que han sido utilizados pueden clasificarse en las categorías que muestra la tabla 2.1

2.1.1.1. Orientados a reglas

En los métodos utilizados están los que requieren de la programación de reglas, estos métodos también son supervisados, pero difieren de éstos ya que requieren una mayor intervención del experto. Los sistemas basados en reglas pueden utilizar varios paradigmas como lógica difusa, lógica booleana, etc. Lo importante de estos métodos es que te permiten encontrar una respuesta clara del por qué esa muestra ha sido clasificada como irónica o no. Para estos métodos suelen utilizarse herramientas como etiquetas Part of speech (POS) que ayudan a la clasificación de las palabras dentro de una clase como adjetivos, verbos, adverbios, sustantivos, etc. Esto sirve, por ejemplo, para dar una descripción de como se puede encontrar una sentencia irónica. Una desventaja de estos métodos es

Clasificación	Artículos
Orientados a reglas	*Formalization and rules for recognition of satirical irony* Kong & Qiu (2011) *How to interpret irony by computer: A comprehensive framework for irony* Utsumi (1995)
Supervisado	*Making objective decisions from subjective data: Detecting irony in customers reviews (POS+features; NB,SVM,DT)*Reyes & Rosso (2012) *The perfect solution for detecting sarcasm in tweets #not (Winnow Classification)* Liebrecht *et al.* (2013) *Humans require context to infer ironic intent (so computers probably do, too) (SVM)* Wallace *et al.* (2014) *Italian irony detection in Twitter:a first approach (DT)* Barbieri *et al.* (2014) *Detecting irony on greel political Tweets: A text mining approach (J48, NB, FT, KStar, Random Forests)* Charalampakis *et al.* (2015) *Character and word baselines for irony detection in spanish short texts*(SVM;RF) Jasso & Meza (2016)
Semi-supervisados	*Semi-supervised recognition of sarcastic sentences in Twitter and Amazon (HFW,CW,KNN-like)* Davidov *et al.* (2010)

Tabla 2.1: Algunos de los métodos que se han usado para clasificar la ironía

que se necesita un experto para la generación de reglas, aunque hay algunos casos de sistemas que pueden generar reglas de manera automática Mitra & Pal (1995) como estos sistemas hay varios. La desventaja es que en estos casos si los sistemas son muy sensibles pueden generar reglas muy generales, que expliquen cosas comunes del idioma e irrelevantes para la tarea, o muy específicas de una oración en particular Kotsiantis *et al.* (2007). Como se puede ver es un método poderoso para la interacción con una persona, y tal vez de más ayuda cuando se quiere modelar una lengua de manera analítica. Sin embargo carece de adaptabilidad, ya que si cambiamos el idioma o el contexto, como lo hacen las redes sociales, todo el sistema debe volver a crearse. Los principales expositores de este método son Kong & Qiu (2011), para el idioma chino y Utsumi (1995), para el idioma inglés.

2.1.1.2. Supervisados

En esta categoría está el método que se propone en esta tesis, los métodos supervisados son aquellos que requieren de datos de entrenamiento etiquetados para poder encontrar el modelo adecuado, los datos deben ser etiquetados por expertos los cuales puedan distinguir a que clase pertenece una muestra, estos métodos suelen necesitar muchas muestras y un tiempo de entrenamiento considerable. La eficiencia del modelo va a depender de la destreza de los expertos para clasificar las muestras por lo que aún es susceptible a fallos. En esta categoría se encuentran métodos que usan POS para encontrar características de las muestras, como lo hace Reyes & Rosso (2012) después de este preproceso se pasa a un clasificador como Support Vector Machine (SVM) , Naive Bayes (NB) y Decision Tree (DT), en el caso de Reyes & Rosso (2012) se extrajeron 6 características de las muestras: n-grams, POS, funny profiling, positive/negative profiling, affective profiling y pleasantness profiling. Una vez con las características y una función de mapeo, que transforme las palabras a este conjunto de características, se puede usar un clasificador como los antes mencionados. Aquí las diferentes aproximaciones se concentran en el idioma ingles, griego, italiano y español, Charalampakis *et al.* (2015), Barbieri *et al.* (2014) y Jasso & Meza (2016) respectivamente, el presente trabajo se centrará en la detección de ironía en español como Jasso & Meza (2016).

2.1.1.3. Semi-supervisados

Los métodos semi-supervisados son aquellos que no requieren la intervención de los expertos. A esta clasificación pertenece únicamente un trabajo previo, el trabajo de Davidov *et al.* (2010) que consiste en dos fases: adquisición de patrones de un conjunto pequeño de muestras etiquetadas del 1 al 5, de nada sarcástico a completamente sarcástico, y la fase de clasificación. Para obtener los patrones del conjunto de muestras etiquetadas se clasifican las palabras en High Frequency Words (HFW) y Content Words (CW), lo cual denota cuales son las palabras más relevantes, dependiendo de un umbral de aparición para la HFW es F_H y CW es F_C, al momento de extraer los patrones se ven más o menos así:

[COMPANY] CW does not CW much does not CW much about CW CW or

Tabla 2.2: Ejemplo de como se ve un patrón en este sistema, como se puede ver los dos ejemplos pueden aparecer simultáneamente en un misma oración, notesé que las palabras [COMPANY] y "."son HFW. Extraído de Davidov *et al.* (2010)

Estos patrones sirven para después convertir las muestras del conjunto no etiquetado, en vectores que describan la posición de ese patrón en un espacio de características. La clasificación es tan fácil como medir la distancia euclidiana, contar cuantos son no sarcásticos y cuantos son sarcásticos en su vecindad, y asignar una clase dependiendo del promedio de su distancia. Así si la muestra está rodeada de muestras sarcásticas será sarcástica completamente y cuando esté entre un conjunto de muestras que algunas son sarcásticas y algunas no, se le asignará una clase porcentual, que indica que tan sarcástica es.

2.1.2. Análisis de los corpus usados

El corpus en general no cambia mucho de una investigación a otra, como muestra la tabla 2.3, la mayoría de los corpus están compuestos por textos provenientes de Twitter ya que provee herramientas simples[1] para extraer tweets automáticamente. También suelen utilizarse corpus de Amazon ya que tienen una Application Programming Interface (API) para extraer textos[2]. Hablando de los idiomas donde se han aplicado está el chino, inglés, holandés, italiano, griego y por último el español, idioma del corpus de este estudio.

Otro ámbito importante de estos experimentos es que consideran un corpus no balanceado, debido a que de acuerdo con Reyes & Rosso (2012), Jasso & Meza (2016) y Barbieri *et al.* (2014), muestran una clara diferencia entre la cantidad de muestras irónicas o sarcásticas de las que no, por lo que nos permitimos inferir que la proporción del uso de la ironía o sarcasmo es de aproximadamente el 12 %. Resulta importante conocer esto ya que nos aporta información sobre la naturaleza del problema y nos ayuda a decidir mejor como es que debemos de abordarlo.

[1] `https://developer.twitter.com/`
[2] `https://developer.amazon.com/services-and-apis`

En la mayoría de las investigaciones que obtuvieron su propio corpus, éste debió pasar por un proceso de normalización donde se removían datos irrelevantes como links de internet, caracteres especiales que no aportan información como '@' y '#' como lo hizo Jasso & Meza (2016). Muchas veces se utiliza un conjunto de palabras que se ignoran llamadas *stopwords* como en el caso de Reyes & Rosso (2012). En adición a esto, algunos autores añadieron el uso de *Term Frequency-Inverse Document Frequency* , que ayuda a medir la importancia de una palabra en un documento, dependiendo de como aparece en el documento y que tanto aparece en el corpus completo, como el caso de Reyes & Rosso (2012), Wallace *et al.* (2015) y Bamman & Smith (2015). Muchas veces los métodos de extracción de información también atrapan documentos repetidos y éstos deben reducirse a mano o crear algún sistema que los detecte y los borré como en el caso de Charalampakis *et al.* (2015) y Reyes & Rosso (2012). En el caso de Ptáček *et al.* (2014) obtuvieron características del texto como los n-grams, etiquetas POS, patrones con HFW, otras como emoticones, signos de puntuación y otras características de los caracteres como el número de mayúsculas, minúsculas, etc., todo esto lo aplicó a un corpus en checo, de 325 muestras sarcásticas y 6,675 no sarcásticos, y otro en inglés, el corpus en inglés balanceado fue de 50,000 cada clase y el no balanceado fue de 25,000 sarcásticos y 75,000 no sarcásticos.

2.1.3. Análisis de resultados

Para hablar de los resultados que otros investigadores han tenido nos referiremos a la tabla 2.4 donde se muestra como la mayoría de los trabajos previos han hecho sus experimentos sobre los datos de Twitter. Además la mayoría han usado un método de aprendizaje supervisado y solo uno semi-supervisado Davidov *et al.* (2010), podemos ver también que la distribución de datos está sesgada a la poca aparición del sarcasmo/ironía, como lo menciona Liu *et al.* (2007) para su corpus de Amazon, esto parece aplicar también para Twitter.

Después de ver más de cerca los resultados que reportan estos investigadores, se puede notar que los mejores resultados parecen ser los de Poria *et al.* (2016) cuando usa CNN combinado con SVM y obtiene un valor de F-score de 0.948 que comparado con la mayoría es más grande, este estudio fue la continuación del trabajo de Ptáček *et al.* (2014) el cual consistía en obtener principalmente

Artículo	Fuente	Idioma	Tamaño
Semi-supervised Recognition of Sarcastic Sentences in Twitter and Amazon Davidov *et al.* (2010)	Twitter & Amazon	Inglés	5.9 millones tweets & 66,000 reseñas
Making Objective Decisions from Subjective Data: Detecting Irony in Customers Reviews Reyes & Rosso (2012)	Amazon, Slashdot & TripAdvisor	Inglés	3,163 → 2,861 reseñas
The prefect solution for detecting sarcasm in tweets #not Liebrecht *et al.* (2013)	Twitter	Holandés	3.3 millones de tweets
Humans Require Context to Infer Ironic Intent (so Computers Probably do, too) Wallace *et al.* (2014)	Reddit	Inglés	3,020 comentarios
Italian Irony Detection in Twitter: a First Approach Barbieri *et al.* (2014)	Twitter	Italiano	25,450 tweets (12.5 %/87.5 %)
Detecting Irony on Greek Political Tweets: A Text Mining Approach Charalampakis *et al.* (2015)	Twitter	Griego	61,427 tweets → 44,438 tweets
Character and Word Baselines for Irony Detection in Spanish Short Texts Jasso & Meza (2016)	Twitter	Español(30/70)	14, 511 tweets irónicos & 33, 859 tweets no irónicos
SOUKHRIA: Towards an Irony Detection System for Arabic in Social Media Karoui *et al.* (2017)	Twitter	Arabe	1,733 tweets irónicos & 3,746 tweets no irónicos1
Sarcasm Detection on Czech and English Twitter Ptáček *et al.* (2014)	Twitter	Checo/Inglés	Checo 325/6,675, Inglés (50/50) y (25/75)

Tabla 2.3: Descripción de los corpus de experimentos anteriores

Investigadores	P	R	A	F1	Detalles
Davidov *et al.* (2010)	0.912	0.756	0.947	0.827	Corpus no balanceado (471/5020), método KNN-like, datos de Amazon
Reyes & Rosso (2012)	0.771	0.725	0.7575	0.747	Corpus balanceado, método SVM, datos de Amazon
Liebrecht *et al.* (2013)	NA	0.75	NA	NA	Corpus balanceado, método Winnow, datos de Twitter
Liebrecht *et al.* (2013)	NA	0.56	NA	NA	Corpus no balanceado (25/75), método Winnow, datos de Twitter
Barbieri *et al.* (2014)	0.75	0.76	NA	0.76	Corpus no balanceado (12/88), método DT, datos de Twitter
Ptáček *et al.* (2014)	NA	NA	NA	0.923	Corpus no balanceado (25/75), método MaxEnt, datos de Twitter inglés
Ptáček *et al.* (2014)	NA	NA	NA	0.582	Corpus no balanceado (325/6675), método SVM, datos de Twitter checo
Poria *et al.* (2016)	NA	NA	NA	0.948	Corpus no balanceado (25/75), método CNN-SVM, datos de Twitter
Jasso & Meza (2016)	NA	NA	NA	0.80	Análisis hecho a nivel carácter con un corpus no balanceado, método SVM, datos de Twitter
Jasso & Meza (2016)	NA	NA	NA	0.80	Análisis hecho a nivel carácter con un corpus no balanceado, método RF, datos de Twitter

Tabla 2.4: Algunos resultados de otros investigadores, P = precisión, R = recall, A = Accuaracy, F1 = F-score

las características más relevantes como los n-grams, etiquetas POS, patrones con HFW, otras como emoticones, signos de puntuación y otras características de los caracteres como el número de mayúsculas, minúsculas, etc., con todo esto se usaron dos métodos SVM y Maximum Entropy Modeling (MaxEnt) el cuál parece funcionar muy bien para el inglés. Sin embargo para el checo no es así ya que SVM tiene mayor puntaje en F-score (0.582) lo cual es muy bajo,

esto según ellos fue por que algunas muestras necesitaban conocimiento general, que con 6,700 muestras no fueron suficientes para obtener, por otro lado en inglés como ya se mencionó sirvió mejor MaxEnt y dio un 0.923 de F-score, esto puede indicar que como Domingos (2012) menciona hay dos formas de mejorar el desempeño de esta tarea, una es conseguir más datos y otra en mejorar o encontrar un algoritmo que requiera menos datos.

Respecto a los demás podemos destacar el trabajo de Reyes & Rosso (2012) quienes principalmente describen una forma de como obtener el corpus y el trabajo de Jasso & Meza (2016) que obtuvo mejores resultados con un análisis a nivel de caracter, con un corpus no balanceado (30/70), tal vez debido a que en Twitter es necesario explotar más el potencial de los caracteres y suelen verse abreviaciones del texto, como por ejemplo “que” pasa a ser “q” cuando el límite de caracteres se ha alcanzado en un tweet.

