2018

CATEGORIZACIÓN DE COMENTARIOS, SEGÚN LA EMOCIÓN EXPRESADA, UTILIZANDO PROCESAMIENTO DE LENGUAJE NATURAL Y LINGÜÍSTICA COMPUTACIONAL

ATUAN TOLEDO, JEAN-PIERRE

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE

"CATEGORIZACIÓN DE COMENTARIOS, SEGÚN LA EMOCIÓN EXPRESADA, UTILIZANDO PROCESAMIENTO DE LENGUAJE NATURAL Y LINGÜÍSTICA COMPUTACIONAL"

JEAN PIERRE ATUAN TOLEDO

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUÍA: **MARÍA JOSÉ ESCOBAR**

PROFESOR CORREFERENTE: **GONZALO CARVAJAL**

Agradecimientos

Resumen

En el presente estudio, se busca generar un modelo predictivo capaz de categorizar comentarios provenientes de la red social Twitter mediante el proceso de estudio denominado "Análisis de Sentimiento".

En particular, el estudio busca encontrar una vía alternativa a una de las funciones de la plataforma de la empresa chilena Wholemeaning, la cual por medio de diversos algoritmos lingüísticos es capaz de categorizar texto de acuerdo al sentimiento (o emoción) presente en él. Esta vía, si bien funciona muy bien bajo otros tipos de categorizaciones, no es capaz de etiquetar correctamente todos los distintos tipos de comentarios realizados bajo esta área. Y es que al tratarse de un contexto chileno, la jerga y la variada forma que tienen las personas de expresar una idea, ya sea haciendo uso de garabatos o con frases muy específicas, hace que la implementación de un modelo lingüístico no logre capturar todas sus variantes.

El método propuesto se basa en el uso del procesamiento natural del lenguaje (PNL) y la lingüística computacional (LC), los cuales a su vez están fuertemente complementados con algoritmos de un área de Machine Learning, como lo es el Deep Learning. Con ello, se generará una herramienta que pueda trabajar procesando texto proveniente de usuarios chilenos, intentando capturar toda la jerga propia del chilenismo que permita una correcta etiquetación de los comentarios.

Abstract

The present study aims to generate a predictive model capable of categorizing comments originating from the social network Twitter, through an analysis process denominated "Emotion Analysis".

In particular, this study looks to find an alternative means of fulfilling one of the functions on the platform of the Chilean company Wholemeaning, which through the use of diverse linguistic algorithms is able to categorize text according to feeling (or emotion) presented in writing. Though the procedure works very well when creating other kinds of categorization, it is incapable of correctly tagging all the different types of commentary realized within the area of emotion. In the Chilean context, the difficulty arises when the jargon and variation in form that people use to express an idea, whether it be through curse words or using very specific phrases, inhibits the linguistic model from capturing all the variants presented.

The proposed method is based on the use of Natural Language Processing (NLP) and Computational Linguistics (CL), which are strongly complemented with algorithms from an area of Machine Learning known as Deep Learning. The resulting tool that can be generated is capable of processing text originating from Chilean users, attempting to capture the jargon of typical Chilean idioms, and permitting the correct tagging of observed comments.

Índice general

Índice de figuras

Índice de cuadros

1. CAPÍTULO: PROBLEMA DE INVESTIGACIÓN

1.1. Descripción de la Problemática

Wholemeaning, una empresa chilena, se ha dedicado los últimos 5 años a desarrollar tecnología dedicada al procesamiento del lenguaje natural, buscando revolucionar el conocimiento de cómo y sobre qué habla la gente. Sus sistemas son capaces de monitorear un gran flujo de datos provenientes de redes sociales, como lo son Facebook y Twitter, cuyo contenido tengan una conexión directa con alguna empresa en particular (clientes de Wholemeaning).

Todos los datos capturados son profundamente analizados mediante un software capaz de catalogar los comentarios de acuerdo a distintos parámetros según un área específica. Por ejemplo, un usuario puede agradecer los servicios de una empresa X del área de comunicaciones. Por lo que este comentario puede ser catalogado como: Servicio - Satisfacción. Ahora, dependiendo del área en que la empresa se encuentre, podrá tener distintos tipos de categorizaciones. Una empresa de Retail puede no necesitar los mismos indicadores que una dedicada a Telecomunicaciones. El algoritmo encargado de realizar el etiquetado de comentarios ha sido desarrollado por lingüistas, quienes por medio de un extenso estudio sobre cómo la gente se expresa de forma escrita, han implementado distintas reglas de semántica y de sintáxis de acuerdo a como ha sido escrito el comentario, para posteriormente ser catalogados.

Es muy difícil poder crear todas las reglas lingüísticas que permitan capturar la información de un texto independiente de cómo y con qué palabras se escriba. El idioma español tiene la facilidad de poder expresar una misma idea de distintas maneras, y si esto lo llevamos al dialecto chileno, su dificultad aumenta aún más al tener una gran cantidad de nuevas

palabras, que pueden ser escritas de muchas formas. Dado esto, es que el software comete errores en sus predicciones, errores que producen que los análisis posteriores realizados sobre sus resultados, produzcan información sesgada con cierto margen de error.

1.2. Propuesta de Tema

Este trabajo se basa en la constitución de un algoritmo de clasificación de comentarios provenientes de la red social Twitter. Una buena predicción de texto, no tan sólo ayudará en la credibilidad asociada a la herramienta, sino también permite que los clientes de Wholemeaning puedan tomar decisiones, respaldados por un confiable soporte de conocimiento.

El modelo a utilizar se basa en el uso del procesamiento natural del lenguaje (PNL) y la lingüistica computacional, los cuales a su vez están fuertemente complementados con algoritmos de un área de la inteligencia artificial, como lo es el Deep Learning. Con ello, se generará una herramienta que pueda trabajar procesando texto proveniente de usuarios chilenos, capturando toda la jerga propia del chilenismo que permita una correcta etiquetación del comentario.

1.3. Objetivos Generales y Específicos

1.3.1. Objetivo General

Mejorar el servicio actual de clasificación de comentarios y mensajes proveniente de la red social Twitter que posee la empresa Wholemeaning. El presente estudio sólo buscará resolver uno de los tipos de etiquetación, que es conocido como “Sentiment Analysis” o análisis de sentimiento en su traducción literal.

Se generará una herramienta capaz de categorizar comentarios de twitter según la emoción expresada en él, por lo que uno podrá ser catalogado como positivo, negativo o neutro si es que carece de emoción.

1.3.2. Objetivos Específicos

Dentro de los objetivos específicos propuestos, se encuentra:

- Catalogar los comentarios de Twitter como positivo, negativo o neutro, según la emoción expresada.
- Abarcar la totalidad de los mensajes asociados al área de telecomunicaciones.
- Superar el porcentaje de acertividad de la actual herramienta de la empresa.
- Permitir que el modelo pueda ser escalable y modificable.
- Que la categorización sea realizada en tiempo real (respuesta inmediata).

2. CAPÍTULO: ANTECEDENTES

2.1. Empresa Wholemeaning

Wholemeaning es una empresa chilena que desarrolla tecnología del área de procesamiento de lenguaje natural. Su objetivo principal es tangibilizar la experiencia y percepción del cliente por medio del análisis de grandes volúmenes de información no estructurada proveniente del feedback del cliente, este feedback es capturado mayoritariamente de redes sociales tales como Facebook y Twitter, plataformas donde las personas mantienen interacciones ya sea directa e indirectamente con empresas de distintos rubros.

Fig. 2.1: Logo empresa Wholemeaning

Todos los datos capturados son posteriormente analizados por medio de distintas herramientas encargadas de categorizar el texto según lo que el expresa, y según el sector de donde proviene. Es decir, comentarios destinados a Bancos no tendrán un mismo análisis que comentarios destinados a empresas de Retail, ya que el vocabulario y la forma de expresarse de las personas difieren mucho uno de otro. Una vez que los datos son transformados en información, estos son presentados en inmediatamente bajo una plataforma web de la empresa, un ejemplo de sus dashboard, asociados a mensajes etiquetados como negativos, positivos y otros, puede ser observado en la figura 2.2. Es a partir de aquí, donde clientes de Wholemeaning pueden transformar la información procesada desde el feedback obtenido, en

conocimiento que puede servir como una base en futuras tomas de decisiones.

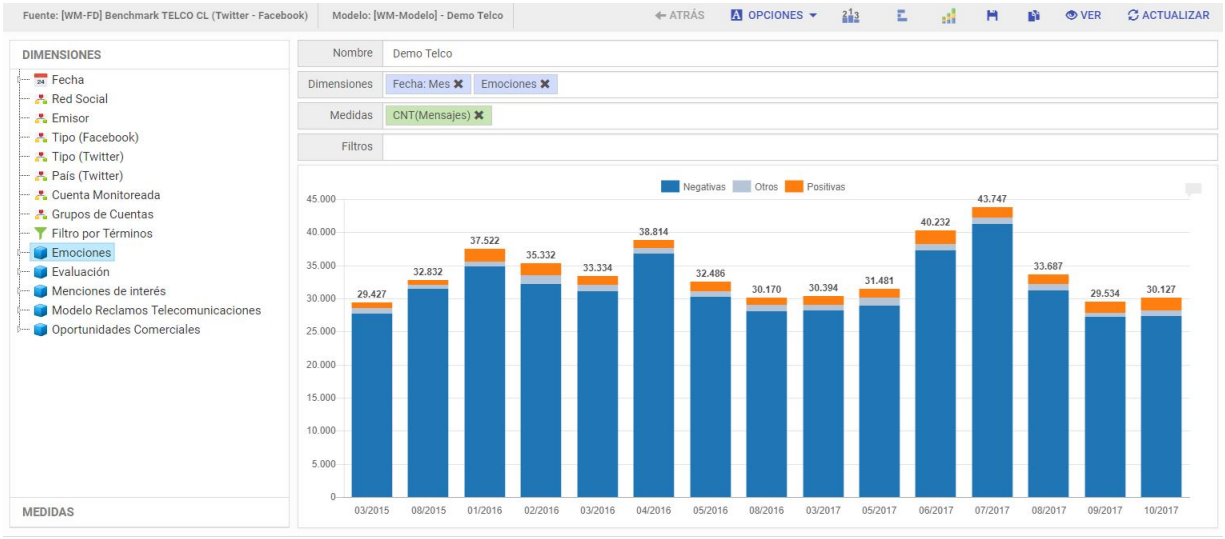


Fig. 2.2: Plataforma web Wholemeaning

La plataforma web es capaz de entregar la información de distintas maneras, permitiendo al usuario adaptar los datos de una manera acorde a sus necesidades, tal como si se tratase de una herramienta de inteligencia de negocios. Esto permite una gran flexibilidad al momento de analizar la información de distintos puntos de vista. La empresa posee además de la plataforma, otros softwares complementarios, que ayudan a sus clientes a mejorar sus servicios.

2.2. Importancia de Redes Sociales

Todos los días son escritos más de 500 millones de Tweets (un comentario realizado por medio de Twitter, se considera como Tweet) alrededor del mundo, prácticamente 6000 por segundo (segun sitio internetlivestats [53]). Esto demuestra el masivo uso que se le da a esta red social, que por lo demás es de muy fácil acceso, y de rápido uso, ya que al ser una aplicación de celular y permitir tan sólo 140 caracteres como máximo (esto prontamente cambiará a un máximo de 280), hace que tan sólo en unos segundos un usuario pueda hacer uso de ella.

Chile no se queda atrás en cuanto a su uso. Según un sitio chileno (Analitic [54]) existen

alrededor de 470 mil cuentas activas en el país, lo que representa una fuente importante de opiniones. Los Tweets no están asociados a un tema en particular, si no más bien, pueden abarcar diferentes temas, que en su mayoría pueden verse influenciados por situaciones del momento, situaciones que pueden producir que un concepto en particular se transforme en tendencia, lo cual quiere decir, que muchas personas realizan comentarios sobre el mismo concepto. Ejemplos pueden haber muchos, un caso particular puede ser el día de Cyber-Monday, en donde muchas empresas se ponen de acuerdo para ofrecer una variada gama de descuentos en sus diferentes productos, lo que produce una serie de distintos comentarios haciendo uso del hashtag # CyberMonday (Hashtag es un concepto de Twitter, que permite que agrupar o etiquetar los mensajes de acuerdo a un concepto, cualquier palabra precedida por un # se considera un hashtag). En estos comentarios se pueden encontrar mensajes de distinta índoles, muchos dirigidos a realizar quejas sobre el mal servicio de ciertas empresas, publicidades engañosas, reportar problemas de compras o simplemente avisar buenas ofertas.

El simple ejemplo expuesto, sobre el evento CyberMonday, permite obtener una gran cantidad de información proveniente de los mismos usuarios, información que puede ser cuidadosamente analizada, y permitir obtener distintas estrategias para en un futuro mejorar sus servicios y aumentar con ello sus ganancias.

Twitter también es ampliamente utilizada hoy en día para brindar ayuda directa a los usuarios. Hasta hace muy poco tiempo, el email y la atención telefónica eran las vías más utilizadas para el comercio electrónico (también llamado e-commerce). Sin embargo, las compañías han incorporado como vía de contacto las redes sociales como Twitter, que gracias a su comunicación bidireccional ,inmediata y simple, permite que cada vez más gente y empresas las usen. Esto no tan sólo ayuda en la rapidez, si no también en la exposición que se ve enfrascada la comunicación, a diferencia de una llamada o un mail donde la comunicación queda acotada entre la empresa y el cliente, en el cual no existe un agente externo que pueda opinar con respecto a la resolución del problema (si es que se tratase de uno). Caso contrario ocurre cuando se realiza por medio de una red social, donde la empresa queda expuesta ante

toda la comunidad, logrando que no tan sólo ella se preocupe en responder con inmediatez, si no que se esfuerce en complacer al cliente por medio de su respuesta. Esta claro que si ella no logra satisfacer a su cliente, la percepción que tendrán los usuarios hacia ella puede repercutir de una manera negativa para la compañía.

Esto a su vez puede verse fuertemente influenciado por otros factores, como lo son la cantidad de seguidores que posee una cuenta. No es lo mismo que una persona de 100 seguidores hable mal de cierta compañía que una que posea más de 50 mil seguidores, estos últimos hacen que muchas más personas se enteren de forma prácticamente inmediata sobre el echo en cuestión. Bajo esta misma situación, si la empresa aludida responde mal, o no responde, la mala imagen con que se ve afectada, repercutirá de una manera más global, haciendo que una simple respuesta tenga un impacto directo en la empresa.