Capítulo 3

Metodología

Una vez que hemos visto que trabajos se han llevado acabo en la detección de la ironía/sarcasmo, se expondrá el trabajo que se llevó acabo en este estudio de modo que se justifique el diseño de nuestro modelo. Los métodos antes mencionados nos aportan una perspectiva más amplia para poder explorar mejor el problema y llegar a una solución aceptable.

3.1. Obtención del corpus

El corpus que se uso para entrenar nuestro sistema fue extraído de Twitter por Jasso & Meza (2016) y presentado en el artículo *Character and word baselines systems for irony detection in Spanish short texts*, este corpus se distingue de los de la bibliografía ya que es el primero en español, y se basaron en lo explicado por Reyes & Rosso (2012) y Liebrecht *et al.* (2013) en sus respectivos artículos, y consideró que las etiquetas que marcaba explícitamente el usuario de Twitter eran correctas. Como Jasso & Meza (2016) explican, Twitter funciona como una red social en donde se publican ideas concretas que no excedan más de 140 caracteres[1], a esto se le conoce como tweet. En los tweets se se pueden

[1]En noviembre del 2017 se cambio el límite a 280, sin embargo cuando se obtuvo el corpus se tenia como límite los 140

identificar referencias a otros usuarios[1], etiquetas llamadas hashtags (usan # para denotarlos), también se pueden usar links a sitios web externos los cuales solo ocupan 23 caracteres cada uno cuando se usan, además en Twitter cuando se desea ampliar una idea se pueden publicar tweets en serie a los que se les llama *hilo*, y entre otras características como la publicación de imágenes, vídeos,etc.

Para nuestros propósitos se tomará únicamente el texto. Por ejemplo del siguiente Tweet:

Figura 3.1: Ejemplo real de Tweet

Se extrae el texto y se ve en nuestro corpus como lo siguiente:

605233873635508224—0—Las mejores!! Originales nada de imitación. http://link

Figura 3.2: Extracción de únicamente texto

Como se ve en la figura 3.2 el primer número es el número de identificación del tweet, el cual en combinación con el id de usuario se puede obtener la liga al tweet.

[1]Se pueden presentar como @user, donde user es el id del usuario

https://twitter.com/Josaa01/status/605233873635508224

Luego el siguiente número simboliza si el texto es irónico o no, un 0 para no irónico y un 1 para un irónico y por último el texto del tweet.

Para la generación de este corpus se considero que el sarcasmo es un subconjunto de la ironía y se confía absolutamente en la etiqueta que los usuarios proporcionan, aún cuando ésta no sea totalmente fiel a la definición de ironía, cualquiera que quiera considerarse. Pero el corpus de esta tesis no es el mismo que el que usaron Jasso & Meza (2016) ya que después de que ellos lo usaron el corpus se sometió a un etiquetado manual. La distribución del corpus es la sugerida en el mismo texto, 90 % no irónicos y 10 % irónicos. Lo cual ya no los hace directamente comparables.

De este modo se puede ver desde un principio que la definición que se busca lo que convencionalmente se considera como ironía y no la definición rigurosa de ésta. Por lo que a pesar de tratar de atajar una característica general del lenguaje, esta solución está limitada a las condiciones para las que fue entrenada, es decir para Twitter en idioma español y lo que las personas que etiquetaron manualmente el corpus consideraron como irónico.

El corpus se obtuvo mediante la API que Twitter provee para el lenguaje Ruby. Esto diferencia el presente trabajo de algunos otros de la bibliografía. Para obtener los tweets que se usan como background se buscaban diferentes palabras con un significado vació, el cual se proponía no sesgar la búsqueda a un tema específico, estas palabras fueron: donde, quien, como, cuando, este, tiene, porque, dónde, quién, cómo, cuándo, esta, está y por qué. Para obtener los tweets con carácter irónico o sarcástico se uso la búsqueda de *#sarcasmo* y *#ironía*. Adicional a esto se hizo una verificación manual de los tweets que son irónicos y los que no. Se omitieron las referencias a usuarios remplazando el *@user* por @ y las ligas por *http://link*.

Este es el primer estudio que se hace con el corpus etiquetado manualmente, lo que lo diferencia de la gran mayoría de la bibliografía que confiaban que fueran los autores de las sentencias etiquetaran correctamente las sentencias. Esto tiene dos puntos de vista. Por un lado con el etiquetado manual, no se encuentra una

universalidad de lo que es una sentencia irónica, sino que entrenamos al sistema a reconocer las sentencias irónicas que una persona reconoce como tal. Por otro lado si no se etiqueta manualmente, no se tiene un mismo criterio para clasificar a las sentencias, éstas podrían resultar muchas veces contradictorias, por que donde uno ve ironía otro ve enojo, tristeza, etc.

3.2. Preprocesamiento

Una vez obtenido el corpus el primer paso que se debe llevar acabo es la interpretación del texto crudo a un dominio que la computadora pueda manejar correctamente, con esta intención existen muchos enfoques, como se vio en el capítulo 2 y en este estudio se uso el siguiente.

3.2.1. Proceso de preprocesamiento

Con el objetivo convertir las cadenas de texto en objetos que la computadora pueda manejar fácilmente se propuso usar una función hash que funciona como diccionario para las palabras que se han visto en el corpus[1].

Tal como se ve en la figura 3.3 nuestra función que interpreta las palabras a objetos (en este caso se usarán números enteros por simplicidad) es una función hash, la cual necesitará primero un conjunto de palabras que conforman los tweets. Dichas palabras[2] deberán aparecer al menos 5 veces[3]. Si la palabra no alcanza ese número de apariciones se considera como *UNK* que significa *unknown* o desconocida lo cual simboliza que existe un pedazo de información que falta o que tiene un significado desconocido. En este proceso que crea el vocabulario no se detuvieron palabras que generalmente no aportan información valiosa para la clasificación, éstas pueden ser como: de, y, a, o, del, el, etc.

Otro enfoque es considerar como unidad de información los caracteres que con-

[1] Más concretamente en el subconjunto de entrenamiento
[2] Considérese una palabra una subcadena que separa de otras por un espacio, coma o punto
[3] Este valor se fijo experimentalmente ya que no se ignoraban tantas y filtraba las palabras que menos se mencionaban

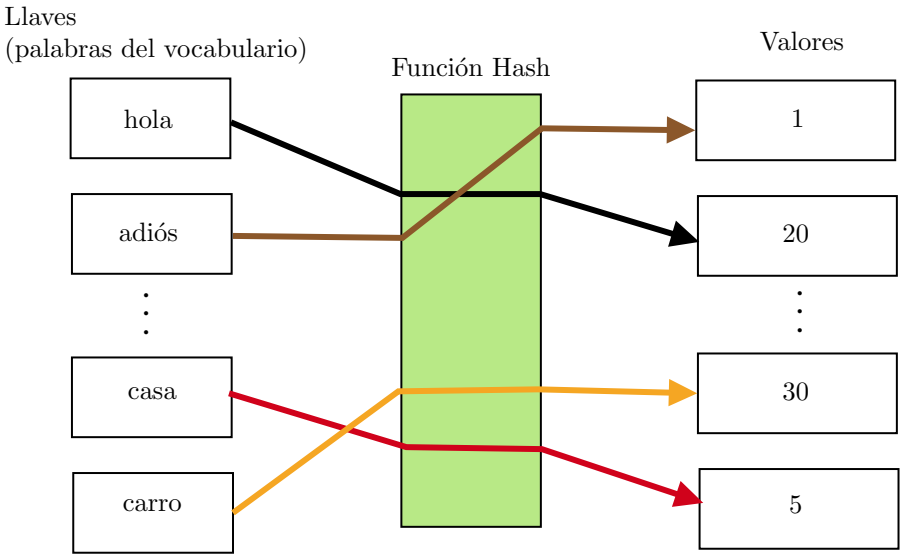


Figura 3.3: Funcionamiento de una función hash(Imagen de autoría propia).

forman la oración. Esto aporta información diferente ya que en lugar de omitir datos que pueden ser considerados basura, se incluyen todos los datos y dejamos que nuestro modelo decida si son útiles o no. Para esto puede aumentarse el tamaño de la unidad mínima, desde un caracter hasta n, este método se le conoce como n-grams.

3.2.2. Diferentes enfoques en el preprocesamiento

Desde la perspectiva de otros investigadores fue más importante considerar un conjunto de reglas que modelará estrictamente lo que es la ironía como vimos con Utsumi (1996) y Kong & Qiu (2011), lo cuál se puede considerar como una buena solución si se desea conocer mejor el lenguaje propio para el cual se hace el modelo, pero ineficiente y poco adaptable a los casos de la vida real donde el lenguaje está vivo y cambia mucho dependiendo del contexto y del uso de las palabras que queramos darle en cierto momento. Nuestro estudio tiene esta ventaja ya que va a adaptarse a lo que algunos usuarios de esta red social han considerado como irónia o no.

Por otro lado algunos investigadores consideraron usar etiquetas POS las cuales dan a las palabras características únicas como la de ser un verbo, adjetivo, adverbio, sustantivo, etc. Estas características resultan importantes para obtener las formulas que describirían un texto sarcástico o irónico, como lo hizo Barbieri *et al.* (2014) y similar a los que hizo Davidov *et al.* (2010). Sin embargo con este acercamiento hay ciertas precauciones ya que es necesario saber como se usa cierta palabra en un momento dado y vienen a la luz dos tipos de problemas. El primero corresponde al uso indistinto de algunas palabras como *'vino'*, cuando se usa como verbo y cuando se usa como sustantivo, este problema podría manejarse tal vez adaptando el árbol de decisiones que se usó en Barbieri *et al.* (2014) para que regrese las diferentes variantes que podría dar como resultado que se considere como verbo o como sustantivo. El segundo problema es cuando se deriva un sustantivo en un verbo, en estos casos no podría entenderse que significado se le puede dar, estos problemas se atajan en nuestra solución debido a no suponemos lo que el usuario quiere decir, con una palabra o que uso quiere darle.

3.2.3. Complejidad del preprocesamiento

La complejidad de este preprocesamiento es lineal ya que debemos leer cada palabra de cada tweet dos veces la primera para obtener las palabras que conforman todo el corpus y contar si esta palabra entrará en el vocabulario y la segunda para asignarle a cada tweet un vector de índices que corresponden al vocabulario que se obtuvo antes.

3.3. Técnica de clasificación

La técnica que se consideró para este problema fue una red neuronal con arquitectura Bidirectional Long Short Term Memory (BI-LSTM) la cual le permite formar un resumen de lo que existe delante de su punto de estudio actual y antes de ese punto. La BI-LSTM se usa en análisis de textos ampliamente, debido a que los textos tienen esta estructura lineal. A continuación se explicará detalladamente en que consiste una red neuronal y posteriormente se describirá

el funcionamiento de un BI-LSTM.

3.3.1. Redes neuronales

En el área de la inteligencia artificial existe una técnica de clasificación inspirada en el funcionamiento del cerebro, para el cual primero se modela la función de una neurona como elemento básico, este modelo fue primero elaborado en 1943 por McCulloch & Pitts (1943) en la que modelaba matemáticamente el funcionamiento de una neurona. El enfoque que ellos tomaban era considerar que las neuronas podían tratarse con lógica proposicional, la cual ha ido evolucionando a través de los años. Rosenblatt (1958) propuso el modelo de un perceptrón simple basado en las ideas de McCulloch & Pitts (1943), la teoría de Rosenblatt se basaba más en un acercamiento probabilístico del funcionamiento de las neuronas en lugar del booleano de McCulloch, este modelo es un perceptron de dos capas, la de entrada y la de salida.

3.3.1.1. Funcionamiento

Este modelo se puede definir como un sistema con entradas y salidas las cuales pasan por una unidad sumadora que las compara con un umbral($bias$) la cual activa una función de salida no lineal. Su funcionamiento es simple y se puede notar que lo que propone es que la salida de un sistema que se puede clasificar es el resultado de la combinación lineal de las entradas con un valor de desplazamiento a la salida($bias$). Luego la función de salida podría interpretarse como booleana un 1 si alcanzó el valor de umbral θ_0 ó un 0 si no. Lo anterior puede interpretarse como que todo lo que este debajo de una recta definida por $(\theta_0, \theta_1, ..., \theta_n)$ se considera que no pertenece a la clase y lo que está por encima de ella pertenece a dicha clase.

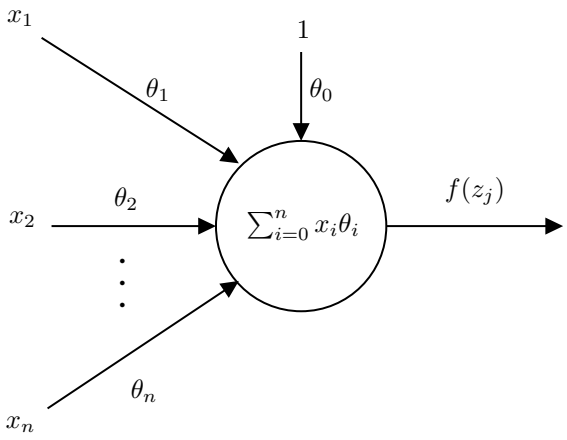


Figura 3.4: Concepto básico del perceptrón simple. (Imagen de autoría propia)

Tiempo después se demostró que no podía resolver problemas no lineales, con una sola capa del perceptrón, como la compuerta lógica XOR.

Entrada A	Entrada B	Salida
0	0	0
0	1	1
1	0	1
1	1	0

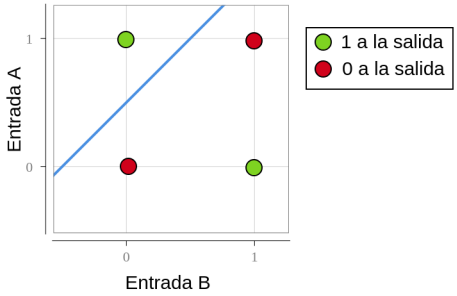


Figura 3.5: No es posible separar con una capa de perceptrón la salida de una compuerta XOR. (Imagen de autoría propia)

Como se ve en la figura 3.5 para poder resolver este tipo de problemas se propuso usar mas capas del perceptrón que podrían resolver el problema fácilmente. Ya que cuando se agregan más perceptrones se pueden *'dibujar'* más lineas que puedan ayudarnos a dividir mejor una clase.