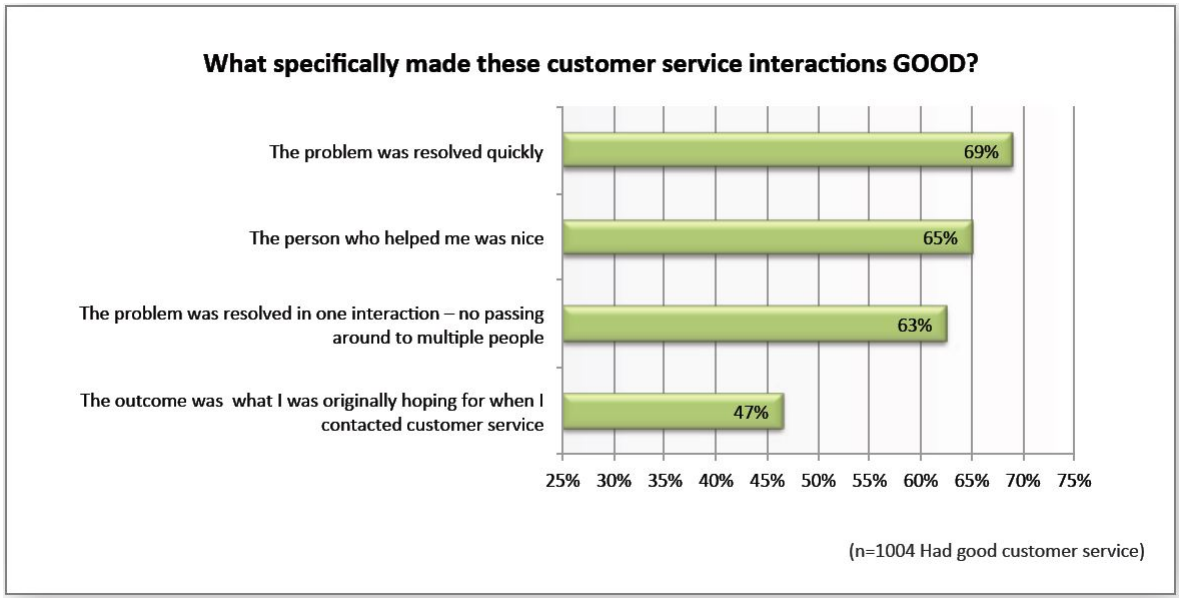


Fig. 2.3: Qué hace que la interacción con un cliente sea buena? (Gráfico obtenido de estudio de Zendesk "Customer Service And Business Results" [55])

Según un estudio realizado por Zendesk, que se puede apreciar en la figura 2.3 (Compañía enfocada en la atención de clientes [55]), realizado a 1046 personas, un 69 % de 1004 (Perso-

nas que tuvieron buenas experiencias) indicó que lo más importante en los servicios brindados por una empresa, es la de poder resolver un problema de manera rápida, un 63 % de esa mismo grupo encuestado, respondió además, que es importante que el problema sea resuelto en sólo una interacción, o sea, sin tener que pasar por múltiples personas. Esto demuestra el alto impacto que puede llegar a generar una empresa por medio del uso de plataformas sociales, siempre que ella ayude de forma rápida y con una correcta atención hacia el cliente.

Twitter, por sus cifras, demuestra ser una fuente prácticamente inagotable de información, que con un debido análisis es capaz de entregar factores que pueden afectar directa y positivamente en las funciones de una empresa en particular.

2.2.1. Análisis de Sentimiento

Actualmente, antes de realizar una compra, visitar un restaurant o alojarse en un hotel, nos informamos por medio de diversos sitios o aplicaciones sobre los comentarios o experiencias que han tenido otras personas referente a ellos, para tomar nuestra decisión de asistir o comprar. El *review* [1] realizado por otras personas genera una clara señal de confianza, al sentirse identificado con los problemas o beneficios que menciona esa otra persona.

La mayoría de sitios de ventas o de servicios tienen implementadas secciones donde los usuarios pueden comentar o dar un análisis con respecto al producto que han adquirido, para que así su experiencia pueda ser tomada en consideración por futuros clientes. Mientras mejores sean los comentarios, existirá por ende, una mayor probabilidad de que la venta de ese producto o servicio se vea positivamente influenciado por ello. Caso contrario ocurriría si es que esto es al revés, y es aquí donde la empresa debe tener un mayor énfasis de estudio, ya que, malos análisis afectan negativamente, clientes que podrían haber estado interesados en adquirirlo, pueden cambiar rápidamente de opinión al leer los reviews.

[1] Es una evaluación o crítica constructiva, que puede llegar a ser positiva o negativa dependiendo de lo expresado.

El uso masivo de redes sociales y blogs ha hecho crecer el interés sobre el análisis de este tipo de comentarios, estudio denominado dentro del área como “Análisis de Sentimiento”, siendo una traducción directa de “Sentiment Analysis”. La opinión, la crítica, calificaciones y recomendaciones realizadas en línea, se han convertido en una fuente de información para poder identificar nuevas oportunidades en la comercialización de productos, y es debido a esto, que se han realizado diversos intentos por poder automatizar el proceso de “entender” el qué esta hablando las personas y capturar el “sentimiento” u “emoción” expresado en cada opinión. Antiguamente esto quizás era posible de realizar por medio de personas que se dedicaban de tiempo completo en realizar un estudio en cuanto a la percepción del cliente sobre algún producto en particular, y a partir de ello tomar ciertas decisiones. Hoy en día esto es prácticamente imposible, dado el volumen de mensajes y lo masivo que se han vuelto las interacciones en Internet.

La tarea más básica en el análisis de sentimiento, es la de poder categorizar un documento, texto o frase en dos estados, es decir que tan positivo o negativo es de acuerdo a la opinión expresada en él. Otros análisis agregan un tercer grupo denominado neutro, el cual ayuda a etiquetar comentarios que no poseen características que permitan determinar su polaridad. Estos dos enfoques son quizás los más estudiados, que, aunque ayudan a tener un mejor desempeño al momento de automatizar su proceso, dado su bajo número de grupos diferentes, no son capaces de alcanzar porcentajes altos de precisión como otras tareas automáticas de clasificación. Esto tiene una razón de ser, y es que para que un sistema automático, sea capaz de obtener el 100 % de texto bien clasificado, se necesita estar de acuerdo completamente con el sentimiento atribuido, y es aquí donde las comparaciones no siempre coinciden, ya que el porcentaje de precisión que logre la máquina, dependerá completamente de las percepciones de que tenga la persona sobre la positividad o negatividad del mensaje.

Los primeros acercamientos hacia este tipo de tareas fueron realizados sobre el análisis de documentos relacionados con review de películas, haciendo una clasificación de ellos no sobre su tópico, como comúnmente se realizaba, si no sobre el sentimiento general que en

él se expresaba para ello, se utilizaron tres algoritmos relacionados con machine learning, Naive Bayes, Maxima Entropía de clasificación y Máquina de Vectores de Soporte (B. Pang, L. Lee and S. Vaithyanathan [1]). Los resultados obtenidos demostraban que este tipo de clasificación sobre la positividad o negatividad de un review, era particularmente más difícil de realizar, que las clasificaciones tradicionales básadas en diferentes temas. Otro acercamiento del mismo tipo, se realizó analizando reviews proveniente de productos (Peter D. Turney [2]) , sobre que tan recomendado o no podría ser, basado en la opinión dejada por usuarios. Su metodología capturaba la orientación semántica de cada frase, es decir, si tenía asociaciones positivas (Ej. "Ambiente romántico") era considerada como una orientación semántica positiva, si esta presentaba asociaciones malas (Ej. "Eventos horribles"), la orientación semántica era tomada como negativa, y por ende como un review categorizado como "No recomendado".

Como fue mencionado, buscar la polaridad de un comentario o review es lo "básico" en este tipo de tareas, algunos autores se quedan con este tipo de clasificación, otros agregan una tercera clase, la neutra, haciendo hincapié en la importancia que tiene esta al momento de hacer el estudio (M. Koppel and J. Schler [3]), ya que no todos los textos demuestran sentimientos o emociones de carácter positivo o negativo, por lo que la clasificación de un comentario de acuerdo a sólo dos tipos distintos no parece ser el mejor camino, ya que forzará a uno de los dos lados aún cuando no tenga ningún rasgo de ellos.

Bajo este último punto, de que no todo es negro o blanco, si no que pueden existir distintos matices, es que el "Análisis de Sentimiento" busca capturar los distintos matices para representar la información de una manera más exacta, o más real. Por ejemplo no hacer una clasificación de acuerdo a la polaridad, si no, intentando predecir la cantidad de estrellas que tendrá un determinado review (B. Pang and Lillian Lee [4]), o bien buscando una clasificación del tipo Muy Negativo - Negativo - Neutro - Positivo - Muy positivo (R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng and C. Potts [5]), y hasta llevando el estudio hacia el tipo de emoción que representa el texto, dividiendo las clases entre Enojo - Disgusto - Miedo

- Felicidad - Tristeza - Sorpresa - Nada,(Shiyang Wen and Xiaojun Wan [6]). Estudios que sin duda elevan aún más el grado de complejidad presente en el estudio.

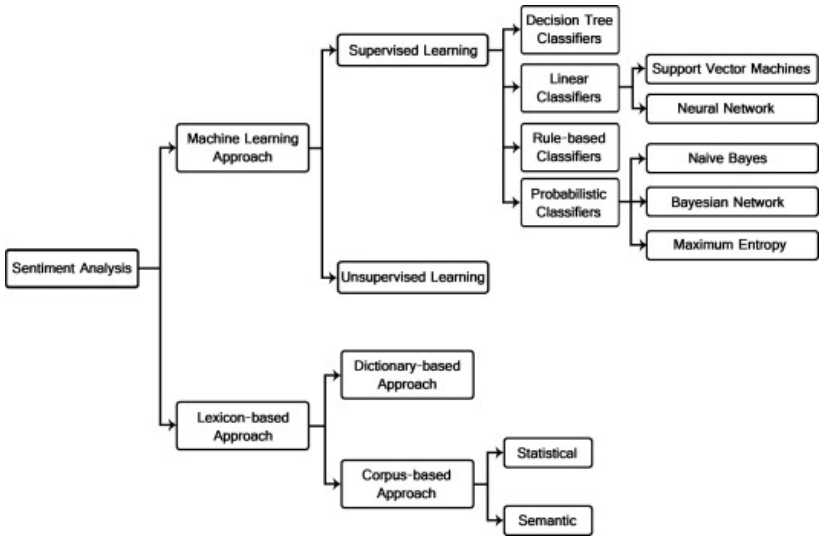


Fig. 2.4: Diferentes técnicas de clasificación (W. Medhat, A. Hassan, H. Korashy [7])

Pero no tan sólo el enfoque que se le da a la clasificación es variada, también lo son las metodologías utilizadas para realizar sistemas de clasificación automáticos, algunas de ellas, organizadas dependiendo de su enfoque, pueden ser observadas en la figura 2.4, de la cual se desmarcan dos grandes grupos, el primero toma en consideración las máquinas de aprendizajes, y el segundo, basado en el léxico. Este último requiere una mayor intervención humana, al basarse directamente en comparaciones realizadas sobre un diccionario base, el cual debe ser cuidadosamente elaborado, intentando de tener en él, la mayor cantidad de frases o palabras de ejemplo, llamados grupos de evaluación (C. Whitelaw, N. Garg and S. Argamon [8]) que orientan a un determinado texto a una clase en específico, por lo que un buen resultado estará directamente ligado a una buena construcción de estos grupos de palabras. Dos diccionarios lexicos bastantes conocidos son WordNet [9] y Thesaurus [10]. Algunos otros enfocados en la semántica, clasifican una opinión, de acuerdo a la relación que existe entre el sujeto y el verbo, y en como es este conjugado dependiendo del sentido de la oración (Isa Maks and Piek Vossen [11]). Si bien estos enfoques pueden tener muy buenos resultados, el trabajo previo y

posterior que se deben realizar es grande y no puede ser utilizado si es que se cambia de área de análisis o de lenguaje, es decir, su construcción estará diseñada específicamente para el uso en su contexto, no pudiendo (quizás si en algunos casos), servir de base para otros casos.

Dada estas dificultados, el uso de machine learning (ML) parece ser una mejor opción, ya que permite resolver ciertas problemáticas con respecto al otro grupo de metodologías. Machine Learning permite que el proceso sea un poco más independiente, ciertas metodologías permiten incluso que pueda ser independiente del idioma. Dentro de ML, los métodos más utilizados están relacionados con algoritmos de aprendizaje supervisados, lo que demuestra aún la co-dependencia que existe entre la metodología y el ser humano ya que, es este último quién será el guía de aprendizaje. Para que estos métodos aprendan, necesitan la existencia de documentos (frases u opiniones) que se encuentren previamente etiquetados, es decir, que existan comentarios ejemplos asociados a una clase que se quiera predecir (de esto se hablará en detalle más adelante, en la sección de Dataset). Si se desea clasificar sólo comentarios positivos y negativos, deben tenerse diversos ejemplos para cada uno de ellos.

El método de ML más utilizado, es el clasificador de Naive Bayes (NB), siendo este un clasificador probabilístico fundamentado en el teorema de Bayes [2], que clasifica un texto de acuerdo a la probabilidad de una clase, basada en la distribución de las palabras dentro del texto. El modelo trabaja con características de BOW (Bag of Words, Bolsa de palabras en su traducción), la cual ignora la posición de las palabras dentro de la oración, por lo que el teorema de Bayes predice la probabilidad de que dada las características de ese BOW, o sea, de las características de las palabras dentro de la oración, pertenezcan a una determinada clase, con lo que, simplificadamente puede verse como:

$$P(Clase|Caractersticas) = \frac{P(Clase) * P(Caractersticas|Clase)}{P(Caractersticas)} \tag{2.1}$$

[2]Teorema que expresa la probabilidad condicional de un evento aleatorio A dado B

Ciertas mejoras a este modelo, dado ciertos problemas que incrementaban la probabilidad a la clase de comentarios negativos, es que se han propuestos ciertas mejoras ayudados también con enfoques léxicos (H Kang and S. J. Yoo [12]). Otro método usado en el análisis de sentimiento es la de máquinas de vectores de soporte (Yung-Ming Li and Tsung-Ying Li [13]) , SVM por su nombre en inglés Support Vector Machine, el cual busca determinar separadores lineales en el espacio de búsqueda, que permitan la mejor separación de clases (Un ejemplo de esto, puede observarse en la figura 2.5).

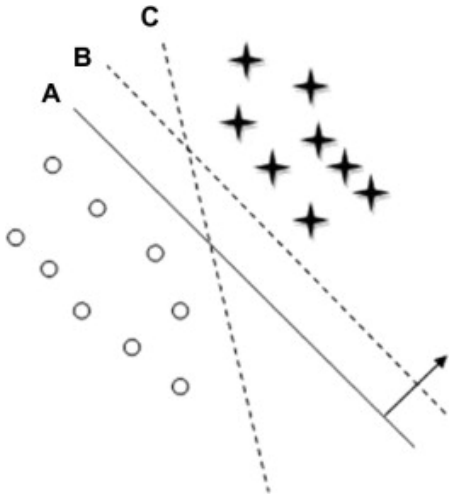


Fig. 2.5: Ejemplo de separadores lineales utilizando Máquina de Vectores de Soporte, cualquier linea paralela que se encuentre entre las lineas A y B representan los mejores casos de separación, ya que en promedio, todos los puntos tienen una misma distancia a la linea divisora, esto a diferencia de la C. (W. Medhat, A. Hassan, H. Korashy [7]

Otros métodos también son aplicados en esta área como; clasificadores Lineales, clasificadores de Máxima Entropía, redes Bayesianas, etc, que pueden tener a su vez distintos enfoques o modificaciones, e incluso también, ser sistemas híbridos, que tomen en consideración más de un método, aprovechando con ello, beneficios propios de cada uno. Últimamente uno de los métodos que han tenido grandes avances en esta área, es la de redes neuronales,

más específicamente el aprendizaje profundo (Por su nombre en inglés, Deep Learning), el cual ha logrado demostrar grandes beneficios a la hora de categorizar texto. Estos métodos serán discutidos más adelante, ya que serán el pilar fundamental del estudio del análisis de sentimiento.

Como bien se ha hablado en esta sección, el AS (Análisis de Sentimiento) ha logrado adquirir una fuerte atención en el mundo de máquinas de aprendizaje, esto dado a su complejidad y a la importancia que conlleva el obtener un buen resultado. Sin duda no lo dejará de ser en el corto tiempo, por lo que nuevos enfoques y nuevas metodologías podrían ser desarrolladas en un futuro próximo.

3. CAPÍTULO: MARCO TEÓRICO

3.1. Sentimiento y Emoción

El área que resume el presente trabajo tiene por nombre “Análisis de Sentimiento” el cual proviene de su traducción directa del inglés “Sentiment Analysis” , y es que a pesar de que se hace una directa alusión hacia un estudio sobre el sentimiento expresado en un texto o frase, en realidad análisis vendrá dado hacia la emoción que en él se manifiesta.

La diferencia entre sentimiento y emoción es sutil, tanto así que se llegan a usar equivocadamente e indistintamente una de otra. La diferencia radica en que la emoción es un tanto básica, se expresa inmediatamente después de una perturbación o estímulo, no da tiempo de reflexionar sobre ello. Mientras que un sentimiento incluye esta capacidad de reflexionar, de pensar de manera consiente sobre lo que se siente o experimenta, como lo puede ser una obra de arte. Aunque ambos conceptos apuntan a cosas diferentes tienen una relación directa, y es que donde exista una emoción puede siempre presentarse uno o más sentimientos, por lo que su diferenciación resulta útil más en la teoría, para poder expresar de mejor manera procesos neurológicos, que para diferenciarlos en echos concretos.

A partir de estas diferencias, parece claro que el concepto más adecuado al categorizar un comentario es la emoción, ya que se manifiesta de manera directa al leerlo, no da tiempo en reflexionar sobre lo leído. Esto sin duda cambia mientras el texto es más largo, dado que pueden ocurrir en él una serie de distintas emociones, que lleven al lector a un proceso de reflexión, a experimentar un sentimiento, como puede ocurrir al leer un poema.

Si bien en el análisis sobre comentarios, el concepto de “Análisis de Sentimiento” no está directamente relacionado, dado que es un análisis más de emociones que de sentimientos, se utilizará igualmente dicho concepto para referirse al área central de estudio, que es la categorización de texto en 3 clases distintas de acuerdo a la connotación positiva o negativa del lenguaje ocupado.

3.2. Categorización de Comentarios

El objetivo de una categorización es la de etiquetar bajo un mismo identificador objetos que poseen características similares en una misma entidad. Durante el presente trabajo, estas agrupaciones son denominadas clases, donde cada una de ellas, presentará características únicas que la diferenciará de las demás.

Las propiedades de cada una de las clases, están bien definidas y es posible categorizar cualquier comentario con ellas, independiente de la fuente de donde provenga. Aún así, el estudio esta enfocado sólo en un área particular de Wholemeaning, que es denominada internamente como Telco, la cual agrupa a todas las empresas que tengan una relación con las telecomunicaciones. Por lo que es posible estas propiedades no se vean reflejadas en su totalidad durante el trabajo.

Si se verán en una mayor cantidad, comentarios que estén relacionados con la satisfacción o insatisfacción con respecto a un servicio brindado por alguna compañía.

Las características de cada una de las clases serán detalladas a continuación.

3.2.1. Comentarios Positivos

Es considerado un comentario positivo, cuando la emoción u opinión expresada en él, puede ser atribuidas a alguno de los siguientes puntos:

- Emociones Positivas: Expresa de manera directa y concisa alguna de las siguientes

emociones; Alegría, Serenidad, Interés, Esperanza, Orgullo, Diversión, Inspiración, Asombro, Amor. Ejemplo: *Que día más maravilloso* !.

- Satisfacción: Comentario refleja satisfacción hacia una respuesta o acción por parte de alguna entidad o persona.
  Ejemplo: Tienen una excelente atención al público, sigan así! :).

- Elogio: Comentario que expresa una felicitación o elogio con respecto a alguna acción o servicio.
  Ejemplo: *@WOM, bacán, respondieron todas mis consultas, gracias! :D*.

- Gratificación: Comentario refleja en él, gratitud con respecto a alguna acción o servicio.
  Ejemplo: *Muchas gracias por la rápida respuesta!*.

3.2.2. Comentarios Negativos

De manera análoga, un comentario es considerado negativo, cuando la emoción u opinión expresada presenta los siguientes puntos:

- Emociones Negativas: Expresa de manera directa y concisa alguna de las siguientes emociones; Ira, Agresividad, Tristeza, Enojo, Rabia, etc.
  Ejemplo: *Me siento pésimo :(!*.

- Insatisfacción: Comentario refleja insatisfacción con respecto hacia alguna respuesta, acción o servicio por parte de una entidad o persona.
  Ejemplo: *Es increíble que aún no arreglen el Internet, llevamos 3 días sin conexión*.

- Improperios : Comentario que expresa por medio de insultos su opinión o sentimiento con respecto hacia alguna persona o entidad.
  Ejemplo: *Son una m..... como empresa!!!*.

3.2.3. Comentarios Neutros

En esta clase (o categoría), se encontrarán todo tipo de comentarios que no cumplen con los puntos mencionados en comentarios tanto positivos como negativos. Es decir, básicamente serán comentarios de carácter:

- Promociones: Comentarios ligados directamente a propaganda o promociones realizadas por empresas o personas.
  Ejemplo: *Aprovecha esta promoción, y pórtate a nuestra compañía!*.

- Sin emociones: Comentarios que no reflejan ninguna emoción que pueda ser atribuida como positiva o negativa.
  Ejemplo: *Desde mañana empieza a funcionar la nueva forma de pago, en todas nuestras sedes a lo largo de Chile*.

3.3. Procesamiento de Lenguaje Natural y Lingüística Computacional

Hoy en día existe mucha confusión a la real diferencia entre estos dos grandes conceptos del área linguística, básicamente por que ambos apuntas hacia un mismo objetivo principal, que es la de poder comprender el complejo lenguaje que utiliza el ser humano para comunicarse.

Está claro que la forma que tenemos de comunicarnos es muy variada, podemos decir algo de muchas formas distintas, y en el caso del idioma español, llegando a utilizar una alta gama de palabras dependiendo del origen del hablante. Esto hace que utilizar un computador para que, por ejemplo, pueda interactuar con alguien del mismo modo que lo haríamos con otra persona, parece un reto casi imposible. Es exactamente bajo estas dificultades que el área de la lingüística intenta resolver utilizando distintos algoritmos y herramientas computacionales para el estudio y análisis del lenguaje humano.

La lingüistica es básicamente el estudio científico de todos los aspectos relacionados con el lenguaje natural, es decir, aquel lenguaje humano que se adquiere o aprende inconsciente e involuntariamente, bajo la necesidad de poder comunicarse. Esto marca una diferencia respecto al lenguaje artificial que se aprende voluntaria y conscientemente, como pueden ser los lenguajes de programación utilizados para el desarrollo de programas informáticos. Con la masificación del uso de la tecnología, es que la lingüística actual y científica ha hecho uso de la computación como herramienta para su estudio, análisis, y comprensión del idioma humano. Es precisamente de aquí, que el área de la inteligencia artificial a tomado parte de ella, para poder realizar algoritmos capaces de comprender e interactuar con un humano.

Lingüística computacional (LC) y Procesamiento Natural del Lenguaje (PNL) están muy relacionados, tanto que incluso son hoy en día palabras que se usan indistintamente para referirse a un mismo fin. Tanto es así, que ha sido denominado un caos terminológico (Carlos Periñán, 2012) [15] dado lo relacionado que están ambos conceptos. Aún así tienen ciertas matices que hacen diferir una de otra, y la cual radica en que la lingüística computacional se enfoca en el estudio del lenguaje natural haciendo uso de la computación, es decir, la construcción de sistemas computacionales que analicen y/o generen textos en el lenguaje natural. En cambio el PNL se centra en la confección de herramientas (ya sean algoritmos, datasets, etc) que ayuden a la búsqueda de soluciones, a los problemas que plantea la lingüistica computacional.

En este caso de estudio, la LC vendría ligada a toda herramienta computacional que ayude a hacer un mejor estudio del problema que se busca solucionar, más concretamente la utilización de herramientas de vectorización de palabras (campo que se detallará más adelante), y de limpieza de texto (cuya importancia también será explicada más adelante). Con el uso de ellas, es posible aplicar el procesamiento natural del lenguaje, es decir, aplicar diferentes metodologías que permitan que un computador pueda “entender” de cierta forma el lenguaje natural escrito, permitiendo así que este pueda diferenciar entre un comentario negativo, neutro o positivo. Estos algoritmos están derechamente ligados al área de la inte-

ligencia artificial, más exactamente a el aprendizaje profundo (mayormente conocido como Deep Learning), como es el caso de las redes convolucionales y redes recurrentes.

3.4. Deep Learning

Vivimos en una era donde la información presente en Internet, ha llegado a cantidades exorbitantes, tanto así que han aparecido últimamente conceptos nuevos como el de Big Data, término que describe el gran volumen de datos, tanto estructurados como no estructurados, cuya manipulación es fundamental para poder extraer conocimiento de ellos. Pero este análisis no es sencillo de realizar, labores que para nosotros los seres humanos son sencillos de hacer, como reconocer el animal en una foto, son muy complejos de hacer por medio de un programa, esto dado que es prácticamente imposible codificar un software que sea capaz de detectar las características propias de los animales para poder diferenciarlos, y ser usado en cualquier imagen. Es aquí donde el Deep Learning (Aprendizaje profundo) toma un rol importante.

Deep Learning es un concepto dentro del área de Machine Learning, el cual a su vez se ocupa de un aspecto de la Inteligencia Artificial, lo que hace que sean términos muy cercanos entre sí, creando confusión en las reales diferencias existentes entre ellos.

La Inteligencia Artificial (IA) es un subcampo de la informática, creada en la década de los 60, que busca solucionar tareas que son sencillas para los seres humanos, pero difíciles para las computadores, es decir, el tratar de realizar tareas tales como el reconocimiento de objetos, sonidos, lenguaje, etc.

Machine Learning o en su traducción, aprendizaje automático es una parte de la IA el cual, tal como dice su nombre, son sistemas que aprenden de forma autónoma a tomar decisiones, sin que estas reglas o mecanismos de elección se encuentren previamente programados. Esto es lo fundamental en esta área (así como también en Deep Learning), ya que, por ejemplo,

siguiendo el caso de los animales, es tan complejo codificar un programa que sea capaz de diferenciar animales, que es menos complejo diseñar un algoritmo que pueda aprender por si mismo, encontrar dichas diferencias y realizar predicciones a partir de ellas. El como aprende, depende mucho de el tipo de algoritmo que se esté utilizando, pero es esto lo que lo diferencia de un término más global como la inteligencia artificial, el cual, no necesariamente aprenderá por si sólo.

Un ejemplo de aprendizaje de un algoritmo de machine learning puede ser por medio de la predicción de imágenes, colocando un caso específico, si queremos que nuestro sistema sea capaz de decir si un gato está o no presente en una imagen, necesitamos entrenarlo con imágenes en donde aparezcan gatos y otras donde no, y decirle explícitamente de la existencia de uno, cuando corresponda. Si una foto en particular, el sistema indica la existencia de uno, siendo que en realidad no lo hay, el sistema es "castigado", para que ajustes sus parámetros internos, y permita que la siguiente vez que se le pregunte por la misma imagen, este diga que no hay uno. Obviamente el programa estará propenso a cometer errores, ya que su nivel de aprendizaje estará sujeto a la cantidad de información que uno le provee al momento de entrenarlo, mientras más imágenes de gatos se le provea al algoritmo durante su etapa de aprendizaje, mejores predicciones podrá realizar.

El aprendizaje profundo por su lado, también es capaz de aprender por si mismo con la ayuda de ejemplos al momento de entrenarlos, pero lo que lo hace particular, es la forma como lo hace, ya que lo realiza haciendo uso de una red neuronal artificial. Estas son estructuras lógicas que intenta de cierta forma asemejarse a la organización del sistema nervioso de los mamíferos, siendo estas compuestas por capas de unidades de proceso, o sea neuronas artificiales, que se especializan en detectar determinadas características que logren cumplir una labor específica, en este caso de estudio, que ayuden a detectar ciertas características de las oraciones que permitan determinar si es positiva, neutra o negativa. La diferencia que existe con las redes neuronales clásicas, es que el Deep Learning aprovecha estas, y las expande formulándoles más capas de neuronas, de allí que proviene el nombre de aprendizaje profun-

do, esto permite tener un mayor grado de abstracción, permitiendo a los algoritmos trabajar de una mejor manera en tareas que hasta entonces, eran muy difíciles de realizar.

A continuación serán detallados los 3 tipos distintos de algoritmos de DL, que serán utilizados en este estudio, estas son las Redes Feedforwards, Convolucionales y Recurrentes, en estas, específicamente las redes LSTM.

3.4.1. Conceptos básicos de Redes Neuronales.

A modo de introducción de las redes neuronales que serán utilizadas durante este estudio, es necesario dar énfasis en los conceptos básicos que tienen en común cada una de ellas. Como se mencionó anteriormente, las redes neuronales intentan de alguna forma imitar el sistema del cerebro humano, esto se debe a que es el mejor modelo de un sistema inteligente capaz de aprender por si mismo. El cerebro es un sistema altamente complejo, que posee aproximadamente 100 mil millones de neuronas en la corteza cerebral, y que forman entre ellas, un entramado de más de 500 billlones de conexiones neuronales, esto quiere decir, que una sola neurona puede llegar a estar conectada a 100 mil de ellas. Imitar exactamente el comportamiento del cerebro humano es una tarea aún imposible de realizar, no tan sólo por lo complejo, si no que aún no existe hardware capaz de manejar tal cantidad de datos.

Si bien, es imposible emular exactamente un cerebro, las redes neuronales intentan aplicar un modelo basados en ellos, pero de una manera más compacta. Es así, como en el año 1943, se presenta el primer modelo matemático de una neurona artificial, la cual fue creada para resolver tareas muy simples. Esta primera aproximación fue un trabajo en conjunto entre un matemático de nombre Walter Pitts, y un psiquiatra y neuroanatomista Warren McCulloch (W. Pitts and W. McCulloch 1943 [14]). Esta no era nada más que una maquina binaria con varias entradas y salidas, pero fue el primer apronte hacia un modelo más elaborado. Fue así como en 1958, se obtiene un modelo más avanzado, en lo que hoy se conoce como perceptrón, la unidad básica de una red neuronal, lo que vendría siendo una neurona artificial (Frank Rosenblatt 1958 [16]). A través de los perceptrones Rosenblatt demostró, que al entrenarlos

usando conjuntos de entrenamientos linealmente separables, se obtenía una solución en un número finito de iteraciones, que conseguía separar correctamente las clases representadas por los patrones del conjunto de entrenamiento.