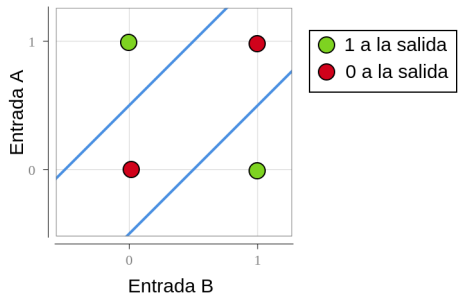
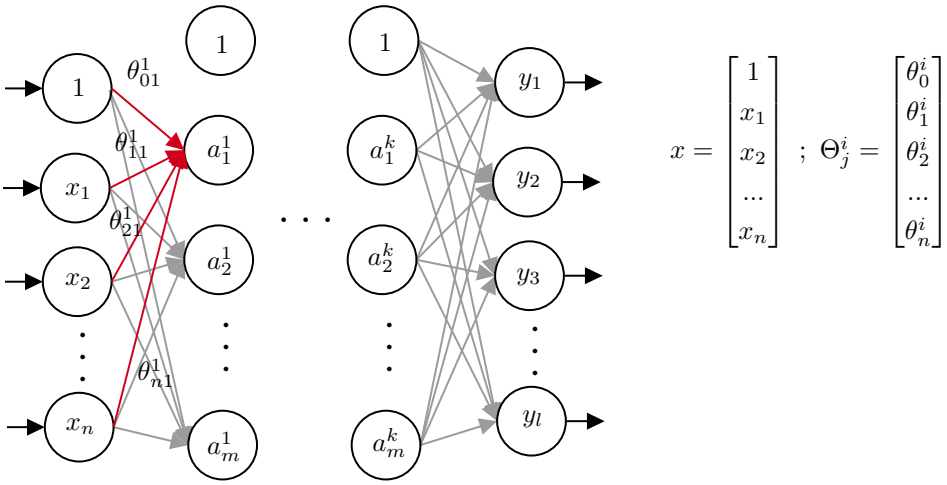


Figura 3.6: Usando una capa más se puede dividir la salida de la compuerta XOR. (Imagen de autoría propia)

Al concepto de red neuronal, como un conjunto de perceptrones interconectados que hacen aún más compleja la salida de un simple perceptrón, se le conoce varios nombres como son *fully connected*, *multi layer perceptron* y red neuronal completamente conectada. Para el caso de cada perceptrón deberán aplicarse las mismas operaciones que se habían descrito para un solo perceptrón, en la figura 3.7 se puede observar como se calcula la salida de la primera neurona de la primera capa.



$$a^1_1 = f\left(1 \times \theta^1_{01} + x_1 \times \theta^1_{11} + x_2 \times \theta^1_{21} + \ldots + x_n \times \theta^1_{n1}\right) = f\left(x^T \Theta^1_1\right)$$

Figura 3.7: Cálculo de la salida de la primera neurona de la primera capa de una red neuronal MLP (Imagen de autoría propia).

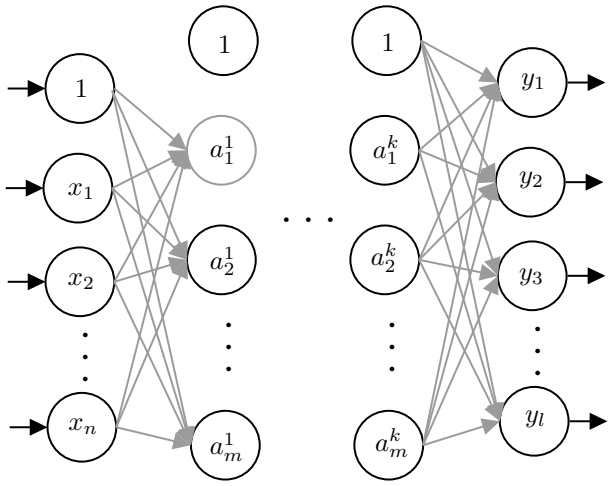


Figura 3.8: Arquitectura de la red neuronal MLP (Imagen de autoría propia).

Si queremos expresar el funcionamiento de una red neuronal de forma matricial se puede hacer lo siguiente, si se observa la arquitectura de la red neuronal de la figura 3.8 se puede ver que las entradas X pueden expresar de forma vectorial igual que las salidas Y y los pesos θ también se puede expresar de forma matricial.

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} \tag{3.1}$$

$$\Theta_j^1 = \begin{bmatrix} \theta_{0j}^1 \\ \theta_{1j}^1 \\ \theta_{2j}^1 \\ . + .. \\ \theta_{nj}^1 \end{bmatrix} \tag{3.2}$$

$$\Theta_j^i = \begin{bmatrix} \theta_{0j}^i \\ \theta_{1j}^i \\ \theta_{2j}^i \\ ... \\ \theta_{mj}^i \end{bmatrix} \tag{3.3}$$

$$A^i = \begin{bmatrix} 1 \\ a_1^i \\ a_2^i \\ ... \\ a_m^i \end{bmatrix} \tag{3.4}$$

$$\Theta^i = \begin{bmatrix} \Theta_1^i & \Theta_2^i & ... & \Theta_m^i \end{bmatrix} \tag{3.5}$$

El superíndice indica la capa a la que pertenece, en el caso de los subíndices de Θ el primer dígito simboliza la neurona de la que procede y el siguiente es a la que entra. A partir de estas ecuaciones se puede ver que para obtener la matriz A de la primera capa se debe seguir la siguiente ecuación:

$$A^1 = F\left(\left(\Theta^1\right)^T X\right) \tag{3.6}$$

Para obtener las matrices A de cada capa se usa la siguiente ecuación:

$$A^i = F\left(\left(\Theta^i\right)^T A^{i-1}\right) \tag{3.7}$$

En estas dos ecuaciones la función F es la función no lineal que se deseé usar para la salida de cada neurona. En algunas de las aplicaciones se usan funciones como Rectified Linear Unit (ReLU) la cual tiene una respuesta como la función escalón, en otras aplicación se usa la función sigmoidal, en otras ocasiones se usa el tangente hiperbólico, en la figura 3.9 se muestran algunas de las funciones más comunes que se usan en las aplicaciones.

Por último la salida vectorial de la red neuronal se calcula usando la siguiente ecuación:

$$Y \ = F\left(\left(\Theta^{k+1}\right)^T A^k\right) \tag{3.8}$$

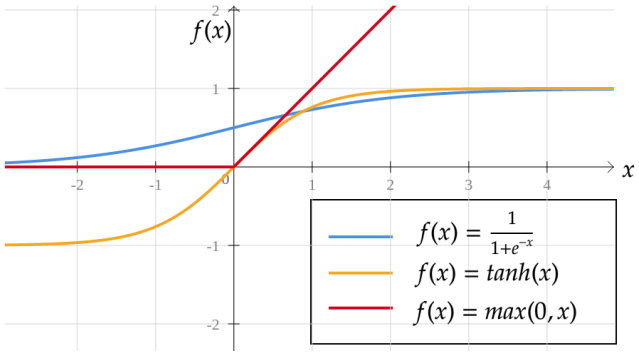


Figura 3.9: Ejemplo de algunas de las salidas de una neurona artificial. (Imagen de autoría propia)

Al proceso que obtiene la predicción en una red neuronal se le llama *feedforward*.

3.3.1.2. Fase de entrenamiento - optimización

La fase entrenamiento de la red neuronal consiste en modificar los pesos θ que corresponden a cada neurona, esto se hace mediante un proceso llamado back-propagation, éste consiste en modificar los pesos de la red de modo que se minimice la función de costo, por medio de la retroalimentación de la red hasta la primera capa. Esta idea fue por primera vez concebida por Rumelhart *et al.* (1986) en su artículo *Learning representations by back-propagating errors*, en el cual se calculaba el error total que se veía a la salida de la red neuronal, el cual se calcula con la ecuación 3.9

$$E = \frac{1}{2} \sum_{c} \sum_{j} (y_{j,c} - d_{j,c})^2 \tag{3.9}$$

donde c es índice sobre las muestras, j es el índice sobre las unidades se salida, y es el valor de salida de la red y d es el valor deseado. Luego entonces si minimizamos este error el desempeño de nuestra red neuronal con respecto al problema será mejor, este error es únicamente para la última capa, para esto es necesario derivar parcialmente la función del error con respecto a los pesos de las sinapsis de entrada de cada una de las unidades, para esto podemos ver el

desarrollo que hizo Rumelhart *et al.* (1986) con respecto a la función sigmoidal.

Se deriva primero el error E con respecto a y_j en cada muestra c, por lo tanto se omite.

$$\frac{\partial E}{\partial y_j} = y_j - d_j \tag{3.10}$$

Esta parcial es únicamente para la última capa, después se verá como calcular el de las siguientes. Después se ve que con la regla de la cadena se puede tener lo siguiente:

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_j} \tag{3.11}$$

En donde y_j está dado por la función sigmoidal en este caso por lo tanto de sustituye la derivada parcial con respecto a x_j.

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{3.12}$$

$$\frac{\partial y_j}{\partial x_j} = \frac{e^{-x_j}}{1 + e^{-x_j}} \tag{3.13}$$

$$\frac{\partial y_j}{\partial x_j} = y_j(1 - y_j) \tag{3.14}$$

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} y_j(1 - y_j) \tag{3.15}$$

En esta parte se puede reemplazar la parcial cuando se usa la función Rectified Linear Unit (ReLU) o la de la tangente hiperbólica. Después se puede hacer de nuevo la regla de la cadena para obtener la derivada parcial de error con respecto a los pesos.

$$x_j = \sum_i y_i \theta_{ji} \tag{3.16}$$

$$\frac{\partial E}{\partial \theta_{ji}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial \theta_{ji}} \tag{3.17}$$

$$\frac{\partial E}{\partial \theta_{ji}} = \frac{\partial E}{\partial x_j} y_i \tag{3.18}$$

Después para conseguir la parcial del error con respecto a x_j se debe hacer lo siguiente:

$$\frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial y_i} = \frac{\partial E}{\partial x_j} \theta_{ji} \tag{3.19}$$

$$\frac{\partial E}{\partial y_j} = \sum_j \frac{\partial E}{\partial x_j} \theta_{ji} \tag{3.20}$$

Con este resultado se puede calcular progresivamente las derivadas parciales del error con respecto a los pesos, en otras palabras la razón de cambio del error con respecto a la modificación de los pesos. Lo cual nos indica hacia donde está el mínimo local. Entonces para encontrar el incremento de los pesos θ podemos aplicar la siguiente ecuación:

$$\Delta\theta = -\epsilon \frac{\partial E}{\partial \theta}$$

En esta ecuación ϵ simboliza el coeficiente de aprendizaje el cual indica que tan rápido se avanzará hacia el mínimo local. Este valor también se puede poner en función del tiempo de modo que a medida que se acerque al valor se vuelve más pequeño el paso. Este método fue primeramente descrito por Rumelhart *et al.* (1986) y tiene una desventaja muy importante, para actualizar una vez los nuevos pesos es necesario computar el error de todas las muestras, por lo tanto tardará un tiempo considerable en terminar una actualización si tenemos muchas muestras, las cuales son necesarias en la mayoría de los casos para encontrar un buen modelo que generalice el problema.

Este algoritmo de optimización tendrá un gran precio computacional, para este propósito se creó el Stochastic Gradient Descent(SGD) el cual para actualizar los pesos en la red neuronal se obtiene el gradiente de un conjunto aleatorio de muestras, y no es necesario computar el gradiente de todo el conjunto de

datos. Sin embargo este método tiene la desventaja que no se encuentra el gradiente general que describirían todas las muestras, sino una aproximación, por lo que el coeficiente de aprendizaje debe ser más pequeño y deberán hacerse más actualizaciones.

Otra opción que surgió con el tiempo fue dotar al cambio de los pesos de un momento esto para que aún cuando había encontrado un mínimo local, el paso pudiera pasarlo por alto con la intención de buscar uno que estuviera en la vecindad, este método no resultó tan bueno ya que podía estancarse en zonas llanas aún más altas que el mínimo local que había ignorado. Luego de este método surgió otro que consideraría una adaptación a las condiciones del gradiente, es decir que en zonas donde fuera necesario que ϵ fuera más pequeño para conseguir converger, pudiera adaptarse, entonces surgió Adagrad. Luego se pensó que se podría hacer variar de igual modo el momento ese fue el surgimiento de Adam, el cual es el método usado en esta tesis, no se pensó en probar con otros ya que Adam ha reportado mejores resultados en la literatura que sus alternativas, este método fue ideado por Kingma & Ba (2014). Sin embargo, como se ha visto en diferentes investigaciones, los mejores resultados se obtienen al usar como optimizador el Root Mean Square(RMS) el cual es un Gradient Descent con momento.

3.3.1.3. Éxito del método

Las redes neuronales han sido ocupadas para muchas aplicaciones hoy en día, sin embargo en sus inicios en 1943 cuando McCulloch & Pitts (1943) investigaba como guardaba información no se pensaba que sería tan utilizado, fue con Rosenblatt (1958) 15 años después cuando apenas se vería un avance significativo con respecto a lo son hoy día, incluso se construyeron proyectos como Mark 1 perceptron. Sin embargo los proyectos que llegaban a ver la luz, apenas podían distinguir entre clases y con un desempeño precario, debido principalmente al poder de computo que es necesario para entrenarlas, fue más bien a partir de los años 2000 con la entrada al mercado y el desarrollo de las API por parte de NVIDIA, AMD e INTEL que la investigación comenzó a tomar el impulso que necesitaba, ahora existen muchas API de desarrollo de sistemas inteligentes como Keras, Tensorflow, Sklearn, etc. con un continuo crecimiento. Hoy en día las

redes neuronales son el algoritmo número uno bio-inspirado que los investigadores usan para hacer sus experimentos (63.04 % en el 2016). Como lo menciona Kar (2016) existen muchos más algoritmos que se pueden explorar más como Artificial Plant Optimization, pero aún por mucho tiempo se prevé que las redes neuronales controlen el mercado y las investigaciones científicas gracias a que aportan resultados confiables y de rápido desarrollo sin la necesidad de conocer rigurosamente las matemáticas detrás de ellas.