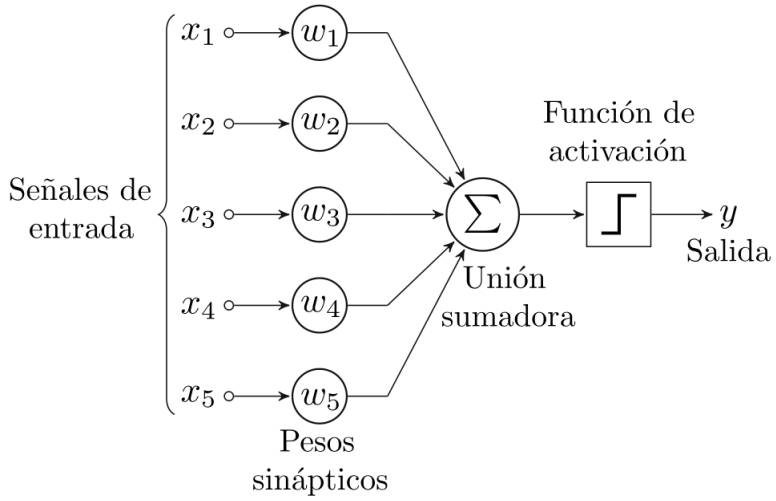


Fig. 3.1: Diagrama de un perceptrón con cinco señales de entrada.

En la figura 3.1 es posible apreciar la imagen de un modelo básico de neurona artificial, donde cada módulo dentro de ella, posee una analogía directa con las neuronas biológicas. Mirada de izquierda a derecha, primero se encuentran las señales de entradas (indicadas con x_i) , referenciando al axón en una neurona biológica, siendo esta una prolongación de las neuronas, cuya función no es otra que conducir los impulsos nerviosos desde el cuerpo de una neurona a otra. Su propósito es el mismo, pero la señal que conducirá, dependerá del problema que se este estudiando.

Además de las señales de entrada, están presente los pesos de conexión (Indicados como w), siendo su símil los pesos sinapticos, o sea la zona donde se transmite la información entre dos neuronas. Estos pesos, en su caracterización como neurona artificial, permitirá darle prioridad a ciertas entradas en desmedro de otras, haciendo que ciertas entradas entreguen mayor información que otras, estos parámetros son modificados constantemente durante el proceso de aprendizaje de la red. La "Unión sumadora" es el centro de una neurona artificial,

llamada Soma (Cuerpo central de una neurona biológica). En una artificial, su función es básicamente la de sumar todos los resultantes entre la multiplicación de una entrada por su peso correspondiente. Dentro del cuerpo, además es utilizada un parámetro llamado Bias, el cual actúa como un valor base a la sumatoria, este también puede ser añadido como una señal de entrada extra de valor uno, con un valor de peso W_0, el cual actuará como Bias.

Inmediatamente después del cuerpo, se encuentra el "Cuello del axón", el cual en redes neuronales artificiales, se le denomina como función de activación. Esta tiene como labor principal mantener los valores dentro de un margen manipulable y generar una respuesta no-linieal, acotando la salida final de la sumatoria entre ciertos valores, evitando que estos aumenten en demasía mientras "fluyen" por la red. Las funciones más utilizadas pueden ser observadas en la figura 3.2. La utilización de cada una de ellas, dependerá explícitamente del problema, ya que puede ocurrir que incluso una función identidad, es decir que su salida es exactamente igual a su entrada, funcione de mejor manera que una que acote mucho más su salida.

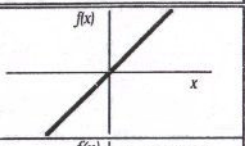
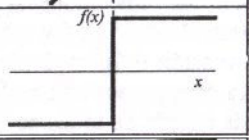
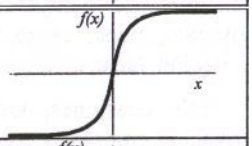
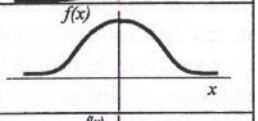
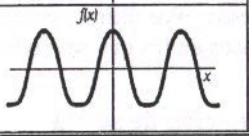
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Fig. 3.2: Funciones de activación habituales.

Finalmente la representación matemática de un perceptron, puede ser observada en la función de la izquierda (3.1) utilizando el bias como un factor a parte, o agregando el bias como un factor de entrada de la neurona (función de la derecha).

$$Y = f(\sum_{i=1}^{n} w_i x_i + b) \equiv f(\sum_{i=0}^{n} w_i x_i) \tag{3.1}$$

La unión de varios perceptrones dio inicio a una nueva forma del estudio de inteligencia artificial, al ser considerada ya no sólo un perceptrón como pilar de computo si no una serie de ellos agrupados en una capa (Minsky and Paper 1969 [17]), dando lo que hoy es conocido como redes neuronales.

Teniendo ya la base de lo que se refiere una red neuronal, a continuación se detallarán algunos conceptos que son utilizados en un estudio de aprendizaje con Redes Neuronales.

Clases: Una clase representa una agrupación de elementos que reúnen características similares y que a partir de ellas, puede diferenciarse de otras clases. En el caso de redes neuronales, a partir de dichas características propias de cada una, la red podrá predecir a qué clase pertenece un objeto determinado. La cantidad de clases dependerá del tipo de problema a cual se vea enfrentado, mientras mayor diferencias existan entre ellas, mejores resultados serán obtenidos.

Señales de entradas: Se consideran señales de entradas a los datos con que la red será alimentada, el formato de estas entradas dependerá exclusivamente del tipo de problema que se esté resolviendo. Por ejemplo, en clasificación de imágenes cada una de las señales de entradas podrían ser cada uno de los pixeles de la imagen y por cada pixel, en su descomposición de color RGB, lo que hace que la cantidad de datos de entrada sea muy grande, tanto más, mientras mayor sea la imagen.

Pesos: Representados por W_{ij}, son la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i. Estos valores son parte fundamental de una red neuronal, ya que la modificación de ellos durante el proceso de entrenamiento implicarán que una red mejore su desempeño. Pero no tan sólo afecta en cómo estos cambian, si no también en cómo sus valores son inicializados. Por ejemplo, la utilización de parametrización de los pesos para garantizar una mejora la velocidad de convergencia de la red (T. Salimans and Diederik P. Kingma [23]), o métodos de iniciación denominados adaptativos (Derrick Nguyen and Bernard Widrow [24]). Uno de los métodos más utilizados en problemas de clasificación, es la iniciación de los valores con valores aleatorios según la distribución Gaussiana, la cual ha mostrado ser una muy buena vía para comenzar el entrenamiento de una red (R. Giryes, G. Sapiro and A. Bronstein [25]).

Dropout: Factor que influye en que tan probable es que una neurona no sea tomada en cuenta durante un batch de entrenamiento. Esta metodología es utilizada para evitar de cierta

manera un Overfitting (o sobre entrenamiento), el cual produce que una red se adapte mucho a sus datos de entrenamiento, y no sea capaz de extrapolar sus predicciones sobre datos nuevos. El Dropout limita esto, haciendo que en cada proceso de entrenamiento, existan neuronas que no afecten en el resultado final, lo que produce que en cierta medida la red tienda a modificar constantemente los pesos de cada neurona en vez de modificar siempre las mismas (N. Srivastava, G. Hinton, A. Krizhevsky, Ilya Sutskever, R. Salakutdinov [19]). En la figura 3.3 se puede apreciar visualmente el cambio que se produce en una red al utilizar un factor de dropout. En una red donde no se aplica dropout, todas las neuronas serán tomadas en consideración en cada batch de entrenamiento (figura de la izquierda), mientras que una con dropout, durante un batch de entrenamiento habrán neuronas que son "apagadas", es decir que no afectarán durante ese periodo, por lo que todas las conexiones hacia ellas, no actuarán (figura de la derecha).

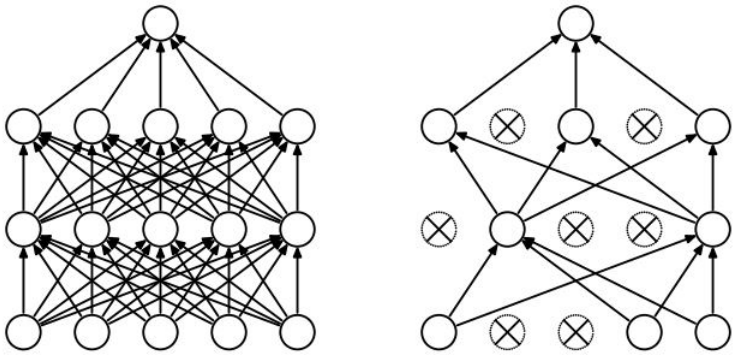


Fig. 3.3: Red Neuronal con Dropout. A la izquierda una red neuronal con 2 capas ocultas. A la derecha un ejemplo de la misma red pero aplicando un dropout.

Funciones de Activación: Como se mencionó anteriormente, las funciones de activación actuarán inmediatamente después de que las entradas sean sumadas y permitirá que la salida de cada neurona se mantenga dentro de ciertos márgenes. Esto hace que los valores internos de una red no aumenten significativamente en su magnitud mientras se aumenten las cantidades de capas. Algunos tipos de funciones utilizadas comúnmente pueden observase en la figura 3.2.

Learning-Rate (Tasa de Aprendizaje): Factor, mayormente denominado η que influye en qué tan rápido son realizadas las actualizaciones de cada peso, ellos son adaptados por medio de Back-Propagation. Un valor muy grande puede hacer que la red nunca encuentre los valores que minimicen su error, haciendo que los cambios de pesos oscilen constantemente. Por el contrario, un valor pequeño garantizará encontrar los valores que satisfacen un error mínimo, pero aumentando el tiempo que le tomará en encontrarlos, en resumen, haciendo que el aprendizaje sea más lento.

Batch de entrenamiento: Es un tipo de metodología utilizado para entrenar las redes neuronales, el cual implica que los pesos de cada neurona serán actualizados después de que la red haya sido entrenada en una cierta cantidad de ejemplos (llamados batch de entrenamiento). Esto implica que al considerarse un promedio de error entre esas entradas, el proceso de aprendizaje siempre llevará aquella dirección, la cual debiese llegar más rápido a un óptimo global. Caso contrario ocurre en el modo on-line, donde cada actualización del peso, se realiza inmediatamente después de un dato de entrenamiento, haciendo que ese error pueda cambiar de dirección (de signo), cada vez que se introduzca un nuevo dato, impidiendo que la red aprenda más rápido debido a los constantes cambios de dirección. Sobre cuál método es mejor, es difícil de dar a conocer a priori, pero hay estudios, que indican que el modo on-line incluso puede llegar a ser más rápido que el entrenamiento por batch (D. Randall and Tony R. Martinez [26]).

Epochs (Épocas): Este tipo de algoritmos son iterativos, es decir, necesitan de repeticiones para poder obtener mejores resultados y llegar a un óptimo global por medio de la variación de parámetros internos de la red. La época se denomina al momento en que todo un dataset es pasado por el proceso de entrenamiento de la red, la cual, dependiendo de la cantidad de batch de entrenamiento, será la cantidad total de iteraciones que realizará el algoritmo hasta alcanzar una época. Si el dataset tuviera unos 10.000 comentarios, y la cantidad de ellos que se utilizarán en un batch es de 500, el total de iteraciones será 20, lo que equivale a una época.

En teoría, mientras más iteraciones de aprendizaje se realicen, mejores resultados se esperaría, pero esto no es siempre así. Normalmente al utilizar un número alto de épocas al momento de entrenar una red, producirá que la red memorice los datos, impidiendo que ella pueda extrapolar la información a nuevos datos (Problema conocido como Overfitting, el cual se hablará a continuación). Esto hace que el número óptimo de ellas, dependa en gran medida al desempeño de la red, a qué tan rápido o lento ella aprenda. Por lo que si bien la cantidad de epochs indica cuantas iteraciones totales realizará la red antes de que termine su entrenamiento, este no debe ser el parámetro que indique que el modelo ha alcanzado su óptimo, si no que debe ser la persona que deba detener su funcionamiento de acuerdo a los resultados después de cada iteración (o cada epoch).

Overfitting (Sobre-Entrenamiento o sobreajuste): La función de una red neuronal es la de poder generalizar, es decir, predecir resultados en casos distintos a los casos con que fue entrenada. El Overfitting es el caso contrario, la red con un sobreajuste no podrá hacer uso de la generalización, puesto que se ha ajustado muy bien a pronosticar sus datos de entrenamiento, impidiéndole trabajar bien con datos distintos a ellos. Esto puede deberse a múltiples factores, los más comunes están ligados a la cantidad de datos con los que la red es entrenada, si estos son pocos, ella no podrá generalizar bien, ya que los ejemplos no abarcarán la cantidad total de distintas alternativas. En este caso de estudio, para poder garantizar de que la red sea capaz de detectar que una frase es negativa, el mejor camino es de poder entregarle la mayor cantidad de ejemplos a la red, de lo que es verdaderamente un comentario negativo, mientras más sean, mejor. Otro factor es el tiempo que se le deja a la red entrenarse, mientras más tiempo se entrene, esta irá memorizando los datos de entrenamiento, haciendo que se produzca nuevamente un overfitting. Un ejemplo de ello puede observarse en la figura 3.4, donde existen dos lineas que separan ambos grupos (puntos azules y rojos), la verde es un caso de overfitting, puesto que se encuentra muy adaptada a la situación presente, pudiendo generar errores de predicción cuando se trabaje con datos fuera de los expuestos. Lo contrario ocurre con la linea negra, la cual si bien no separa correctamente todos los puntos, si logrará trabajar de mejor manera con nuevos puntos.

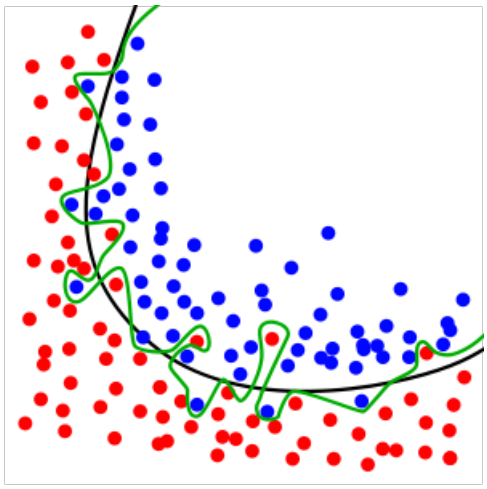


Fig. 3.4: Ejemplo de Overfitting en clasificador binario (linea verde). En linea negra se muestra una función que sería capaz de generalizar sus resultados.

Dada la gran dificultad de poder poseer una gran cantidad de datos de entrenamiento y evitarse este problema, es que han sido ideados distintos métodos que intentan evitar este caso. Como lo son el uso de dropout el cual permite que todas las neuronas mejoren su desempeño durante el entrenamiento (N. Strivastava, G. Hinton, A. Krizhevsky, I. Suskever, R. Salakhutdinov [19]). O con el uso de distintos métodos de entrenamiento, como el uso de cross-validation (Andrew Y. Ng. [20]). El cambio en la forma de tratar distintos parámetros de la red, como el bias (Gaving C. Cawley and Nicola L. C. [21]). O cambios en la forma en que el error es propagado por la red, haciendo uso de gradientes conjugados y métodos que ayuden a predecir el momento exacto en el cual se puede detener el periodo de entrenamiento (Early Stopping), anulando el problema de el sobreajuste (R. Caruana, S. Lawrence and Lee Giles [22])

Capa de Entrada: Corresponde a la primera linea de una red, aquella capa de neuronas que se encuentra directamente frente a las señales de entradas con que será alimentada. El número de neuronas dependerá del tipo de red que se esté utilizando y del problema al que se este enfrentando. Estas además pueden variar dependiendo de la metodología que se esté

utilizando para tratar la información entrante.

Capa de Salida: Corresponde a la última linea de la red, por la cual se obtendrá la predicción final, esto también puede variar dependiendo del tipo de problema que se esté resolviendo, pero en métodos de clasificación, las salidas son más estándar. En su mayoría, la capa de salida tendrá tantas neuronas como tipos distintos de clases existan. La activación de cada una de ellas, dependerá de la metodología que se esté utilizando, la más directa, es que las neuronas se activarán más (su valor será más grande) de acuerdo a la probabilidad que tenga la clase correspondiente a esa neurona, esto posee el inconveniente de que las comparaciones entre ellas, no serán bajo una misma proporción, es decir, la diferencia que exista entre cada uno de los valores, no indicará cuanto más probable es una de otra, si no que sólo dirá, de acuerdo a cual es la mayor, la más probable.

Fig. 3.5: Ejemplo de softmax, donde un vector es llevado su equivalente en una distribución de probabilidad.

Un enfoque distinto es la de tratar esta última capa con una funcion softmax (Función exponencial normalizada), la cual entrega una distribución de probabilidad sobre las K diferentes clases (Figura 3.5). Es decir, cada neurona asociada a una clase, entregará la probabilidad de que sea esa clase (entre un valor de 0 y 1, siendo 1 el valor de un 100 %), por lo que la suma de las salidas siempre tendrá un valor igual a 1, representando todos los casos posibles. Un ejemplo de esto puede observarse en la figura 3.5, la cual representa la transformación de un vector de salida, hacia un vector asociado a la distribución de probabilidad de él. La ecuación asociada a esta distribución puede verse en (3.2), donde la probabilidad de la neurona i, se calcula como el valor de entrada dicha neurona i, dividida en la suma de todas las neuronas

de salida. Siendo $z = (z_1, z_2, ...z_k)\epsilon\, \mathbb{R}^k$ el vector de entrada, donde k depende del número de clases a clasificar.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}} \tag{3.2}$$

Capa Ocultas: Se consideran como capas ocultas a todas aquellas que se encuentran entre la entrada y salida. El número de capas ocultas dependerá del problema, también así su configuración interna, ya que distintos modelos de redes neuronales poseerán también distintas configuraciones. Un ejemplo de ellas puede apreciarse en la figura 3.6, la cual corresponde a una red denominada Feedforward, donde todas las neuronas se encuentran conectadas. En este ejemplo, todas las capas ocultas poseen el mismo número de neuronas, pero esto no necesariamente debe ser así. El número de neuronas por capa puede variar, no existe por ahora un método que permita predecir de ante mano cual debe ser la cantidad óptimas de ellas para obtener un mejor resultado, todo ello dependerá del análisis de resultados a posterior.

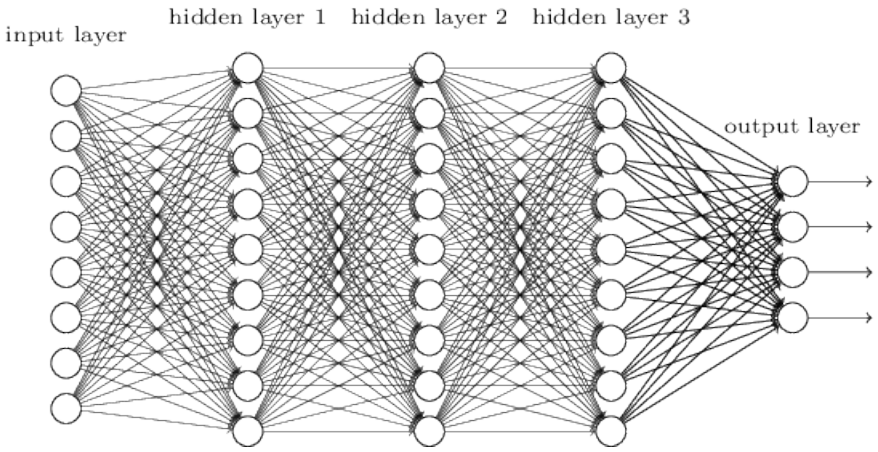


Fig. 3.6: Ejemplo de red con capas ocultas.

Aprendizaje Supervisado: Es aquel proceso de aprendizaje que se realiza mediante un entrenamiento controlado por un agente externo, denominado supervisor, el cual determina la respuesta o salida que deberá predecir la red a partir de una sucesión de datos de entrada. En caso de que la red determine una salida que no corresponda a la deseada, el supervisor pro-

cederá a modificar los pesos de las conexiones internas, con el propósito de lograr conseguir que la salida obtenida sea la que realmente se desea obtener.

Función de costo o error: Se denomina función de costo o error, a aquella utilizada en obtener numéricamente la diferencia existente entre el valor esperado y el predicho por la red. Es a partir de este dato que la red comienza a modificar sus valores internos (Por ejemplo por medio de algoritmos como el Back-Propagation, que se detallará más adelante) para entregar una mejor respuesta. El valor final del error, dependerá del tipo de función, la cual a su vez también depende del tipo de problema al cual se esté enfrentando, ya que no todas ellas podrá obtener un error acorde al tipo de información que se desea predecir.

Ejemplo de funciones de costo.

Error Cuadrático: También conocido como error cuadrático medio, máxima verosimilitud y suma cuadrada de errores:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}\left\|y_i - a_i^L\right\|^2 \tag{3.3}$$

Donde n es el total de número de neuronas de la última capa; la suma es sobre el error existente entre cada una de ellas, x; y_i es la correspondiente salida deseada de la neurona de salida i; L denota el número de capas de la red (ya que se realiza en referencia a la última capa); y a_i^L es la respuesta de la neurona de salida i. Por lo que el error asociado a un ejemplo de entrenamiento, puede ser el promedio de el error individual de cada neurona de salida.

Cross-Entropy La función de Cross-Entropy se define como:

$$CE = -\frac{1}{n}\sum_{i=1}^{n}[y_i\, lna(x_i) + (1-y_i)ln(1-a(x_i))] \tag{3.4}$$

Donde n es el total de ejemplos de entrenamiento durante un batch; $x = (x_1, x_2, ...x_n)$

ejemplos de entrenamiento en un batch, $y = (y_1, y_2, ...y_n)$ corresponde a la salida deseada para cada ejemplo, y $a(x_i)$, corresponde a la salida de la red para cada ejemplo, representada por la ecuación (3.1).

Back-Propagation: Es un metodo usado en redes neuronales artificiales para calcular la contribución de cada una de las neuronas en el error final de la red después de que un batch de datos ha sido procesado (D. Rumelhart, G. Hinton and R. Williams [18]). Back-Propagation hace comúnmente uso del gradiente descendente [1] para optimizar el peso de cada una de las neuronas al calcular el gradiente de la función de pérdida. Su nombre proviene del echo de que una vez que la red ha calculado el error de su predicción con respecto al valor esperado, ella comienza a propagar este dato hacia atrás, haciendo que todas las neuronas se modifiquen los valores de sus pesos con respecto a que tanto han influidos ellas en el error final.

Gradiente Descendente: Es un algoritmo iterativo de optimización de primer orden el cual busca el mínimo de una función. Para esto, el gradiente descendente toma pasos proporcionales al negativo del gradiente (o su aproximación) de la function en su punto actual.

3.4.2. Redes Feedforward

Este tipo de redes es una de las más útilizadas, y a su vez la más simple, también son conocidas como Perceptrón Multicapas. Por si sola no permite realizar tareas muy complejas, pero dada su potencial en clasificación, es que es utilizada en conjunto con otro tipos de redes, como por ejemplo las redes convolucionales, que al recibir los datos proveniente de las salidas de estas, permiten obtener un mejor resultado en clasificaciones de clases.

Su nombre proviene de la forma en que estas funcionan (Feed: Alimentar y Forward: Adelante), ya que todos los datos que llegan a las entradas de la red, se mueven siempre hacia adelante, "alimentando" cada una de las neuronas en las capas continuas. En ellas además no existen ciclos ni bucles entre sus capas, por lo que el flujo de datos siempre se dirigirá hacia

[1] Algoritmo iterativo de optimización, que permite encontrar el mínimo de una función.

la salida. Un ejemplo clásico de este tipo de redes, puede observarse en la figura 3.7, la cual en este caso, corresponde a una red feedforward de 3 capas, con 3 neuronas de entrada, 2 de salida y 4 en la capa oculta.

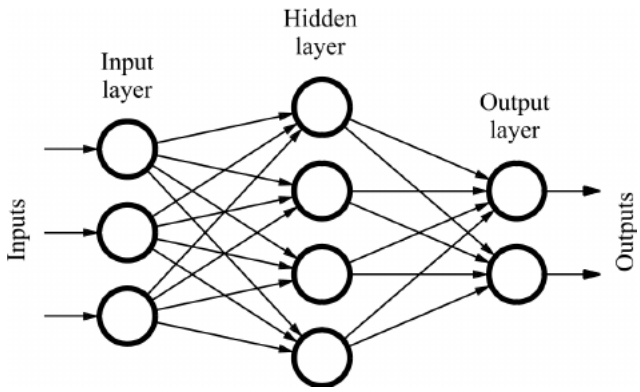


Fig. 3.7: Red neuronal Feedforward de 3 capas

El proceso de aprendizaje viene dado por el uso del algoritmo de de Back-Propagation. Una vez que un batch de entrenamiento es "pasado" por la red, es decir, una cantidad fija de ejemplos de entrenamiento es introducida por la red, se calcula el error promedio de cada uno de ellos de acuerdo a una función de costo establecida. La base del algoritmo de Back-Propagation es la de asumir que este error es producto de la acumulación errores provenientes de cada una de las neuronas, por lo que, para revertir esto, es necesario encontrar cual fue el aporte de cada una en el error final de su predicción.

Para esto es posible utilizar un proceso iterativo de gradiente descendente, donde:

$$\nabla E = (\frac{\partial E}{\partial w_1^l}, \frac{\partial E}{\partial w_2^l}, ..., \frac{\partial E}{\partial w_n^l}) \tag{3.5}$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i^l} \qquad \text{para } i = 1, ..., n \tag{3.6}$$

$$w_i^{l'} = w_i^l - \eta \frac{\partial E}{\partial w_i^l} \qquad \text{para } i = 1, ..., n \tag{3.7}$$

De la ecuación 3.5 se determina el error asociado en cada uno de los pesos de la neurona n en la capa número l. El cómo obtener exactamente estos valores, escapa del propósito del presente estudio, por lo que el detalle de la metodología detrás de esto, puede ser encontrado en bibliografía ([31]). Una vez obtenido cada uno de ellos, es posible obtener el delta de actualización que afectará al actual valor del peso de una neurona (Ecuación 3.6). El valor de η representa el "Learning Rate", o la tasa de aprendizaje de la red, es decir, que tan rápido o lento se realizará la actualización de los valores. Finalmente son sumados, y así obtener el nuevo valor del peso de dicha neurona (Ecuación 3.7), con el cual debiera afectar de tal manera, que el error asociado a el mismo batch de entrenamiento con el que se obtuvo su error, debiese disminuir.

Este tipo de redes es bastante utilizado como parte final de otros sistemas o redes, ya que su funcionamiento permite clasificar de muy buena forma distintas clases. Por ejemplo, una red Convolucional (De la cual se hablará acontinuación) no es capaz de clasificar por si sola, por lo que en la mayoría de las vece es acompañada en su salida, por redes FeedForward, para que suplan esta deficiencia y permitan obtener un buen funcionamiento.

3.4.3. Redes Convolucionales

Las redes convolucionales son muy similares a las redes Feedforward en su estructura y funcionamiento, ambas se componen de neuronas asociadas con pesos y biases que pueden ir mejorándose con el pasar de las iteraciones. Al igual que el perceptrón multicapas cada neurona de la capa de entrada recibe algunas entradas donde se realizan productos escalares pasando posteriormente por una función de activación, finalizando con una capa de salida la cual está asociada a una función de pérdida o costo, como lo puede ser la función softmax.

La diferencia radica en que las redes convolucionales se han adaptado para analizar imágenes (Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. [32]), lo que permite de cierta manera modificar ciertas propiedades en la arquitectura de las redes multiperceptrones, lo que permite poder mejorar la eficiencia y reducir la cantidad de parámetros de la red que

deben ser actualizados. Esta especialización también se encuentra inspirada en sistemas de la naturaleza, exactamente en la corteza visual de los animales, la cual en fracciones de segundo es capaz de identificar objetos dentro de su campo de visión, identificando de ellos su profundidad, sus contornos, etc. La corteza visual contiene una disposición jerárquica compleja de neuronas, donde por medio de etapas, se ocupan de obtener características distintas, por ejemplo, las neuronas dentro del área visual primaria, llamada V1, son las encargadas de las características visuales de bajo nivel, como pueden ser los contornos, información básica de contraste, de colores, entre otros. Para posteriormente alimentar, por medio de V1, áreas distintas como V2,V4,V5, las cuales se encargan de extraer aspectos muchos más específicos.

Las redes neuronales convolucionales funcionan de la misma manera, distintos niveles de capas estarán encargadas de extraer distintas características de la imagen, donde a medida que se encuentre a más profundidad la capa, podrá analizar propiedades más específicas. Por ejemplo, las primeras capas pueden percibir bordes, o sea se especializa en establecer patrones de detección de bordes, las siguientes quizás de extraer información con respecto a las tonalidades, iluminación,etc. Para finalmente a partir de toda la información recopilada de una imágen de entrada, realizar su predicción, dependiendo del tipo de problema.

Como se ha mencionado, la estructura se ve modificada, y adaptada para trabajar con imágenes, por lo que existen nuevos conceptos, uno de ellos son las llamadas capas convolucionales, con stride y padding, y capas de pooling, las cuales se explicarán a continuación:

Capa Convolucional

Es en la cual se utiliza el operador matemático llamado convolución, el cual transforma dos funciones f y g en una tercera función que representa la magnitud en la que se superponen f y una versión trasladada e invertida de g. En la práctica esto se realiza por medio de filtros que actuarán como la función g, o sea haciendo convoluciones en ciertas secciones de la imagen original, que actúa como función f, siendo expresado para el caso de imágenes, como $f(x,y) * g(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)g(x-m, y-n)$, donde $x = 0, 1, 2, 3..., M$ e $y = 0, 1, 2, 3..., N$.