Entre los usos que se les han dado están los siguientes: en el artículo *Performance Analysis of Domestic Refrigerator Using Hydrocarbon Refrigerant Mixtures with ANN* de Reddy *et al.* (2019) lo usan para analizar el rendimiento de un refrigerador usando diferentes refrigerantes, en el artículo *Assessing the culture of fruit farmers from Calvillo, Aguascalientes, Mexico with an artificial neural network: An approximation of sustainable land management* de Santos (2019) se buscan soluciones para el cultivo sustentable en la región de Calvillo en Aguascalientes, en el artículo *LED color prediction using a boosting neural network model for a visual-MIMO system* de Banik *et al.* (2018) se usa un modelo de red neuronal que predice el color de un LED, entre otras.

3.3.1.4. Bi-directional Long Short Term Memory

Como se vio en la sección anterior las redes neuronales han servido ampliamente a muchas áreas del conocimiento, sin embargo para cada uno de los problemas se debe usar una arquitectura de red neuronal que sea apropiada y que se justifique que puede resolver el problema. Esta arquitectura no siempre es fácil encontrarla y muchas veces es necesario probar con más de una arquitectura con el fin de comparar resultados y decidirse por una. En otros casos es necesario utilizar modelos de forma secuencial de manera que la salida de uno sea la entrada de otro, como se puede ver, no existe nada escrito en cuanto al diseño que debe tener un modelo para resolver determinado problema. A continuación se explicará un poco sobre la arquitectura que se uso en nuestro problema y la justificación que se tuvo en mente.

Para explicar una BI-LSTM primero debemos abordar ¿Qué es una red neuronal recurrente?¿Qué es una LSTM?

Una red neuronal recurrente es una red neuronal cuyas neuronas tienen una entrada secuencial, por ejemplo una serie de palabras, una serie de caracteres, de imágenes etc. Lo que se busca con esta arquitectura es que las neuronas tengan una forma de retener información sobre los datos que han sido vistos, por lo que se puede decir que posee la capacidad de tener memoria, el modelo de esa neurona es el siguiente:

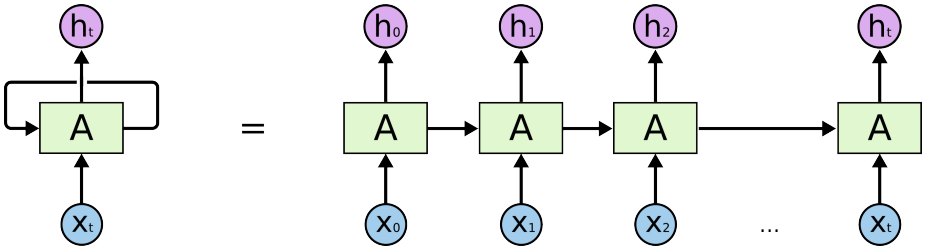


Figura 3.10: Modelo de una red neuronal recurrente. (Imagen extraída de Olah (2015))

El problema principal de que las neuronas guarden toda la información de lo que han visto es que en muchos casos es mucho más relevante la información actual que todo lo que se ha visto anteriormente, por ejemplo si en nuestro conjunto de muestras tenemos un tweet que contiene lo siguiente:

"Ayer me desperté temprano para ir a trabajar y cuando llegué ahí me dí cuenta que era sábado, de regreso a mi casa se me recargo una señora en el metro, gracias reloj #thanks",

En este texto se puede ver que si la red neuronal esta recordando todo lo que ve, entonces ésta "considera" que para identificar algo como sarcasmo es igual de relevante la parte en la que se le recargo una señora que haber ido a su empleo en su día de descanso. Para dicha tarea en la que la información puede o no ser relevante se tiene la Long Short Term Memory(LSTM) la cuales tiene una estructura similar a la siguiente:

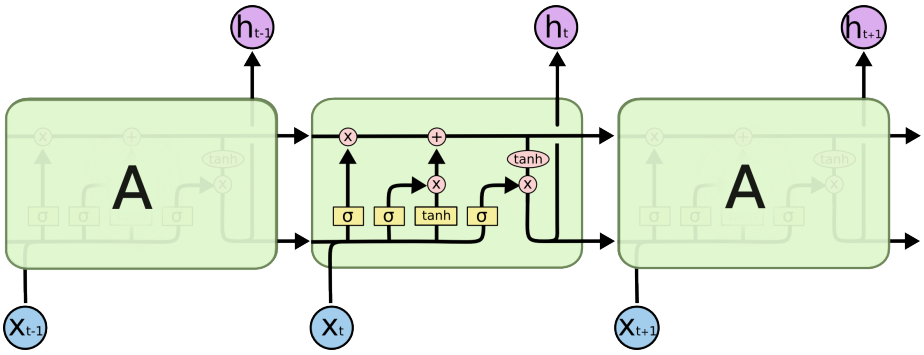
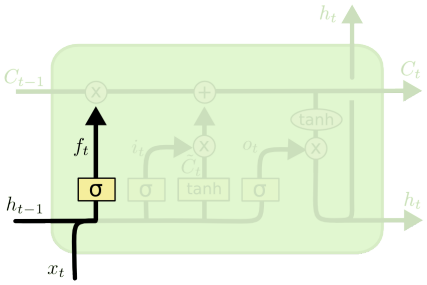


Figura 3.11: Arquitectura de una LSTM. (Imagen extraída de Olah (2015))

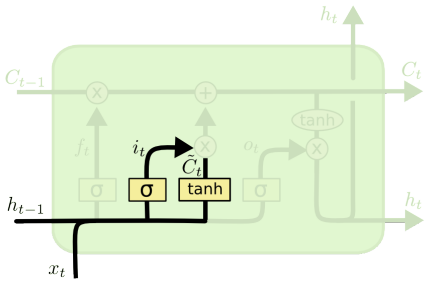
Esta arquitectura tiene diferentes partes, dentro de las cuales se pueden distinguir las siguientes: la parte que decide si olvidar o no, la parte que añade el concepto actual al concepto acumulado y la parte que pasa los parámetros de salida a la entrada de la siguiente unidad.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figura 3.12: Sección de la unidad LSTM que decide si olvida o no.(Imagen extraída de Olah (2015))

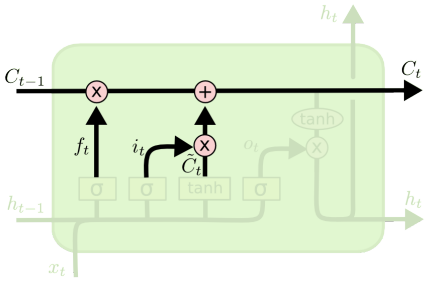
En la figura 3.12 σ es la función sigmoidal que recibe la concatenación del valor que le aporta la unidad anterior y la información que actualmente puede ver, y tiene el propósito de pasar a la unidad que multiplica un valor cercano a 1, que indica que recordará todo lo que ha visto, o un valor 0 para borrar completamente lo que había visto.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 3.13: Sección que decide que valores del vector resumen se actualizarán. (Imagen extraída de Olah (2015))

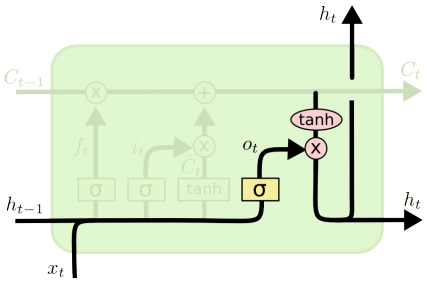
En la figura 3.13 se puede ver como se crean dos vectores uno $\widetilde{C_t}$ y i_t los cuales simbolizan respectivamente los valores candidatos para el resumen y una máscara que filtrará los valores que la red neuronal, después de la retroalimentación, considere importantes. El filtro se aplica cuando se pasa por la unidad que multiplica.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 3.14: Sección que toma los valores que genera el conocimiento que se añadirá al conocimiento anterior. (Imagen extraída de Olah (2015))

En la figura 3.14 se muestra como las entradas de una unidad cambian para convertirse en la entrada de la siguiente unidad, la unidad para multiplicar tiene el propósito de borrar la memoria de la unidad mientras que la suma tiene el propósito de añadir conocimiento a la siguiente unidad.



$$o_t = \sigma\left(W_o\ \left[h_{t-1}, x_t\right]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figura 3.15: Sección que borra o añade el conocimiento al previo. (Imagen extraída de Olah (2015))

En la figura 3.15 se puede ver que en la unidad se genera la salida actual de la unidad h_t que también es el entrada que recibe la siguiente unidad el cual determinará si la siguiente unidad olvidará o no.

Una BI-LSTM cambia el concepto de LSTM para encontrar la información hacia delante y hacia atrás. Añadiendo una capa que lee hacia atrás, entonces ésta se puede conecta a otra capa de red neuronal (completamente conectada) que los convierte en una salida binaria que decidirá si es irónica o no. Para que podamos ver como es la arquitectura de todo el proyecto podemos ver la figura 3.16.

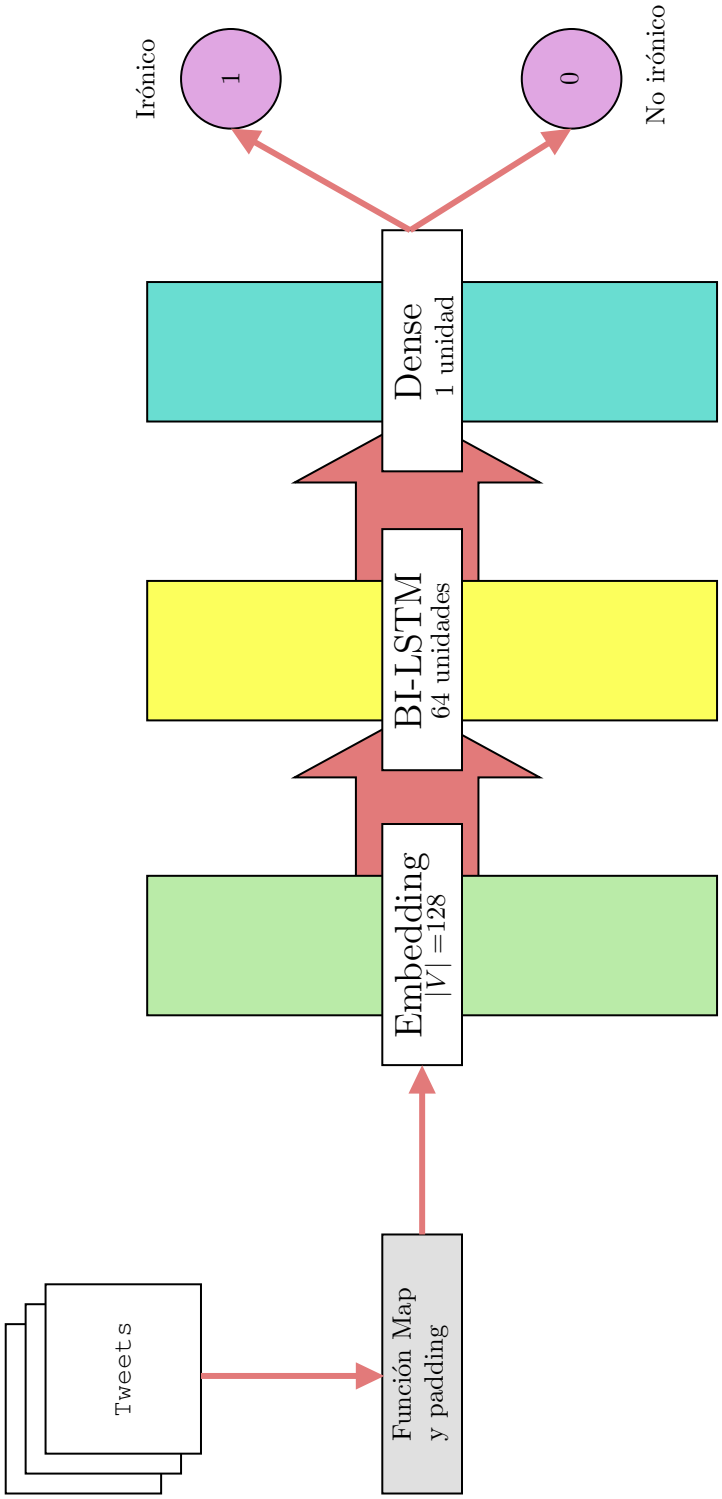


Figura 3.16: En esta figura se puede ver como es la arquitectura general del modelo que se plantea, primero de los textos extraídos de Twitter se pasan por una función de mapeo que convierte el texto en vectores, estos vectores pueden tener tener diferentes longitudes, por lo que pasa por un padding que regulariza su longitud, después pasa al embedding que tiene como salida vectores de 128 dimensiones, estos a su vez pasan a la capa de BI-LSTM que 64 a su salida un vector de 64 unidades, las cuales pasan a una única neurona con salida binaria, la cual indica si el texto que entró es irónico o no.

3.4. Técnica de evaluación

Una de las partes fundamentales de los modelos de inteligencia es la evaluación, en este proceso se mide de manera cuantitativa como se desempeña un sistema. Para esto existen diferentes métricas para cuantificar este desempeño. Para este experimento se usaron *precision*, *recall* y *F-score*.

3.4.1. Precision

La *precision* o precisión en español se refiere a una medida de desempeño que se usa principalmente en tareas no balanceadas, esta medida trata de reconocer cuantas predicciones positivas fueron correctas entre el total de las respuesta correctas, esto se refiere principalmente a que tan bien funciona nuestro modelo en detectar nuestras muestras relevantes(en este caso las positivas). La fórmula es la siguiente:

$$Precision = \frac{TP}{TP + FP}$$

TP = True positive (muestras que se predicen positivas y que lo son) , FP = False Positive(muestras que se predicen negativas y que lo son)

3.4.2. Recall

El *recall* o reclamo en español se refiere a la medida de desempeño de un modelo que igual que la precisión se usa en tareas con datos no balanceados. Su interpretación índica cual es la porción de muestras que se predicen positivas y lo son, entre el total de las muestras que son positivas, su fórmula es la siguiente:

$$Recall \ = \ \frac{TP}{P} \ = \ \frac{TP}{TP + FN}$$

TP = True positive (muestras que se predicen positivas y que lo son) , FP = False Positive(muestras que se predicen negativas y que lo son), P total de medidas que son positivas

3.4.3. F-score

El *F-Score* es una medida del desempeño que combina el *recall* y el *precision*, para obtener la media armónica, esto es por que las dos esta relacionadas y si una sube la otra baja. Lo importante es tener un equilibrio el cual haría el *f-score* más grande. Una media armónica que trata de no sesgar el resultado de la media al valor más grande sino al mas bajo. La fórmula del f-score es la siguiente:

$$F - score \ = \ 2 \times \frac{precision * recall}{precision \ + recall}$$

3.4.4. Descripción de la forma de evaluación cruzada

Para la forma de evaluación el corpus se subdividió en 5 partes del 20 % cada una, de las cuales se formaron 5 distribuciones del corpus, esto se puede ver mejor en la figura 3.17.