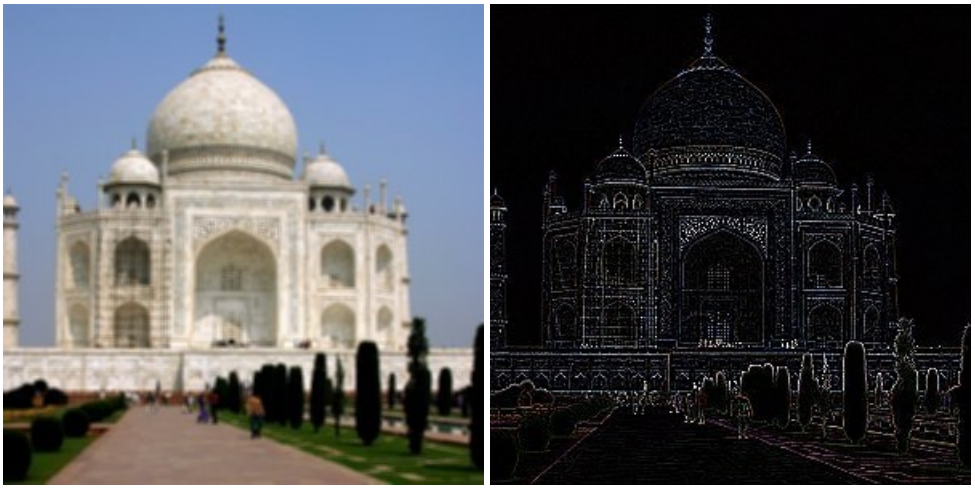


Fig. 3.8: Ejemplos, a) imagen de la izquierda sin filtro, b) imagen de la derecha con filtro de detección de contorno.

Estos filtros son matrices que irán extrayendo distintos tipos de información de una imagen, mientras se realizan convoluciones con ella . En términos prácticos esta representa el producto Hadamard[2] entre cada matrices. Por ejemplo, si se tiene la imagen a) de la figura 3.8 y se le aplica la siguiente matriz como filtro, se obtiene como resultado, la imagen b) de la figura 3.8.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.8}$$

Una capa convolutional, puede tener una gran cantidad de filtros (o también llamados kernels) que actuarán a lo largo de toda la imagen por secciones de igual tamaño que el filtro, es decir, si el filtro es de tamaño 3x3, sólo tomará secciones de ese tamaño. El movimiento del filtro por la imagen de entrada, vendrá dado por un factor llamado stride, stride de valor 2 indicará que la siguiente convolución se realizará con el filtro desplazado en 2 pixeles desde

[2] Para dos matrices A,B de misma dimension mxn, el producto Hadamard, $A \odot B$ es una matriz de misma dimension, cuyos elementos vienen dado por: $(A \odot B)_{i,j} = (A)_{i,j}(B)_{i,j}$.

el punto anterior. Esto puede acarrear ciertos inconvenientes no deseados, por ejemplo, si se tiene una imagen de tamaño 32x32, y se aplica un filtro convolucional de tamaño 5x5,y stride 1, la imagen resultante tendrá una dimensión 28x28. Normalmente se busca que el tamaño de las imágenes permanezcan constantes después de una capa convolucional, por lo que para no producir este efecto, es que existen ciertas metodologías como aplicar un zero-padding, lo que no es más que agregar ceros alrededor de la imagen, aumentando su tamaño, para que el efecto de reducción en dimensionalidad al aplicar filtros, no la afecte. Por lo que para el caso anterior, es necesario aplicar un zero-padding de tamaño 2, como puede observarse en la figura 3.9, para que se mantenga en un tamaño 32x32.

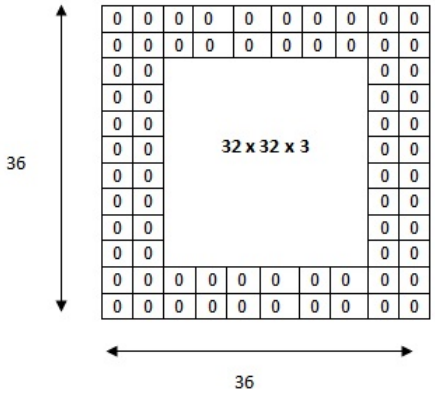


Fig. 3.9: Imagen con zero-padding de tamaño 2.

Al pasar una imagen por una capa convolucional con una gran cantidad de filtros, permitirá entre otras cosas captar distintas características propias de ella, que permitan poder cumplir con un fin específico.

Capa de reducción o Pooling

Tal como su nombre lo indica, su utilidad principal radica en la reducción de las dimensiones espaciales del volumen de entrada para la siguiente capa convolucional. Esto afecta sólo a su ancho y alto, y no a su profundidad, en imagenes RGB o de tipos similares.

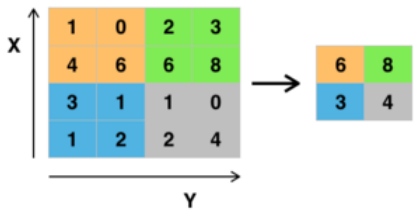


Fig. 3.10: Ejemplo de aplicar un Max-Pooling de tamaño 2x2 a una matriz de tamaño 4x4. Resultando una reducción del Tamaño.

Existen distintos tipos de pooling, siendo el más utilizado la capa Max-Pooling, la cual, divide a la imagen de entrada en un conjunto de rectángulos donde, en cada uno de ellos solo se mantendrá el pixel con mayor valor, reduciendo la imagen a un tamaño acorde al tamaño de la capa Max-Pooling. Un ejemplo puede observarse en la imagen 3.10, la cual lleva una matriz de tamaño 4x4 a una de tamaño 2x2, al realizar un Max-Pooling de tamaño 2x2.

Capa de Clasificación

La capa de clasificación es la última fase de una red convolucional, la cual a partir de toda la información recopilada por medio de las capas anteriores, es decir, de reducción y convolucionales, debe poder predecir de la mejor manera. Básicamente es una red feedforward que es anexada a la salida donde su capa de salida tendrá tantas neuronas como tipos de clases distintos, en la cual se aplica una función softmax, para que la respuesta de cada neurona, esté en relación a la clase predicha por la red.

Diagrama general de una red convolucional

El conjunto total de las capas, puede ser representado por la figura 3.11, donde a partir de una imagen, se quiere predecir si es que en ella se encuentra un perro, gato, pájaro o un bote. Esta estructura puede llegar a ser más profunda si es que en vez de 2 capas de convolución, existiesen más, permitiendo encontrar aún más características que permitan mejorar una predicción.

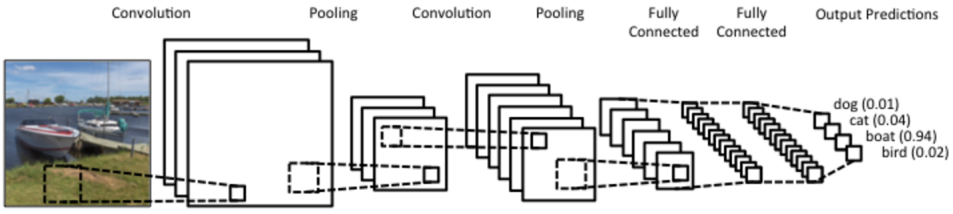


Fig. 3.11: Red neuronal convolucional completa.

Red Convolucional sobre texto

Hasta ahora se ha presentado este tipo de redes sobre imágenes, dado que fue para dicho propósito que fueron diseñadas, y su explicación es más fácil de realizar. Pero estas también son posible de utilizar para analizar o clasificar texto, pero para ello es necesario reformular de cierta manera la forma en que se presentan las palabras. Esta transformación debe ser realizada de tal manera, que su representación final pueda asemejarse a una imagen, es decir, que su estructura sea cuadrada, o en su defecto rectangular.

Una alternativa de transformación, es la utilización de un diccionario de letras, que representará a todas las que serán posteriormente analizadas, cualquier letra fuera del diccionario, tendrá una representación fija. La posición de una letra en particular dentro del diccionario, permite obtener un One Hot Vector, es decir, un vector rellenado de ceros excepto en la posición de la letra dentro de dicho diccionario. Un ejemplo de esto puede apreciarse a continuación:

$$\text{“}abcdefghijklmnopqrstuvwxyz0123456789 : () = *, .-_{[}]@?/ <?\text{”} \tag{3.9}$$

Por ejemplo, la representación en “one hot vector” de la letra d, será representada por el vector 3.10(notar que antes de la letra a en el diccionario, existe un espacio vacío).

$$[0, 0, 0, 1, 0, 0, 0, 0..., 0, 0, 0, 0] \tag{3.10}$$

Si bien el diccionario expresado en 3.9 parece lógico, ya que sigue el orden del abecedario español, además de agregar los números en orden ascendente junto además a distintos caracteres, esta puede que no sea la forma óptima de ordenarlos. Como se verá más adelante, estos "one hot vector" son filtrados de acuerdo a ciertos parámetros, que influirá directamente en cómo estos vectores estan armados, por lo que en conclusión, el orden de las letras dentro del diccionario no es trivial, mejores resultados se pueden obtener al modificar el diccionario. Por ejemplo, si dentro del idioma español, las letras "o" y "n" se encuentran más frecuentemente juntas que "i" y "n", en la confección del diccionario será mejor ubicar las dos primeras letras ("o" y "n") más juntas. Obtener un orden óptimo de las letras, es una tarea difícil de llevar a cabo, por lo que un orden básico, como el expuesto en 3.9, es un buen ejemplo y un buen comienzo.

Al tratarse el presente estudio, en el análisis de comentarios provenientes de Twitter, se sabe de antemano que la cantidad máxima de caracteres permitidos, es de 140, y como la cantidad de letras sobre el diccionario es de 48, la representación final del mapeo de texto es una matriz de 0's y 1's de tamaño 140x49, lo que exactamente no es una representación fiel de una imagen, pero permite que dicha transformación pueda ser analizada con redes convolucionales.

3.4.4. Redes Recurrentes

Otro tipo de redes son las llamadas redes recurrentes, las cuales son muy populares en tareas del procesamiento natural del lenguaje, dado su buen desempeño. Esto se debe a que, a diferencia de las dos tipos de redes anteriores, esta red toma en consideración la secuencia de información de entrada, mientras que para las redes Feedforward y Convolucionales todas las entradas son completamente independientes unas de otras. Es exactamente esta diferencia

que las convierte en modelos idóneos pra realizar trabajos PNL, tales como traducciones de texto (Ej, traducción japonés-inglés. E. Greenstein and D. Penner [33]) , clasificaciones de texto (P. Liu, X. Qiu and X. Huang [34]), predicciones de palabras , transcribir audio a texto (A. Graves, A. Mohamed and G. Hinton [35]), etc. Básicamente las RNN (Por su nombre en inglés, Recurrent Neural Networks) trabajarán bien con cualquier tipo de datos, en los cuales su secuencia tenga importancia.

El nombre recurrente es porque realiza una misma tarea para cada uno de los elementos de entrada, esto quiere decir que todas las células (cada unidad de cómputo, también conocidos como nodos) serán las mismas, pero las respuesta de cada una de ellas, dependerá de cálculos previos y actuales. Básicamente poseen capacidad de memorizar cuales han sido los datos analizados anteriormente (realizado por medio de bucles), para a partir de ellos, y de la información actual entregar un valor a la siguiente neurona.

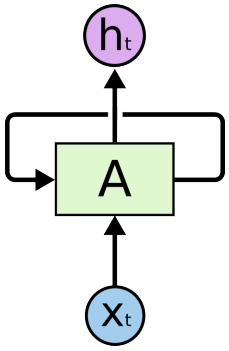


Fig. 3.12: Representación de una red recurrente.

Una representación básica de una célula de este tipo de redes puede apreciarse en la figura 3.12, donde X_t representa la entrada y h_t su salida. En ella se observa que difiere a los perceptrones clásicos, ya que tiene implementada en ella un bucle, el cual le permite poder mantener información de un paso de la red al siguiente. Si se utiliza secuencias de ellas, se obtendrá representada una red neuronal recurrente, como la de la figura 3.13. En ella se aprecia, que a partir que una sucesión de células conectadas contiguamente, donde cada

una de ellas posee una entrada y una salida, la información que antes era retenida por una, ahora esta es pasada hacia la siguiente, para que así obtenga una especie de contexto que relacione su propia entrada, con un estado anterior. Esto permite que la información fluya hacia adelante y permita tomar en consideración esta secuencia de datos.

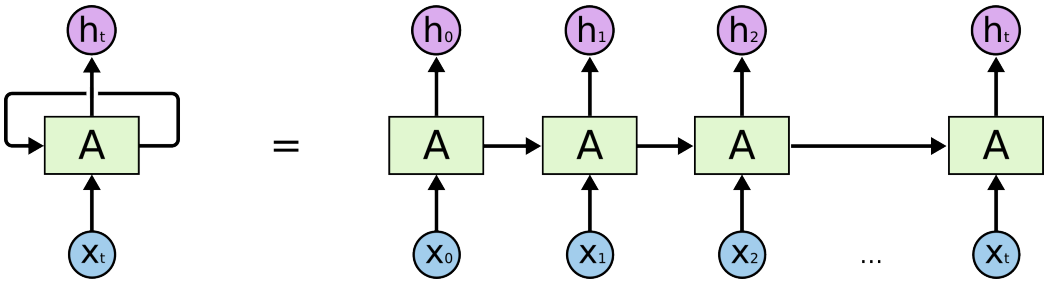


Fig. 3.13: Representación de una célula de red recurrente.

Como se mencionó previamente, este tipo de redes funciona bien en distintos tipos de tareas secuenciales, y es dependiendo sobre qué se desee predecir, que su estructura sufrirá ciertos cambios para adaptarse de la mejor manera al problema en cuestión. Distintas estructuras pueden observarse en la figura 3.14, en la cual, los cuadros azules representan la salida de una neurona, los rojos su entrada, y el verde la propia neurona con su correspondiente paso de información hacia la siguiente (Flecha verde). Mirando las redes de izquierda a derecha,

1. Uno a uno: representa sólo una célula, con una entrada y una salida.

2. Uno a muchos: representa una secuencia de salida, por ejemplo captación de una imagen, es decir, a partir de una foto, construir una secuencia de palabras acorde a ella.

3. Muchos a uno: Forma utilizada en clasificaciones de secuencias, como la de texto.

4. Muchos a muchos: Representan modelos donde a partir de una secuencia de datos, entrega otra secuencia, un ejemplo de ello son las traducciones de texto.

5. Variante de muchos a muchos: Una variante de 4, puede ser utilizada en la clasificación de video, donde cada frame, representada por cada entrada, puede ser etiquetado de acuerdo a su contenido.

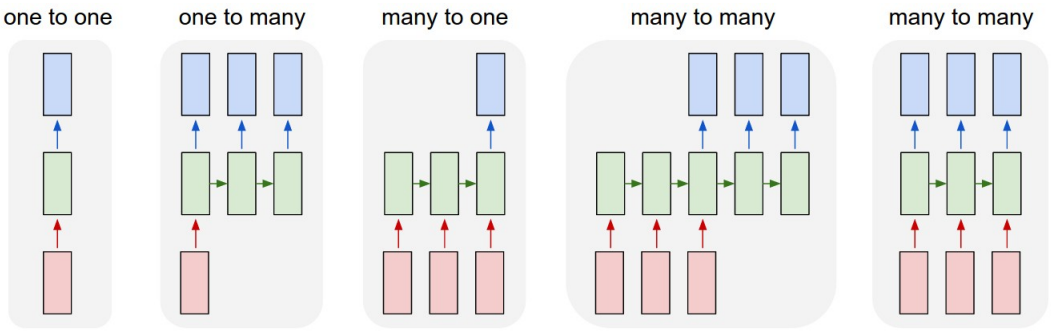


Fig. 3.14: Tipos de forma de una red recurrente, dependiendo del tipo de trabajo que se esté analizando (Andrej Karpathy blog [39]).

Si bien las redes recurrentes permiten mejorar considerablemente los resultados al ser utilizado sobre datos cuya secuencia necesita ser considerada, estas presentan ciertos problemas, que limitan tanto su funcionamiento como el nivel de precisión que pueden llegar a alcanzar.

Cuando una red recurrente tradicional se encuentra en su etapa de back-propagation, cada matriz de peso de una neurona recibe una actualización proporcional al error de su respectiva función de error con respecto al peso actual en cada iteración durante el proceso de entrenamiento. El problema es que en algunos casos, el gradiente comenzará a ser muy pequeño a medida que el peso de la neurona se encuentra más alejada de la salida de la red, haciendo que los pesos de las neuronas más cercanas a la entrada de la red no sean actualizados, impidiendo que la red mejore su desempeño, a este problema se le conoce como "vanishing gradients" o problema de desvanecimiento de gradiente por su traducción directa (R. Pascanu, T. Mikolov and Y. Bengio [40]). También ocurre un problema opuesto al desvanecimiento, llamado "exploding gradients" o problema del gradiente explosivo, el cual hace que la propagación de error durante el back-propagation se hace cada vez más grande a medida que se distribuye por la red, haciendo que la red diverga, y no logre encontrar un estado interno que logre minimizar su error durante el proceso de entrenamiento. Esto último perjudicará en el desempeño de la red, cuando tenga que trabajar sobre nuevos datos, datos distintos a los entrenados.

Uno de los problemas asociado al desvanecimiento de gradiente, es que impide que la red pueda memorizar debidamente palabras (o datos) durante un periodo largo de tiempo, afectando negativamente cuando cierta información se vean influenciados por sucesos anteriores dentro de la serie de tiempo. Si una red recurrente ha sido entrenada para predecir la siguiente palabra de un texto, esta no tendrá un buen desempeño si es que la palabra a predecir, tiene insidencia directa con sucesos explicados al comienzo de él. Un ejemplo, si se quiere predecir la siguiente palabra de "Crecí en Alemania .. (más texto) .. así que hablo fluído " , está claro que se necesita el contexto de que la persona es alemana para determinar que ella habla fluído alemán, pero si la red no es capaz de memorizar el contexto de que ella creció en Alemania, no podrá predecir de buena forma la siguiente palabra.

Redes Long-Short Term Memory (LSTM)

Las redes LSTM o "Long Short Term Memory" (memoria de largo y corto plazo) son un tipo específico de redes recurrentes, capaces de manetener en ella, dependencias de largo plazo, evitando, o más bien, mitigando el problema de desvanecimiento de gradiente.

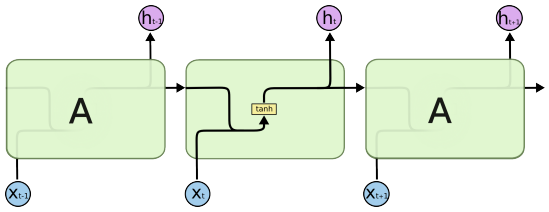


Fig. 3.15: Representación célula RNN.

Este tipo específico de redes recurrentes, conocidas tan sólo como LSTM (S. Hochreiter and J. Schmidhuber [41]) están diseñadas para evitar los problemas de las RNN comunes, permitiendo que la información pueda ser recordada durante largos períodos de tiempo, siendo esto algo intrínsico de este tipo de redes, y no algo que sea necesario entrenar. Para permitir esto, la estructura interna de una célula RNN se ve modificada por una estructura más compleja.

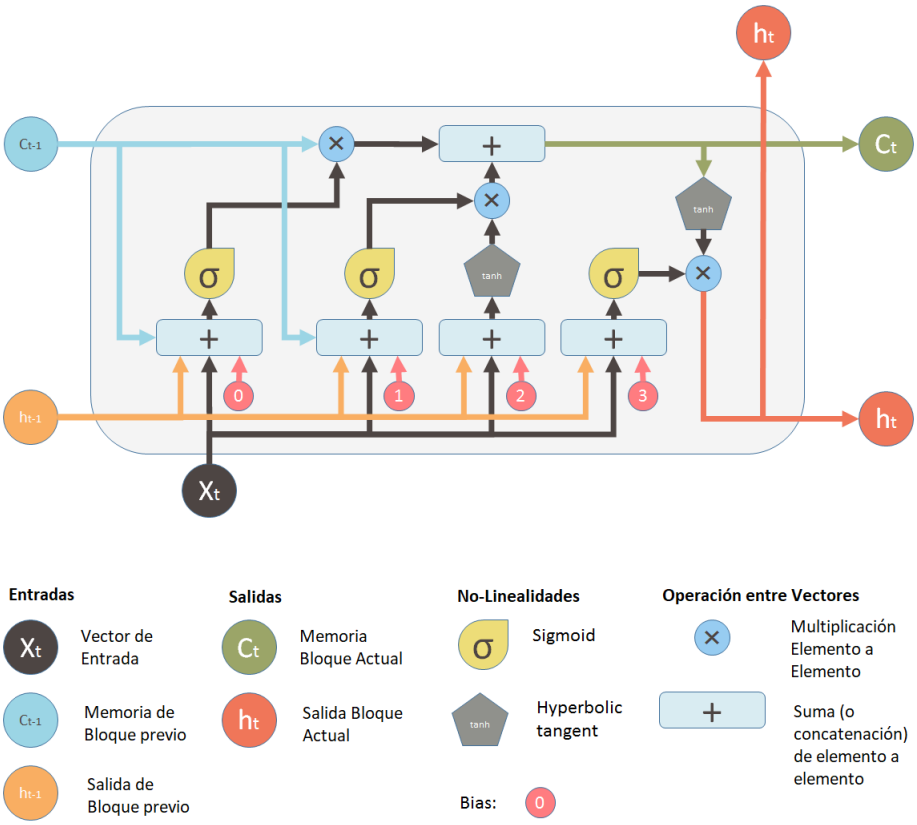


Fig. 3.16: Representación célula LSTM.

A diferencia de una célula de una red recurrente tradicional (Figura 3.15, que tan sólo se ve influenciada por una función de activacuón "tanh" (tangente hiperbolica, la cual lleva los valores de salida de la célula a una salida entre -1 y 1) las células de las redes LSTM contienen una combinación de operaciones entre vectores y compuertas que deciden el flujo interno de datos. Una representación de esta estrucuta, se observa en la Figura 3.16 (diagramas realizados por Shi Yan [42]), donde es posible apreciar los distintos componentes, cuyas funcionalidades serán detalladas a continuación.

Elementos de LSTM.

- Entradas
  - Vector de Entrada: Simbolizado como X_t, representa el vector de entrada perte-

neciente al dato t-ésimo de la serie de tiempo.

- Memoria de Bloque Previo: Simbolizado como C_{t-1}, representa el vector asociado a la memoria, proveniente del bloque (o célula) anterior.
- Salida de Bloque Previo: Representada por h_{t-1}, indica el vector asociado a la salida del bloque anterior.

- Salidas
  - Memoria Bloque Actual: Representada por C_t, indica cual el vector asociado a la nueva memoria de la célula actual.
  - Salida Bloque Actual: Representada por h_t, indica el vector de la salida de la célula, el cual puede ser usado como salida final de la red LSTM, o ser pasada como una nueva entrada del siguiente bloque de la red.

- Funciones no lineales

  Al igual que las funciones de activación presentadas anteriormente, estas cumplen el mismo propósito, que es la de establecer ciertos márgenes de los datos internos. Los utilizados son la función de sigmoid , función que mapea su entrada a un valor entre 0 y 1, y tanh (tangente hiperbólica) que lo hace entre valores de -1 y 1. El bias por su parte, permite, al igual que en las redes Feedforwards, entregar un valor base de operación.

- Operaciones entre vectores
  - Multiplicación: Es una operación entre vectores, que actua elemento a elemento, permite que ciertos valores de un vector puedan ser atenuados o aumentados en su valor, para que su influencia dentro de la red modifique el comportamiento general. Por ejemplo si un valor de un vector es multplicado por 0, quiere decir que su valor no es necesario para satisfacer los requerimientos de la red, por lo que es “olvidado”.
  - Suma: Permite simplemente poder representar como un vector, la interacción de dos o más vectores. Esta suma se realiza elemento a elemento, al igual que la

multiplicación.

Funciones

El funcionamiento de un bloque de LSTM se puede caracterizar bajo 4 funciones principales, las cuales serán detalladas a continuación.

- Mantener Memoria Previa

  En la figura 3.17 se aprecia el canal asociado a la memoria del bloque (o célula) actual, la cual permite mantener cierto porcentaje del vector de memoria del bloque anterior, esta memoria, representada por C_{t-1}, es pasada por el factor X denominada como compuerta de olvido, la cual por medio de una multiplicación de elemento a elemento, permite minimizar o derechamente anular ciertos valores de la entrada C_{t-1}, el factor con el que será multiplicado será detallado más adelante.

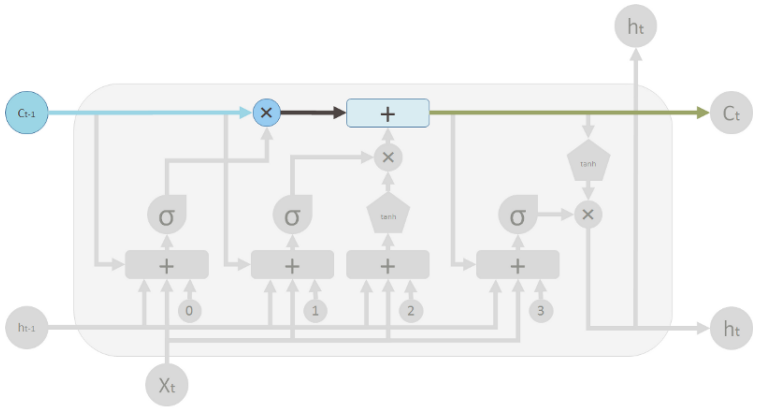


Fig. 3.17: Canal de memoria de un bloque LSTM.

Posteriormente es operada por $+$, el cual permite combinar por medio de la suma de sus elementos, el valor resultante de la memoria anterior, junto con la nueva memoria, asociada al vector de entrada actual del bloque. Que tanto de la nueva memoria será agregada a la memoria anterior es controlada por medio del operador X que antecede, por la parte inferior al factor $+$.

- Control compuerta de olvido

  Lo que permite controlar, que tanto de la memoria anterior debe seguir fluyendo por la

red LSTM, es una simple capa de red neuronal que toma como entradas , la memoria del bloque previo C_{t-1}, la salida del bloque previo h_{t-1}, el vector de entrada del bloque actual X_t y un cierto valor de bias.

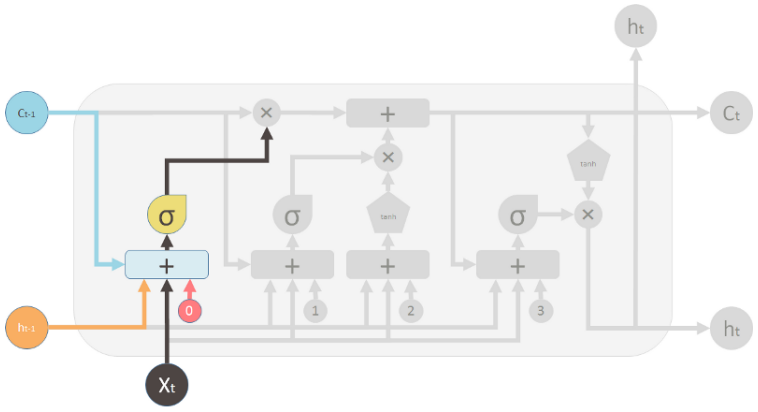


Fig. 3.18: Representación compuerta de olvido.

La estructura interna asociada a la "compuerta de olvido" es posible observarla en la Figura 3.18, donde después de que cada factor de entrada es sumada por la operación $+$, esta es mapeada a través de una función de activación sigmoidal para que sus valores estén entre un factor de 0 y 1. El resultado de esto, permitirá poder controlar que tánto de la memoria del bloque anterior, debe mantenerse en el flujo de la red, para así olvidar todo factor que no ayuda a realizar la función principal.

- Memoria Actual

  La Figura 3.19 muestra los canales involucrados en la obtención del estado de la nueva memoria del bloque, su valor se define por dos redes neuronales simples, una encargada de controlar cuánto de la nueva memoria será sumada al vector asociado a la memoria del bloque anterior y una segunda, encargada de obtener un vector asociado a la memoria actual del bloque, esta última representada con la función de tanh.

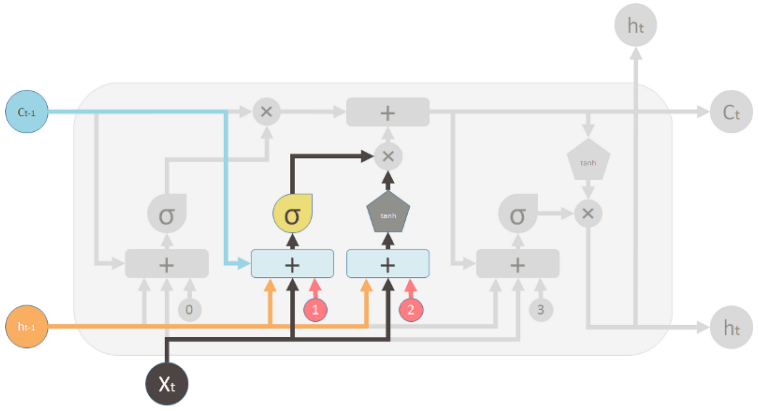


Fig. 3.19: Representación de obtención de nueva memoria asociada al bloque.

Por último, el resultado entre la multiplicación de ambos vectores, permitirá obtener el vector asociado a la nueva memoria del bloque, la cual por medio de una suma de elemento a elemento, será agregada al vector de memoria proveniente de los bloques anteriores. Esta representación puede observarse en la figura 3.20, la cual muestra que el valor resultante, será llevado a las células posteriores, para que la memoria de la red, pueda mantenerse durante todo el proceso de entrenamiento.

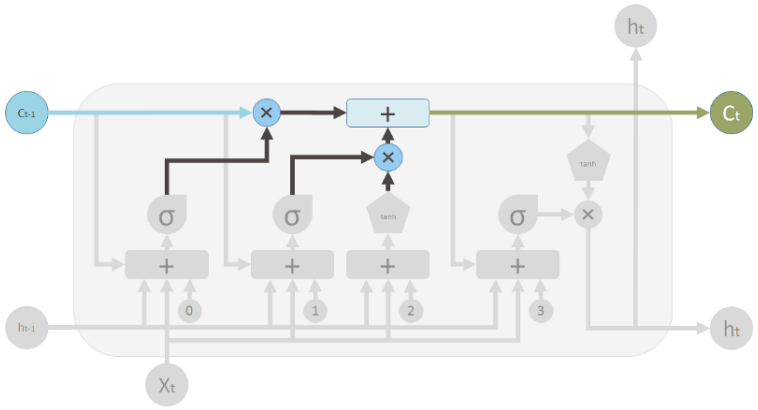


Fig. 3.20: Representación memoria de bloque actual.