Sobre estos 5 grupos se entrenó el mismo clasificador usando el 80 % para entrenamiento y el 20 % para la fase de evaluación. Los datos que se ven en el capítulo 4 se promedian y se reportan.

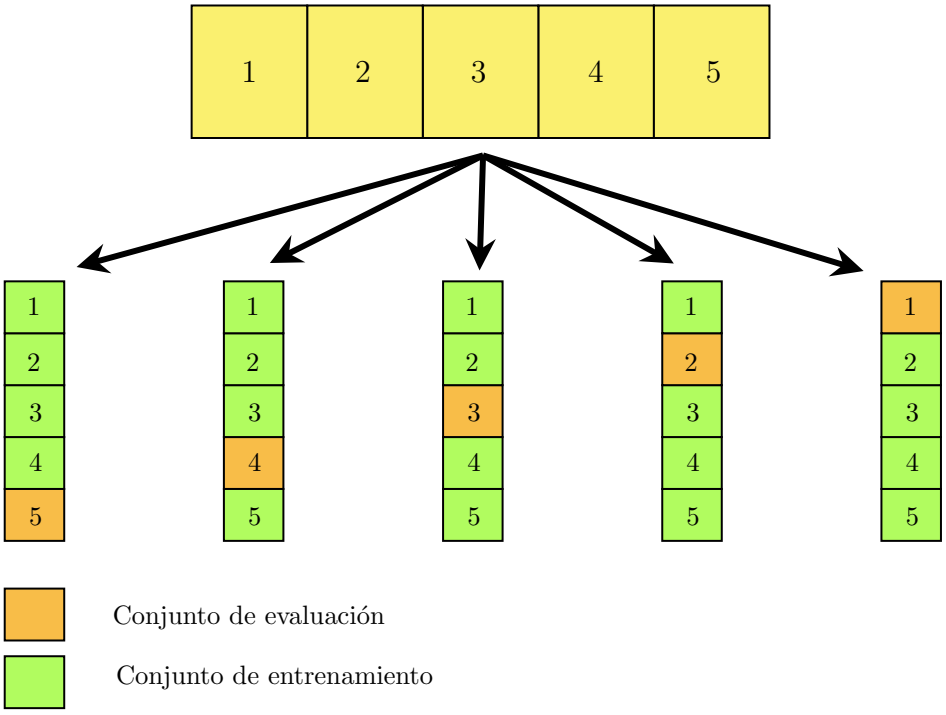


Figura 3.17: Del corpus total se separaron 5 versiones las cuales cambian su sección de prueba y de entrenamiento, para después promediar sus puntajes.

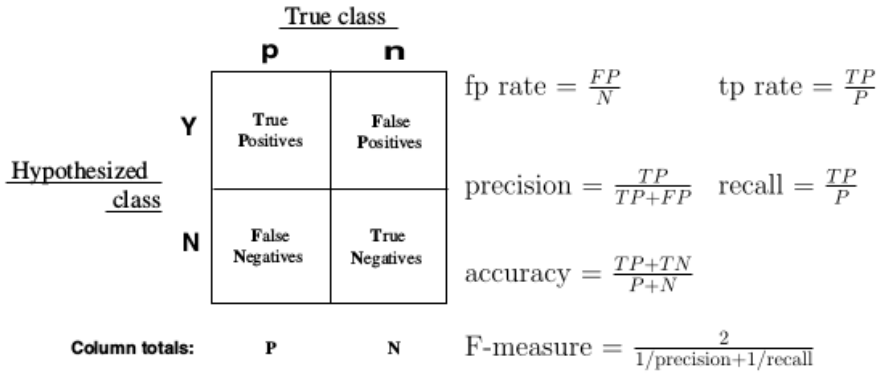


Figura 3.18: Matriz de confusión. Imagen extraída de Fawcett (2006)

3.4.5. ROC y AUC

Otra forma de evaluación que se realizará es la métrica ROC y AUC. Una gráfica *'receiver operating characteristics (ROC)'* es un método gráfico para calificar el desempeño de un clasificador.

En nuestro caso consistirá desplazar el umbral de discretización para observar como se da el aumento de los *true positive rate* y *false positive rate*. Esta gráfica es bastante sencilla, sin embargo su interpretación no es nada trivial, por lo que no se tratará a fondo en esta tesis.

Esta métrica nos ayuda a visualizar fácilmente que pasa con el comportamiento de un clasificador cuando se le cambia el umbral. Para entenderla se debe analizar una curva ROC muestra.

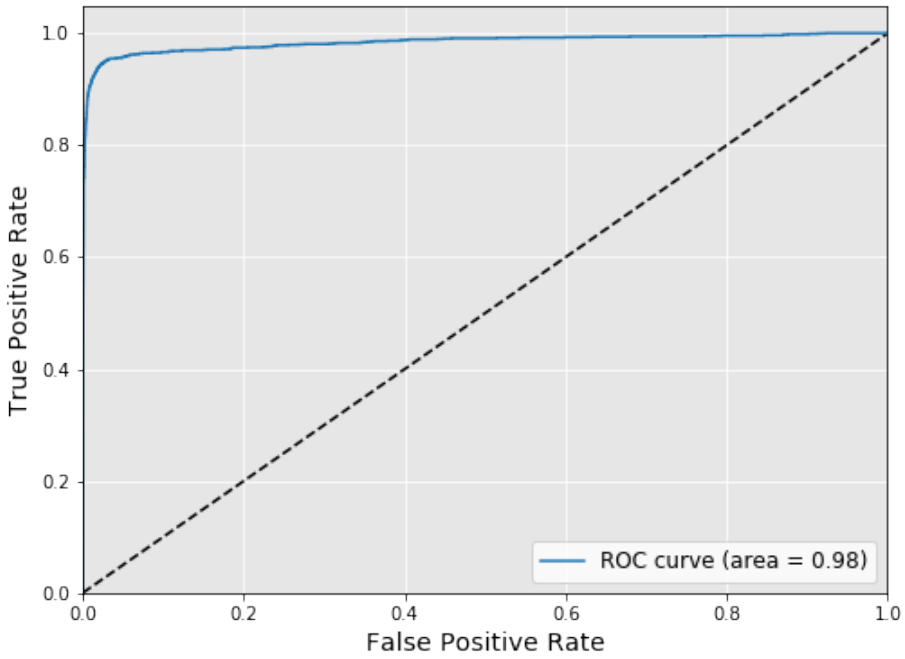


Figura 3.19: Curva ROC experimento 3

En 3.19 se puede observar una recta a 45° la cual es la recta que se obtendría si se hace una clasificación aleatoria. Luego la recta azul es la recta que se obtuvo de mover el umbral de 0 a 1. Se Puede ver que cuando el umbral es 0, todas las muestras serán positivas entonces nuestro *false positive rate* y el *true positive rate* aumentarán a 1 o lo que es lo mismo se tendrá un punto en (1,1). Por el lado contrario si se tiene el umbral en 1 el *false positive rate* y el *true positive rate* disminuirán a 0 y se tendrá un punto en (0,0). Lo que buscamos con nuestro modelo es tener el *true positive rate* en 1 y el *false positive rate* en 0 o lo que es lo mismo tener un punto en (0,1). Como es muy probable que nuestro modelo tenga dicho punto, se puede considerar que entre más cerca mejor y esto se logra si tenemos bien separadas nuestras clases. Es decir, cuando tenemos una clase en un extremo de nuestro umbral y la otra del otro, de este modo el movimiento de nuestro umbral no afectará tanto en la clasificación. Esto es lo que mide la gráfica ROC.

Hablado ahora del *area under the curve* es la reducción de la gráfica ROC a un

escalar el cual es representado por el área bajo la curva, esto nos dirá que tanto nos pudimos acercar al punto deseado (0,1) y que tan estable es nuestro modelo al clasificar.

3.4.6. Justificación

El *recall* y el *precision* son las medidas que generalmente se usan para medir el desempeño, principalmente por que aportan información sobre que tan bien realizan la tarea cuando los datos no están balanceados, lo cual sucede la mayoría de las veces. Además el *f-score* nos aporta una mejor utilidad ya que aporta información de ambas medidas, sin tender a sesgar el resultado por el valor más grande sino por el más pequeño.

La gráfica ROC y el AUC son métodos gráficos ampliamente usado que pueden servir para rápidamente identificar si un modelo tiene un buen desempeño o no. Por lo que se considera valioso para los experimentos.

Capítulo 4

Experimentos

El modelo propuesto de clasificación de ironía se implementó en Python. Este capítulo consiste en presentar, comparar y dar una explicación sobre por que salen dichos resultados, para cada uno de los experimentos se probó con diferentes configuraciones para compararlos entre ellos, con el objetivo de encontrar la que nos presente mejores resultados.

4.1. Primer experimento

En este primer experimento se probó con una arquitectura con una primera capa de embedding de 128 unidades de salida, seguida de una capa BI-LSTM de 200 unidades, después una capa de dropout con parámetro de 0.5, seguida de una capa Dense con una única salida que representa el resultado binario que indica si la oración es irónica o no. Este experimento se distingue de los demás ya que no tomamos en cuenta los caracteres especiales, únicamente el texto el cual se normaliza a ASCII con el propósito de no darle tanta importancia a la ortografía. Se agrega al final un caracter que delimita si es el final de la sentencia.

4.1.1. Evaluación

	Accuracy	Precision	Recall	F1-score
Subdivisión1	0.9698	0.9277	0.7589	0.8349
Subdivisión2	0.9671	0.8422	0.8246	0.8333
Subdivisión3	0.9692	0.9045	0.7720	0.8330
Subdivisión4	0.9677	0.9569	0.7097	0.8150
Subdivisión5	0.9698	0.9146	0.7694	0.8358
Promedio	0.9687	0.9092	0.7669	0.8304
Desviación estándar	0.0011	0.0378	0.0366	0.0078

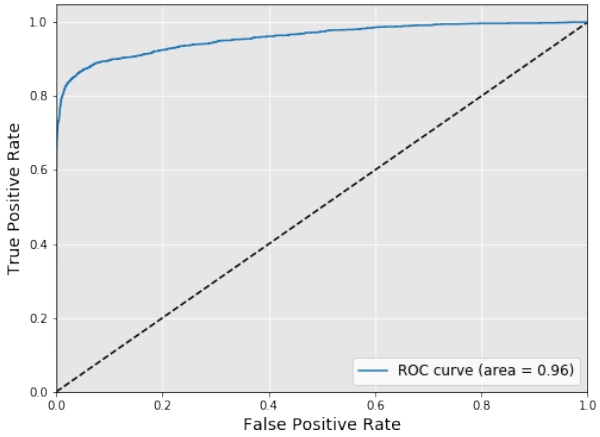
Tabla 4.1: Tabla de métricas experimento 1.

Como se puede apreciar en la tabla 4.1 la precisión medida sobre el modelo, es similar a la obtenida en la bibliografía, lo cual indica que este modelo se desempeña bastante bien en reconocer la ironía. Sin embargo como vemos, el *recall* es considerablemente bajo, por lo que este modelo tiende a no encontrar un gran porcentaje de las muestras irónicas. Adem´a se puede ver que el *F-score* es bastante similar a los de la bibliografía.

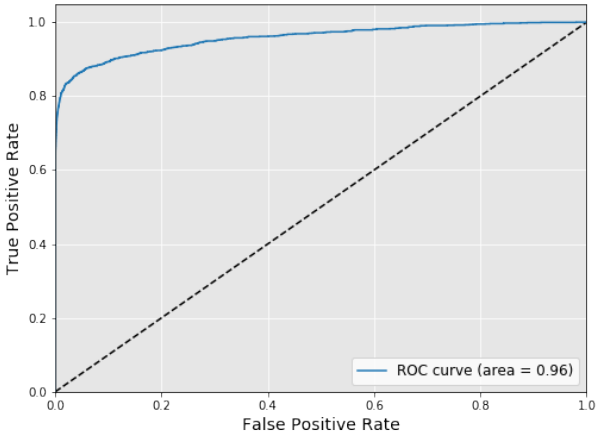
A continuación en la figura 4.1 se muestran algunas de las gráficas ROC que muestran que tan bien se desempeña esta arquitectura y preprocesamiento.

4.1.2. Propuesta de mejora

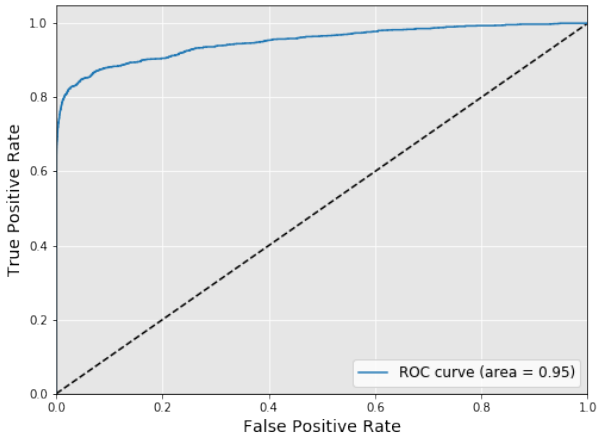
Mi propuesta de mejora es que en lugar de considerar las palabras como tokens, se consideren los caracteres. Esto podría mejorar el desempeño ya que dejaríamos que la red neuronal aprendiera de la forma que escriben los usuarios y no se esforzaría tanto por encontrar correlaciones con palabras que no se encuentran en una oración. Similar a como los humanos leemos las palabras, ya que nosotros podemos extraer características diferentes de dos textos que tienen un contenido similar por ejemplo: ‘hola’ y ‘holaaaaaaa’ en la primera podemos notar que es ortográficamente correcta. No obstante no da mucha información



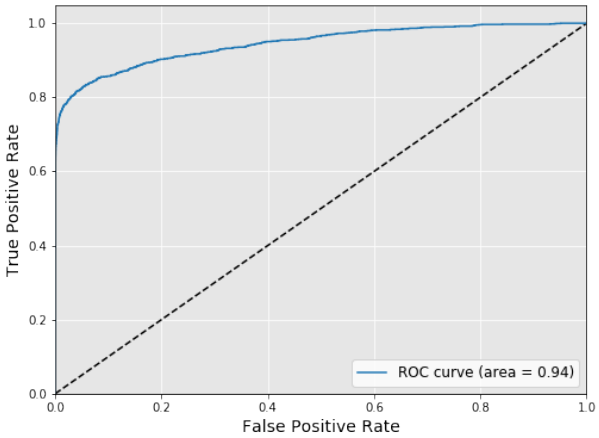
Segmento 1



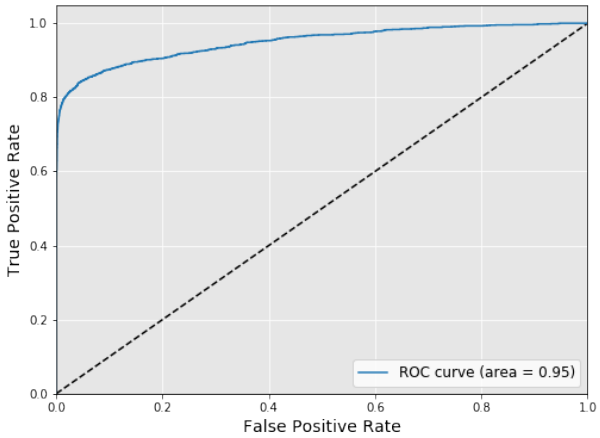
Segmento 2



Segmento 3



Segmento 4



Segmento 5

Figura 4.1: Gráficas ROC de los segmentos en que se dividió el corpus. Primer Experimento preprocesamiento por palabra.

por si misma, con la segunda podemos ver que no es ortográficamente correcta, por lo que muy probablemente no este en el vocabulario que se extrajo del conjunto de entrenamiento. Si la palabra tiene los últimos caracteres repetidos puede indicar que se quiere hacer algún tipo de énfasis, o que el emisor quiere señalar algo evidente. Estos son ejemplos de características que se pueden obtener si se hace un análisis caracter por caracter y la principal motivación para el siguiente experimento.

4.2. Segundo experimento

En este segundo experimento se quiso cambiar el preprocesamiento, realizando lo más sencillo, pasar el tweet a una lista de caracteres y considerar cada uno como unidad de información. Entonces el vocabulario sería cada uno de los caracteres que se encuentran presentes en el corpus distinguiendo si eran mayúsculas o minúsculas. Tampoco se excluyeron los caracteres especiales, para de este modo captar la mayor información posible.

4.2.1. Evaluación

	Accuracy	Precision	Recall	F1-score
Subdivisión1	0.9840	0.9450	0.8928	0.9181
Subdivisión2	0.9842	0.9646	0.8737	0.9169
Subdivisión3	0.9811	0.9477	0.8568	0.8999
Subdivisión4	0.9801	0.9562	0.8402	0.8944
Subdivisión5	0.9833	0.9623	0.8674	0.9124
Promedio	0.9825	0.9552	0.8662	0.9084
Desviación estándar	0.0016	0.0078	0.0175	0.0095

Tabla 4.2: Tabla de métricas experimento 2.

En la tabla 4.2 se puede notar un mejora considerable ya que pasa de tener un

83.04 % de *F-score* promedio a un 90.84 % . Esto es debido a que se pudieron obtener mejores características de las secuencias de caracteres. A continuación en la figura 4.2 se muestran algunas de las gráficas ROC que muestran que tan bien se desempeña este modelo.

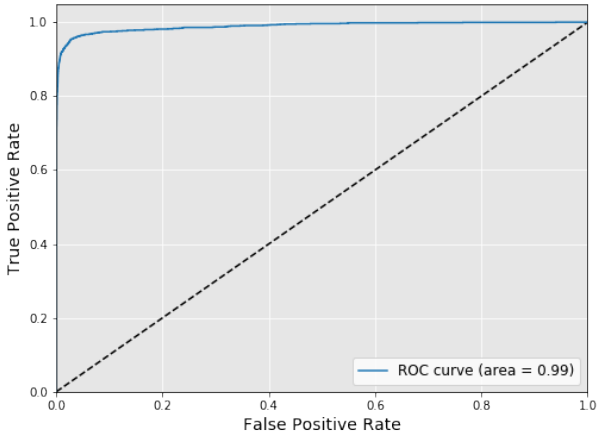
Se puede notar que los segmentos de este experimento están siempre tiene un AUC por encima del máximo del experimento 1. Lo cual indica directamente que es un mejor modelo.

4.2.2. Propuesta de mejora

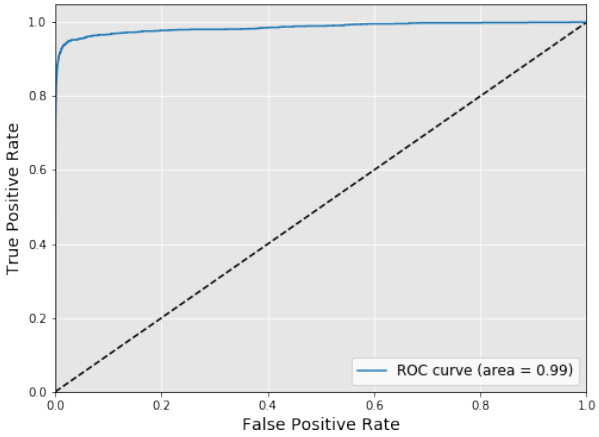
Al analizar los resultados obtenidos se pudo observar una mejora considerable, tanto el *recall* como en *precision* lo que indica que este modelo es mucho mejor que el primero. Esto puede ser debido a que en las redes sociales las reglas gramaticales y de ortografía son más flexibles, y para poner un ejemplo: escribir "hola ¿Que haces?" es lo mismo que decir "ola k ase", semánticamente hablando, por lo que analizando palabra por palabra la similitud entre estas palabras es pequeña mientras que caracter por caracter es más grande debido a la red neuronal puede encontrar secuencias de caracteres que realmente importa como 'ola' y también encontrar una correlación entre 'que' y 'k' lo cual significaría que se adapta al uso de la lengua.

El *recall* como se explicó previamente es la medida que nos indica que porcentaje de muestras relevantes es encontrada. Los números nos están diciendo que en el experimento 1 de 100 muestras irónicas, pudo distinguir 76 y en el experimento 2 pudo distinguir 86. Mientras que el *precision* es la medida que nos indica que porcentaje de las muestras clasificadas como irónicas fue de verdad irónica. Por lo tanto en números el experimento 1 detectó 100 muestras irónicas, de las cuales el 90.92 % fue de verdad irónica y el experimento 2 de 100 detectó 95.52 % irónicas. En ambas métricas el segundo experimento es mucho mejor.

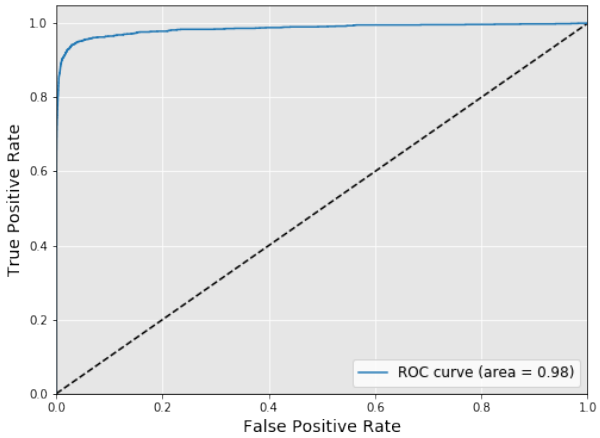
Con el fin de mejorar este resultado se propone aumentar el tamaño del n-gram a dos. Esto aportaría mayor información de la vecindad de los caracteres, pudiendo encontrar mejores patrones que describan más globalmente la esencia de los datos.



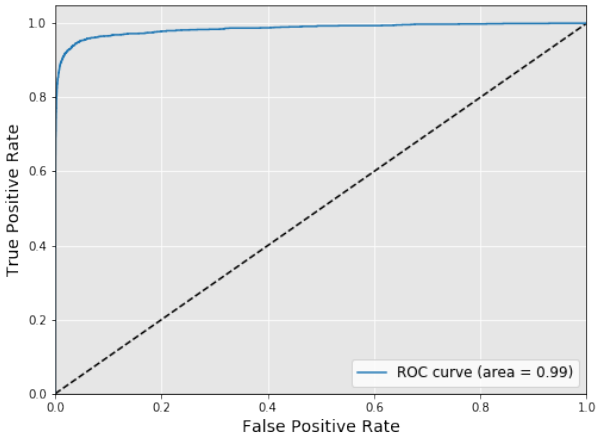
Segmento 1



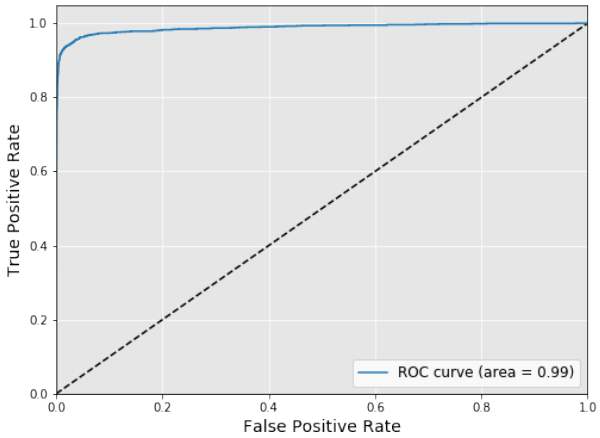
Segmento 2



Segmento 3



Segmento 4



Segmento 5

Figura 4.2: Gráficas ROC de los segmentos en que se dividió el corpus. Segundo experimento por caracteres n-gram es 1.

4.3. Tercer Experimento

En este experimento se incrementó el tamaño de los tokens ahora serán tomados por parejas de caracteres. Por lo que algunos tokens podrían ser los siguientes: 'ho', 'ol', 'la'. Lo que se busca con este experimento es obtener más flexibilidad que cuando procesa por palabra y mayor rigidez que cuando procesa por caracter. Los resultados fueron los siguientes:

4.3.1. Evaluación

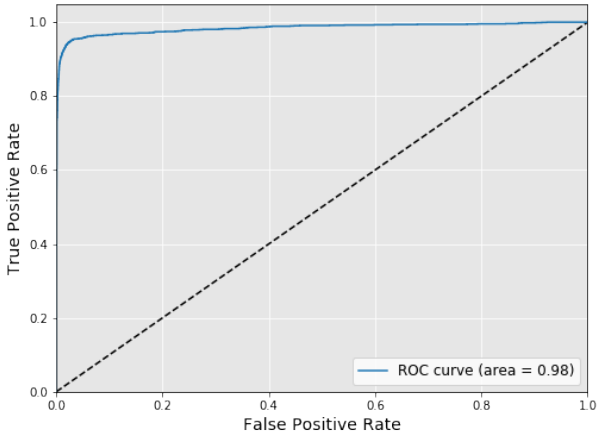
	Accuracy	Precision	Recall	F1-score
Subdivisión1	0.9828	0.9311	0.8954	0.9129
Subdivisión2	0.9842	0.9362	0.9031	0.9194
Subdivisión3	0.9824	0.9076	0.9159	0.9117
Subdivisión4	0.9831	0.9436	0.8845	0.9131
Subdivisión5	0.9850	0.9670	0.8798	0.9213
Promedio	0.9835	0.9371	0.8958	0.9157
Desviación estándar	0.0010	0.0192	0.0130	0.0039

Tabla 4.3: Tabla de métricas experimento 3.

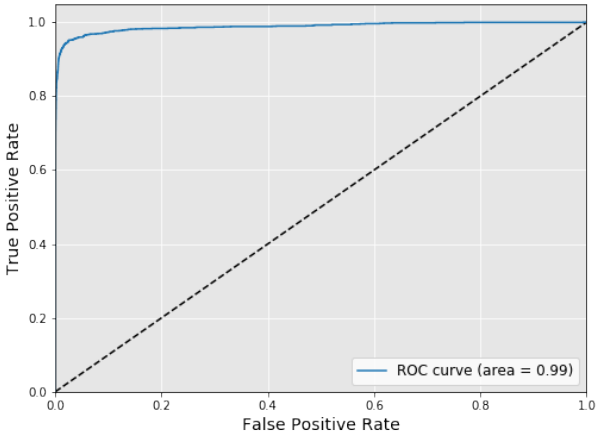
En la tabla 4.3 se puede ver una mejora leve con respecto al experimento anterior con un promedio de *F-score* de 91.57 % un aumento de casi 1 %. A continuación en la figura 4.3 se muestran algunas de las gráficas ROC que muestran que tan bien se desempeña este algoritmo.

4.3.2. Análisis

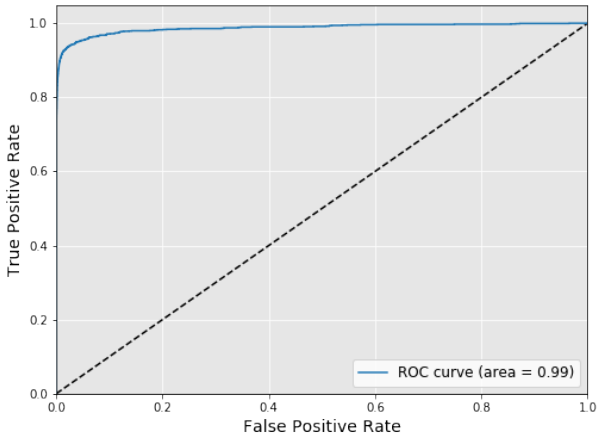
En este experimento se pudo notar una mejora leve con respecto a los resultados previos. Se pudo ver que los resultados de *accuracy*, *recall* y *f-score* mejoran con respecto al segundo experimento, lo que nos indica directamente que es un



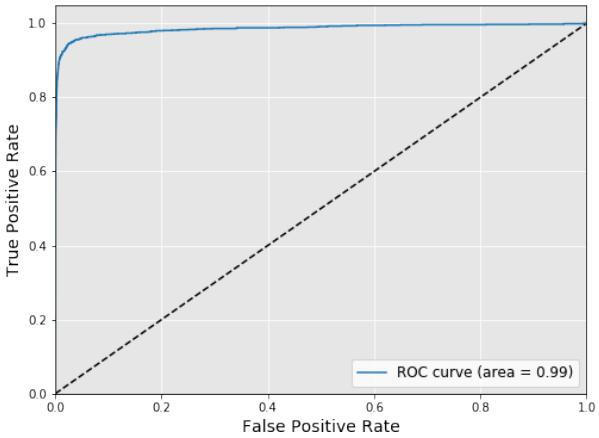
Segmento 1



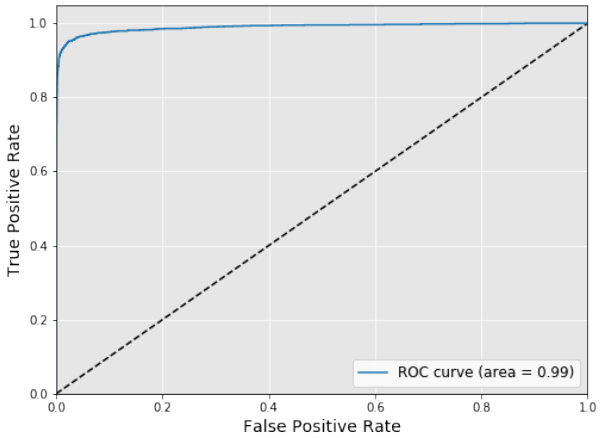
Segmento 2



Segmento 3



Segmento 4



Segmento 5

Figura 4.3: Gráficas ROC de los segmentos en que se dividió el corpus. Tercer experimento por caracteres n-gram es 2.

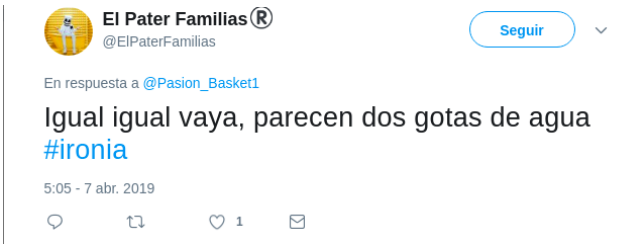


Figura 4.4: Texto irónico tomado a prueba. En esta muestra se obtuvo un 0.985333 de porcentaje de ironía. Para el experimento 3.

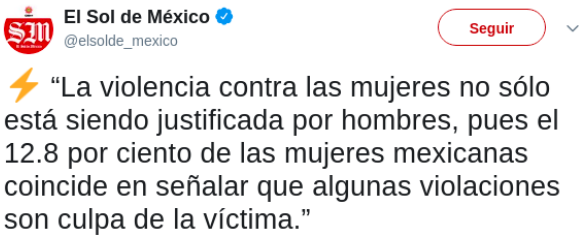


Figura 4.5: Texto no irónico tomado a prueba. En esta muestra se obtuvo un 0.003657 de porcentaje de ironía. Para el experimento 3

mejor modelo. Sin embargo la única métrica que no mejora es la *precision* lo que indica que de las muestras que reconoce como irónicas el segundo experimento tiene menores probabilidades de equivocarse. Sin embargo también tiene menores probabilidades de encontrar las muestras irónicas (*recall*). Y si consideramos el promedio de las dos métricas, se comporta mejor este modelo que el primero.

Como propuesta para siguientes experimentos, puede aumentarse el tamaño del n-gram para tratar de encontrar un punto medio que mejore estas métricas, aunque probablemente ya se haya encontrado o esté por encontrarse.

En la figura 4.4 se puede ver que es una muestra irónica que tiene un alto porcentaje de predicción y en la segunda figura 4.5 se puede ver que no es un texto irónico y presenta un bajo porcentaje de predicción. En estas ejemplos se puede ver como se la predicción que hace el sistema.

4.4. Cuarto Experimento

En este experimento se probo aumentar el tamaño de la palabra de n-gram, esto con el fin de obtener una mayor flexibilidad que usando como tokens las palabras y una mayor rigidez que cuando se usa n-gram de dos.

4.4.1. Evaluación

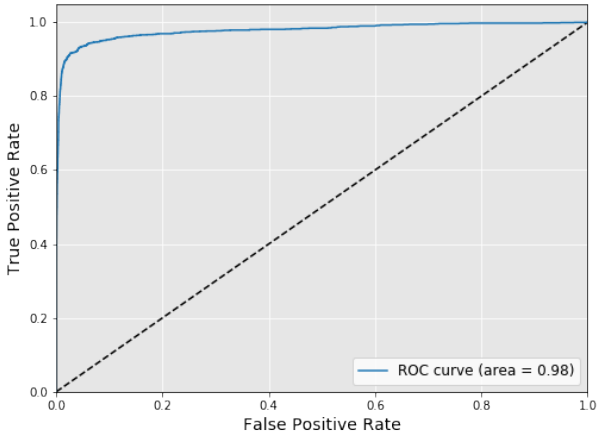
	Accuracy	Precision	Recall	F1-score
Subdivisión1	0.9754	0.8687	0.8895	0.8790
Subdivisión2	0.9294	0.5959	0.9110	0.7205
Subdivisión3	0.8768	0.4436	0.9409	0.6029
Subdivisión4	0.9364	0.6259	0.9080	0.7410
Subdivisión5	0.9507	0.7014	0.8837	0.7821
Promedio	0.9338	0.6471	0.9066	0.7451
Desviación estándar	0.0325	0.1389	0.0200	0.0896

Tabla 4.4: Tabla de métricas experimento 4.

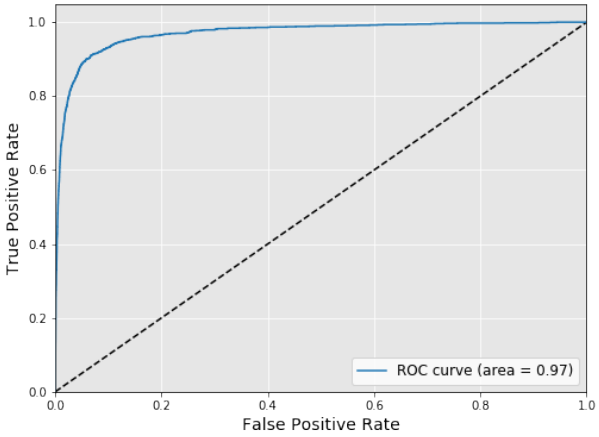
Se puede notar en la tabla 4.4 que el *f-score* es el peor de todos los experimentos, lo cual indica que el equilibrio que se buscaba al aumentar el n-gram ya se había alcanzado con el tercer experimento.

Sin embargo un punto a recalcar en este experimento es que su medida de *recall* es la mejor de los 4 experimentos. Lo cual sería importante si fuera uno de esos problemas en lo que es más importante predecir la mayor parte de las muestras positivas aunque haya muchas veces falsas alarmas, como en sismos, desastres naturales, etc.

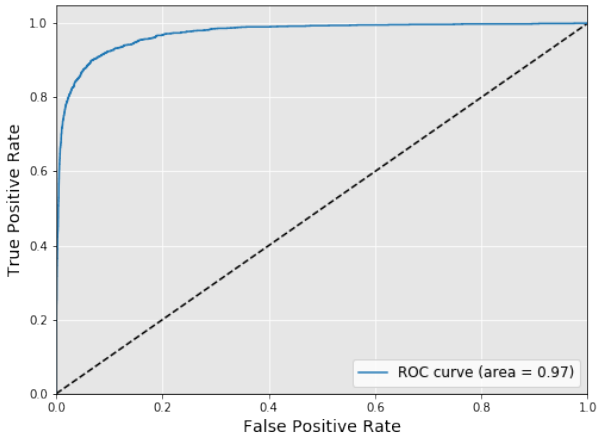
A continuación en la figura 4.6 se muestran algunas de las gráficas ROC que muestran que tan bien se desempeña este algoritmo.



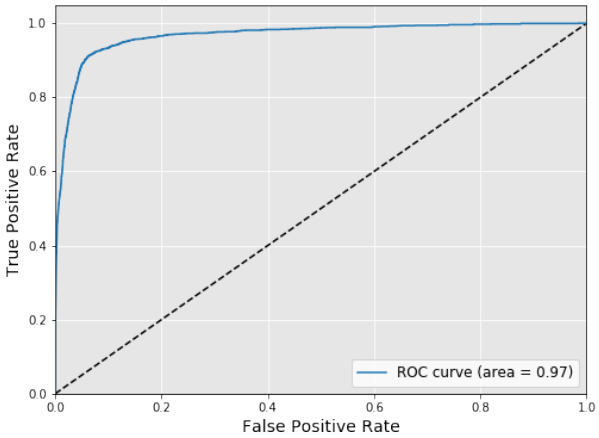
Segmento 1



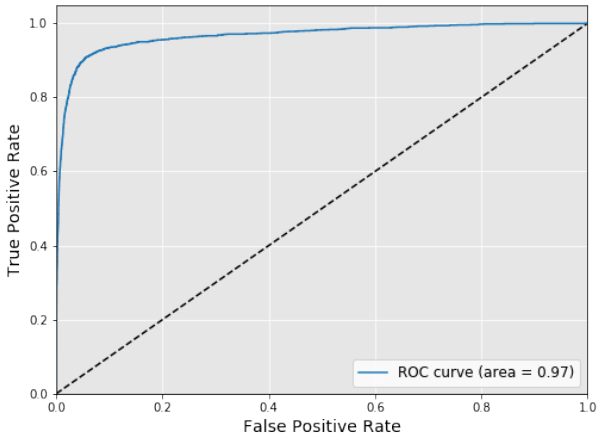
Segmento 2



Segmento 3



Segmento 4



Segmento 5

Figura 4.6: Gráficas ROC de los segmentos en que se dividió el corpus. Cuarto experimento por caracteres n-gram es 3.

Figura 4.7: Texto medianamente irónico tomado a prueba. En esta muestra se obtuvo un 0.356429 de porcentaje de ironía. Para el experimento 4.

Figura 4.8: Texto no irónico tomado a prueba. En esta muestra se obtuvo un 0.003036 de porcentaje de ironía. Para el experimento 4

En estas gráficas se puede ver que son peores que la de cualquier experimento, pero siguen teniendo una buena área bajo la curva, lo que indica que es un experimento bastante bueno.

Para concluir este capítulo se muestra una tabla que compara directamente las diferentes medidas obtenidas en esta fase de experimentos.

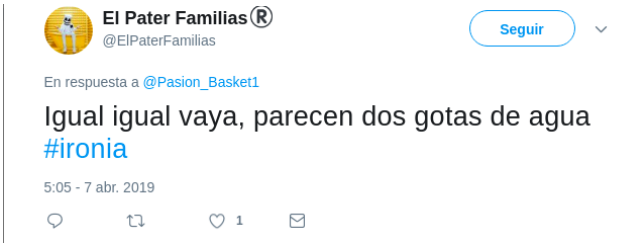


Figura 4.9: Texto irónico tomado a prueba. En esta muestra se obtuvo un 0.992789 de porcentaje de ironía. Para el experimento 4

# Experimento	Accuracy	Precision	Recall	F1-score
Experimento 1 (palabras)	0.9687	0.9092	0.7669	0.8304
Experimento 2 (1-gram)	0.9825	0.9552	0.8662	0.9084
Experimento 3 (2-gram)	0.9835	0.9371	0.8958	0.9157
Experimento 4 (3-gram)	0.9338	0.6471	0.9066	0.7451

Tabla 4.5: Tabla total de métricas.

En esta tabla se puede notar que el tercer experimento de 2-gram tiene dos métricas mejor que cualquier otro experimento, la primera es el *accuracy* con un 98.35 % y el *F-score* de 91.57 % lo cual está indicando se comporta en general como un mejor modelo que cualquiera de los otros 3. Sin embargo, en el *recall* se puede ver que es mejor el del cuarto experimento, llegando a 90.66 % lo cual como se mencionó puede ser mejor en algunos casos que sea muy importante encontrar la mayor parte de las muestras positivas, sin embargo su *precision* cae abruptamente. Hablando del *precision* se puede ver que es mejor el segundo experimento, sin dejar caer tanto el *recall* por lo que sería un mejor modelo en un caso en el que se requiera obtener una gran parte de muestras positivas sin disparar tantas falsas alarmas.

Capítulo 5

Conclusiones

Como se mencionó en el capítulo 1 el objetivo era obtener resultados aceptables en la clasificación de la ironía en textos cortos, y por lo visto en el capítulo 4 los resultados descritos se asemejan a los de la bibliografía y en algunos casos superan lo esperado, por lo que el objetivo puede considerarse cumplido. Sin embargo, dentro de las observaciones de mejora que se reportan está cambiar la arquitectura del modelo. Como se trato en el capitulo 4 la tesis que se presenta se enfoca más en el preprocesamiento ya que como demuestran los resultados impacta de forma directa en el desempeño del modelo. En este proyecto se usa una arquitectura del modelo que ha reportado buen desempeño en la clasificación de reseñas de películas de IMDB. La cual es una tarea que se acerca bastante a la clasificación de la ironía, ya que se trata de extraer una opinión en ambas tareas, la cual puede ser positiva o negativa, en nuestro caso irónica o no irónica.

# Experimento	Accuracy	Precision	Recall	F1-score
Experimento 1 (palabras)	0.9687	0.9092	0.7669	0.8304
Experimento 2 (1-gram)	0.9825	0.9552	0.8662	0.9084
Experimento 3 (2-gram)	0.9835	0.9371	0.8958	0.9157
Experimento 4 (3-gram)	0.9338	0.6471	0.9066	0.7451

Tabla 5.1: Tabla total de métricas.

La tarea de clasificación de la ironía se ha llevado acabo por diferentes centros de estudio, con diferentes enfoques y justificaciones, desde los sistemas de reglas que propuso Utsumi (1996) a las técnicas modernas como arboles aleatorios Jasso & Meza (2016). Como se explico brevemente en el capítulo 4 la tesis presente se enfoca principalmente en definir cual de los 3 enfoques de preprocesamiento se adaptan mejor a la clasificación de la ironía, y resultó que en promedio de *F-score* el mejor modelo fue el del tercer experimento.

A lo largo de los experimentos se pudo ver que sus métricas cambiaban bastante, debido principalmente a como se llevaba a acabo el preprocesamiento. Este preprocesamiento indicó que fue mucho mejor realizar un preprocesamiento por n-gram de dos, debido a que el *F-score* es mejor que en cualquiera de los experimentos. Para retomar un poco el análisis que se realizó en los diferentes experimentos, se describirá brevemente cual fue el preprocesamiento, la configuración de la red neuronal y los resultados.

En el primer experimento se tomaron las palabras separadas por signos de puntuación o espacios, como tokens. A estos se les aplicó una discretización, transformándolos a un entero, el cual fue su índice en un diccionario. De este preprocesamiento se extrajeron vectores por tweet, los cuales se pasaron por una red neuronal con una configuración de una capa de embeddings, una de BI-LSTM y por último una capa totalmente conectada, de la cual se obtenía un 1 si la sentencia fue irónica y un 0 si no lo era.

Sus resultados fueron relativamente buenos, ya que obtuvo un promedio de *F-score* de 83.04 %. Respecto a sus medidas de *precision* y *recall*, este experimento tuvo 90.92 % y 76.69 % respectivamente.

En el segundo experimento se hizo un preprocesamiento llamado n-gram de 1, por caracter. En este experimento no se excluyeron los caracteres especiales. Se tuvo una configuración similar en la red neuronal, con excepción que la entrada ahora fueron vectores de 200 elementos.

Sus resultados fueron significativamente mejores que los del primer experimento ya que sube a 90.84 % de *F-score*. Tanto el *precision* como el *recall* mejoraron teniendo 95.52 % y 86.62 % respectivamente. Este segundo modelo se considera por los resultados mejor que el anterior en todo sentido.

En el tercer experimento se aumento el n-gram a 2 y se uso la misma red neuronal que la del segundo experimento.

Sus resultados fueron ligeramente mejores que el anterior, llegando a tener un 91.57 % de *F-score*. Sin embargo, el *precision* disminuye a 93.71 %, pero aumenta el *recall* a 89.58 %. Por estas pequeñas diferencias se podría considerar un mejor modelo el del tercer experimento. No obstante, si se desea un modelo que reporte una menor cantidad de falsos positivos, el primer modelo sería mejor.

En el cuarto experimento se aumento el n-gram a 3 y se uso la misma configuración de red neuronal.

En cuanto a sus resultados fueron los peores de los 4 experimentos.Cayo el *F-score* a 74.51 %. Su *precision* cayó a 64.71 %. Sin embargo su *recall* aumento respecto a cualquier experimento a 90.66 % lo cual lo hace el experimento con mejor *recall*. Este modelo es el peor de todos en promedio, sin embargo, si lo que se quiere es un sistema que encuentre la mayor cantidad de muestras positivas este es el mejor de todos, aunque terminará prediciendo también una gran cantidad de falsos positivos.

El modelo más confiable respecto a las métricas es el tercero por su promedio de *F-score*, el cual indica que es el mejor equilibrado y que aunque no es el que tiene el mejor *recall* o *precision* tiene la mayor confiabilidad de los tres ya que se equivocará menos veces.

Cabe recalcar que estas son medidas que describen el comportamiento del modelo. Sin embargo, es posible que en algunos problemas sea mucho más importante tener un mejor *precision* o un mejor *recall*. Esto dependerá del uso que tenga el modelo.

Por todas estas razones se puede ver que el objetivo de proponer un modelo de red neuronal que pueda identificar la ironía, se ha cumplido. Debido a lo descrito el capítulo 4 se encuentra que el *F-score* promedio del experimento 3, es el mejor del los 4 experimentos que se realizaron. A pesar de que los resultados son superiores a los reportados en el capítulo 2, estos no se puede comparar directamente debido a que los corpus usados en la bibliografía confiaban enteramente en que los usuarios etiquetaban correctamente los textos irónicos. Sin embargo, esto no

Figura 5.1: Ejemplo de una muestra que no esta bien etiquetada.

es así siempre. Si los cálculos reportan que el porcentaje de muestras irónicas es de alrededor del 10 % se debe tener en cuenta que también este porcentaje de muestras etiquetadas como irónicas pueden aportar información confusa que impactará directamente en el desempeño, y las métricas después no podrán ser completamente fidedignas ya que no se toma en cuenta que no es la ironía lo que se está detectando, sino cuando los usuarios usarán la etiqueta de #ironía. Como se puede ver en la figura 5.1.

Por esta razón no se pueden comparar los resultados extraídos en esta ocasión, ya que no hay trabajo previo con el se puedan comparar directamente. Aún así el modelo aquí propuesto tiene un desempeño con un 98.35 % de accuracy, 93.71 % de *precision*, 89.58 % de *recall* y 91.57 % de *F-score*, lo cual significa que el 98.35 % de las ocasiones acertará en la clasificación asignada, el 93.71 % de las muestras detectadas como irónicas son de verdad irónicas y el modelo encontrará el 89.58 % de las muestras irónicas. Dicho desempeño puede considerarse bueno ya que su *F-score* supera el 80 % promedio.

5.1. Trabajos a futuro

Como sugerencia para trabajos posteriores se puede establecer una metodología de exploración entre los diferentes modelos que podrían dar mejores resultados. Como la combinación de la arquitectura CNN y la LSTM. Incluso se podrían explorar los nuevos algoritmos de optimización como el Artifitial Plant Optimization o algoritmos genéticos.

Por otro lado se pueden implementar o aumentar el preprocesamiento que se usó en este trabajo, tal vez añadiendo etiquetas POS para que éstas aporten diferentes características a la clasificación. Además de esto mi sugerencia es aumentar el tamaño del corpus para obtener una muestra más significativa de la ironía. También puede considerarse una revisión a fondo de los tweets que componen el corpus con el fin de depurar errores de concepto de ironía.

Además podría realizarse un estudio en el que se extraigan más clases de ironía como sarcasmo o sátira. Para obtener esto se deben tener claros los conceptos de ironía, sarcasmo y sátira, por lo que se sugiere antes de hacer un modelo que distinga de estos tres, realizar un modelo que distinga el sarcasmo de lo que no lo es, y otro que distinga la sátira de lo que no lo es. Esto sugiero se lleve acabo en un corpus con muestras irónicas y sarcásticas/satíricas con el fin de que el modelo también pueda distinguir el sarcasmo/sátira de la ironía.

Los textos usados en este trabajo proceden de diferentes países. Sin embargo, el idioma no se comporta de la misma manera en todos los lugares donde se habla. Por lo que otro factor a tomar en cuenta para realizar un mejor modelo que detecte mejor la ironía en español es recabar datos de igual tamaño de todos los países donde se hable este idioma. Entonces se tienen dos opciones, realizar el modelo. Por cada uno de los países realizar otro modelo que detecte de que país es el texto, un problema bastante complejo, y de este modo redirigir el texto al modelo que detecte la ironía para ese país. La otra opción es crear un modelo que tome todo el corpus de todos los países y se entrene para detectar la ironía. Si se realiza de la primera forma se tendrá una forma más certera de encontrar la ironía ya que se adaptaría el modelo a la forma de hablar en ese país. De la segunda forma se ahorraría el primer modelo que clasifique la nación, y solo se tendría que entrenar un modelo, por lo que se ahorraría tiempo. Sin

embargo, dado que se usaría la misma arquitectura para detectar la ironía en diferentes países, el modelo tendería a tener un peor desempeño debido a que se entrenaría para un dialecto universal del español, el cual en la realidad no existe. Esto provocaría que se contradiga algunas veces debido a que las mismas palabras tienen diferentes significados en diferentes países.

Bibliografía

Bamman, D. & Smith, N.A. Contextualized Sarcasm Detection on Twitter. *ICWSM* **2**:15 (2015)

Banik, P.P., Saha, R., & Kim, K.D. LED color prediction using a boosting neural network model for a visual-MIMO system. *Optics Communications* (2018)

Barbieri, F. & Saggion, H. Automatic Detection of Irony and Humour in Twitter. En *ICCC*, págs. 155–162 (2014)

Barbieri, F., Ronzano, F., & Saggion, H. Italian irony detection in twitter: a first approach. En *The First Italian Conference on Computational Linguistics CLiC-it*, pág. 28 (2014)

Bhatia, R. Is neural networks the greatest algorithm of all times (2017)

Cardellino, C. Spanish Billion Words Corpus and Embeddings (2016)

Charalampakis, B., Spathis, D., Kouslis, E., & Kermanidis, K. Detecting irony on greek political tweets: A text mining approach. En *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS)*, pág. 17. ACM (2015)

Davidov, D., Tsur, O., & Rappoport, A. Semi-supervised recognition of sarcastic sentences in twitter and amazon. En *Proceedings of the fourteenth conference on computational natural language learning*, págs. 107–116. Asso-

ciation for Computational Linguistics (2010)

Domingos, P. A few useful things to know about machine learning. *Communications of the ACM* **55**(10):78–87 (2012)

Fawcett, T. An introduction to ROC analysis. *Pattern recognition letters* **27**(8):861–874 (2006)

Filatova, E. Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing. En *LREC*, págs. 392–398. Citeseer (2012)

Frenda, S. Ironic gestures and tones in twitter. En *4th Italian Conference on Computational Linguistics, CLiC-it 2017*, tomo 2006, págs. 1–6. CEUR-WS (2017)

Goldberg, Y. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* **57**:345–420 (2016)

Huang, H.H., Chen, C.C., & Chen, H.H. Disambiguating false-alarm hashtag usages in tweets for irony detection. En *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, tomo 2, págs. 771–777 (2018)

Jasso, G.L. & Meza, I.R. Character and word baselines systems for irony detection in Spanish short texts. *Procesamiento del Lenguaje Natural* **56**:41–48 (2016)

Joshi, A., Bhattacharyya, P., & Carman, M.J. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)* **50**(5):73 (2017)

Kar, A.K. Bio inspired computing–A review of algorithms and scope of applications. *Expert Systems with Applications* **59**:20–32 (2016)

Karoui, J., Zitoune, F.B., & Moriceau, V. SOUKHRIA: Towards an Irony Detection System for Arabic in Social Media. *Procedia Computer Science* **117**:161–168 (2017)

Kingma, D.P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014)

Kong, L. & Qiu, L. Formalization and Rules for Recognition of Satirical Irony. En *Asian Language Processing (IALP), 2011 International Conference on*, págs. 135–138. IEEE (2011)

Kotsiantis, S.B., Zaharakis, I., & Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering* **160**:3–24 (2007)

Liebrecht, C., Kunneman, F., & van Den Bosch, A. The perfect solution for detecting sarcasm in tweets# not. *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (2013)

Liu, J., Cao, Y., Lin, C.Y., Huang, Y., & Zhou, M. Low-quality product review detection in opinion summarization. En *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (2007)

Maynard, D. & Greenwood, M.A. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. En *LREC 2014 Proceedings*. ELRA (2014)

McCulloch, W.S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5**(4):115–133 (1943)

Mitra, S. & Pal, S.K. Fuzzy multi-layer perceptron, inferencing and rule generation. *IEEE Transactions on Neural Networks* **6**(1):51–63 (1995)

Nafis, S.T.O.P.T. & Khanna, S. An improved method for detection of satire from user-generated content. *International Journal of Computer Science and Information Technologies, Vol. 6* (2015)

Nozza, D., Fersini, E., & Messina, E. Unsupervised Irony Detection: A Probabilistic Model with Word Embeddings. En *KDIR*, págs. 68–76 (2016)

Olah, C. Understanding LSTM Networks (2015)

Poria, S., Cambria, E., Hazarika, D., & Vij, P. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815* (2016)

Ptáček, T., Habernal, I., & Hong, J. Sarcasm detection on czech and english twitter. En *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, págs. 213–223 (2014)

Reddy, D.R., Bhramara, P., & Govindarajulu, K. Performance Analysis of Domestic Refrigerator Using Hydrocarbon Refrigerant Mixtures with ANN. *Numerical Heat Transfer and Fluid Flow: Select Proceedings of NHTFF 2018* pág. 113 (2019)

Reyes, A. & Rosso, P. Making objective decisions from subjective data: Detecting irony in customer reviews. *Decision Support Systems* **53**(4):754–760 (2012)

Romero, S. Borges incorregible: sus frases más irónicas (2012)

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65**(6):386 (1958)

Ruder, S. An overview of gradient descent optimization algorithms. *CoRR* **abs/1609.04747** (2016). `1609.04747`

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. Learning representations by back-propagating errors. *nature* **323**(6088):533 (1986)

Santos, A.L. Assessing the culture of fruit farmers from Calvillo, Aguascalientes, Mexico with an artificial neural network: An approximation of sustainable land management. *Environmental Science & Policy* **92**:311–322 (2019)

Utsumi, A. How to interpret irony by computer: A comprehensive framework for irony. En *Proceedings of the International Conference "Recent Advances*

in NLP, págs. 315–321 (1995)

Utsumi, A. A unified theory of irony and its computational formalization. En *Proceedings of the 16th conference on Computational linguistics-Volume 2*, págs. 962–967. Association for Computational Linguistics (1996)

Wallace, B.C., Kertz, L., Charniak, E. *et al.* Humans require context to infer ironic intent (so computers probably do, too). En *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, tomo 2, págs. 512–516 (2014)

Wallace, B.C., Charniak, E. *et al.* Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. En *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, tomo 1, págs. 1035–1044 (2015)