- Salida de unidad LSTM

  Finalmente es calculada la salida del bloque actual, esta es influenciada por el estado resultante de la memoria actual de la unidad LSTM C_t, junto a la salida del bloque

anterior h_{t-1}, la entrada actual del bloque X_t, y un bias asociado.

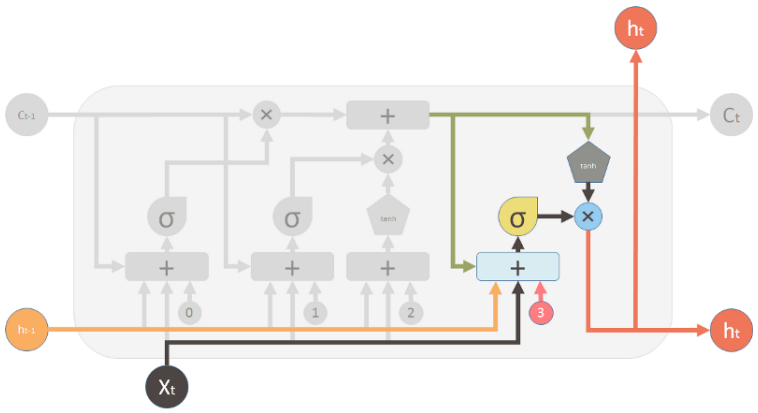


Fig. 3.21: Representación salida de bloque actual.

El diagrama asociada al cálculo del estado actual del bloque, puede observarse en la Figura 3.21. La salida final h_t, puede ser utilizada de al menos 3 formas distintas.

- Entrada siguiente bloque: La salida generada h_t será la entrada del siguiente bloque, representada como h_{t-1}.
- Entrada bloque de capa superior: Algunas arquitecturas de redes recurrentes tienen más de una capa de bloques LSTM, teniendo estructuras bidireccionales, por lo que la salida h_t, puede ser utilizada por otro bloque, como una entrada X_t.
- Salida de la red: Algunas arquitecturas de redes recurrentes necesitan que cada salida de un bloque, sea tomado en consideración para realizar predicciones, por lo que la presente salida h_t, puede ser la entrada de otro tipo de red, como una Feedforward, para realizar a través de ella predicciones. O en su defecto, si el bloque actual es el último de la red LSTM, puede ser directamente la salida de la red, representando su salida, la categorización o predicción final que realiza sobre los datos analizados.

3.5. Pre-Procesamiento

Para obtener mejores resultados a la hora de utilizar algoritmos asociados a máquinas de aprendizaje, es necesario trabajar previamente los datos que serán utilizados. Esto permitirá ayudar a los sistemas a que puedan trabajar de mejor manera y con ello obtener mejores resultados.

Dada la necesidad de realizar este pre-procesamiento de datos, es que a continuación se detallarán algunos asociados al manejo de texto en este tipo de algoritmos.

3.5.1. Vectorización de Palabras

Actualmente los computadores son incapaces de poder comprender el lenguaje humano de la misma forma que nosotros lo hacemos. Para ellos, la palabras no son nada más que un conjunto de caracteres que carecen de significado y que no logran tener una relación con otra palabra. Por lo que, a pesar de que un computador sea capaz de detectar palabras dentro de un texto u oración, le será imposible llegar a relacionar cada una de ellas a nivel semántico y morfológico como un humano, limitando profundamente las capacidades analíticas que un computador puede lograr.

Es por esto, que se han realizado estudios desde el área del procesamiento natural del lenguaje, para poder representar de mejor manera un texto, y que este pueda ser "comprendido" por una máquina. Uno de ellos es por medio de la vectorización de palabras (T. Mikolov, K. Chen, G. Corrado and J. Dean [36]), el cual desde un espacio multidimensional, logra relacionar un vector en particular con una palabra, permitiendo que las distancias entre ellos, representen las dependencias entre palabras. Cada dimensión es capaz de representar características propias de cada una de ellas.

Esta transformación, permite no tan sólo poder representar sus dependencias, si no además, poder manipularlas matemáticamente y obtener con ello un tercer vector, el cual puede

ser atribuido a alguna palabra cercana a él. Un clásico ejemplo puede ser observado en la figura 3.22, en la cual el vector, puede ser obtenido por medio de sumas y restas de otros. Por ejemplo:

$$hombre - mujer + reina = rey \tag{3.11}$$

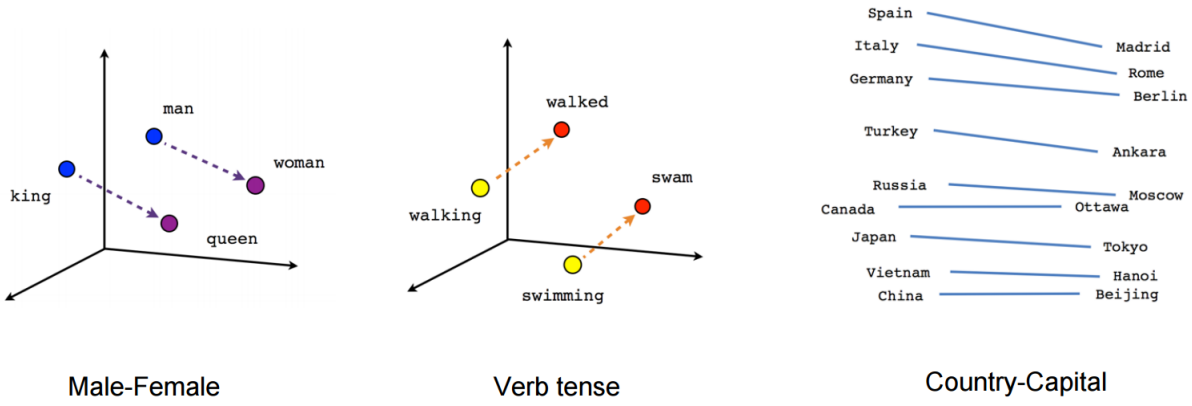


Fig. 3.22: Representación Vectorial.

Esto es posible, dado que la vectorización de palabras busca poder representar de la mejor manera tanto la semántica como su significado, es decir, palabras que poseen una misma relación, tendrán por consiguiente una misma distancia. Es por eso que la distancia existente entre el $hombre - mujer$ es la misma que entre $rey - reina$, distancia que caracterizara la distinción de $masculino - femenino$. Lo mismo ocurre para relaciones entre $pas - ciudad$, donde la diferencia entre las palabras Santiago-Chile, es igual a Lima-Perú. Los ejemplos mencionados anteriormente, representan un contexto ideal, donde cada palabra, se encontrará representada de manera fiel a un vector, esto en realidad no es posible de replicar, dado que además de ser una tarea que tomaría mucho tiempo, es demasiado difícil de realizar, tomando en cuenta la gran cantidad de palabras distintas que presenta un idioma.

Dada la dificultad, es que se han desarrollado diversos métodos por medio de la lingüistica computacional, para realizar la misma labor, pero de manera automática. Los primeros avan-

ces, toman la idea de que, palabras similares tienden a encontrarse bajo un mismo contexto (Miller and Charles, 1991) [28], permitiendo realizar algoritmos que asignaban distintas clases a las palabras, según la frecuencia de sus ocurrencias dentro de un contexto dado (Brown et al., 1992) [29].

Con el progresivo avance de técnicas provenientes de Machine Learning en los últimos años, ha sido posible entrenar cada vez más modelos sobre una larga cantidad de datos no estructurados, es decir que no siguen un orden lógico (por ejemplo frases de distinto largo, distintas palabras, etc), logrando superar de gran manera modelos de hasta entonces, más simples. El concepto que más ha ayudado en este ámbito, es la de usar representaciones de palabras distribuidas (G.E. Hinton, J.L. McClelland, D.E. Rumelhart, [37]), permitiendo que muchos modelos de redes neuronales mejoren significativamente en comparación a modelos basados sólo en N-Gram (H. Schwenk [38]), el cual se basa en un modelo probabilista que permite hacer una predicción estadística de próximo elemento de una secuencia de elementos. Con estos previos avances, se llega a dos de las arquitecturas más utilizadas, base de uno de los softwares más conocidos en esta área, como lo es Word2Vec (T. Mikolov, K. Chen, G. Corrado and J. Dean [36]).

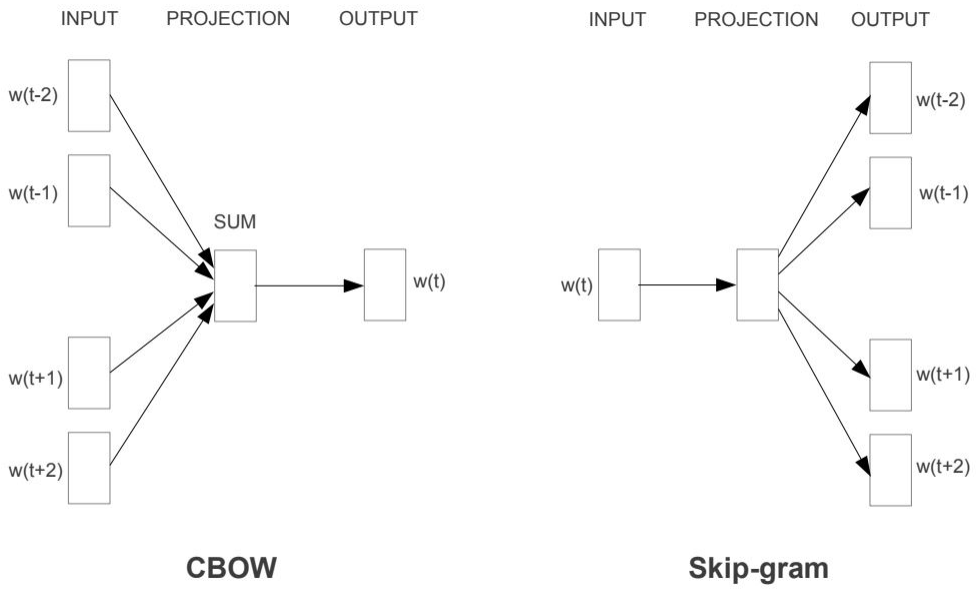


Fig. 3.23: Arquitecturas de modelo Word2Vec [36].

Las dos arquitecturas corresponden a CBOW (Continuous Bag-of-Words, bolsa de palabras continuas) y a Skip-gram (no posee traducción), estas pueden ser observadas en la figura 3.23. Ambos algoritmos son un tanto similar, excepto que CBOW (Modelo de la izquierda) predice una palabra a partir de un conjunto de ellas, es decir, tratará de encontrar la palabra correcta que logre relacionar todas ellas bajo un mismo término. Skip-Gram (Modelo de la derecha) realiza lo mismo pero de forma inversa, predice el conjunto de palabras a partir de un término. Este último puede tender a presentar mejores resultados, ya que trata cada nuevo término con un nuevo contexto, permitiendo que en datasets más grandes resulte mejor que con CBOW.

Existen muchas aplicaciones reales utilizando la vectorización de palabras [27], y en especial bajo el tema del presente estudio, en el análisis de sentimiento, donde son un pilar fundamental cuando se utilizan redes neuronales recurrentes, ya que permiten capturar de la mejor manera, características propias de cada palabra con que es alimentada la red.

Texto n°1	||	2
Texto n°2	||	1
Texto n°3	||	0
Texto n°4	||	2
Texto ..	||	..

Tab. 3.1: Ejemplo de formato de un dataset.

3.5.2. Dataset

Dataset se denomina al conjunto de datos que pueden o no estar previamente catalogados, a partir del cual será realizado todo el análisis y estudio. Este conjunto es una de las partes principales del trabajo dada la importancia que tiene en el resultado final.

Los algoritmos de Deep Learning utilizados para la clasificación de comentarios, basan sus entrenamientos y pruebas sobre un dataset, por lo que la correcta formulación de este permitirá obtener buenos resultados.

Los datos de un dataset en el área de “Análisis de Sentimiento”, se tratan de texto necesario para el entrenamiento de redes, y dado que se utilizan algoritmos que son supervisados, es decir que se necesita proveerles la clase a cual pertenece, es que se agrega, además del texto, el número correspondiente a su clase. Por ejemplo, tomando en consideración el “Análisis de Sentimiento”, un comentario positivo puede arbitrariamente estar asociado a un 2, uno neutro a un 1 y uno negativo a 0, lo importante es que exista una diferenciación con respecto a cada una de las clases. El dataset es quien permite que la red aprenda por lo que una mala identificación de estos perjudicará enormemente el desempeño de ella.

En 3.5.2 es posible observar un tipo de formato típico para un dataset, estos pueden cambiar dependiendo de las necesidades de cada algoritmo. En este caso, en cada linea se ubica sólo un texto (frase, párrafo o comentario) separado por algún carácter especial, en el ejemplo se utiliza un "||", dado que su uso en un texto español es casi nulo, por lo que es posible separar la clase (o sea el número) del comentario mismo.

Para los algoritmos supervisados, es necesario dividir el conjunto de datos en un mínimo de 2 partes, denominadas Dataset de Test (Prueba) y de Training (Entrenamiento), puede existir un tercero, denominado Validation (Validación), pero este último dependerá de específicamente de quién este elaborándolos, ya que , sólo permite validar si el algoritmo está bien implementado o no.

El conjunto de entrenamiento permite, tal como lo dice, entrenar la red neuronal. Por otro lado, el de Prueba, permite corroborar el funcionamiento de la red sobre datos que nunca ha "visto", es decir, para medir que tan bien se comporta el algoritmo extrapolando su funcionamiento sobre nuevos comentarios, lo que viene a ser su capacidad de generalización. La proporción existente entre estos dos conjuntos, dependerá básicamente de la cantidad total de datos correctamente catalogados que se tenga. A una mayor cantidad, es posible dividir el conjunto en una muestra de 70:30, es decir 70 % del total en el dataset de entrenamiento, y el resto en el de prueba. Si la cantidad total es pequeña, se pueden usar valores cercanos a un 90:10, o derechamente usar otros algoritmos como el llamado, k-fold, el cual permite entrenar de distintas maneras a partir de un mismo dataset, llegando a presentar incluso mejores resultados con una gran cantidad de datos [30]. Esta última metodología no será discutida, dada que no se utilizará en el desarrollo de este estudio.

Dada la importancia de un buen dataset, estos en lo posible, deben poseer ciertas características que permitan dar un grado de confianza alto, estos son:

- Los datos deben estar perfectamente etiquetados. Al no cumplirse esto, genera que la red aprenda de una manera errónea.
- Los datos deben ser únicos, es decir, no se deben repetir a lo largo del dataset. Esto, para que la red no se especialice sobre frases particulares.
- En lo posible, los datos deben estar intercalados a lo largo del dataset. La red, al recibir constantemente datos de una clase, afecta en su aprendizaje final.

- El conjunto de datos debe ser lo más diverso posible y con la mayor cantidad posible de ellos. Mientras más diversos sean, permite que la red pueda extrapolar sus resultados de mejor manera.

- Mientras mayor sea la cantidad de datos, mejor resultados se obtendrán, ya que la red estará menos proclive a caer en un sobre-entrenamiento (Overfitting), es decir, que a una menor cantidad de ellos, las redes comienzan muy rápido a memorizarlos limitando enormemente su acción sobre comentarios nuevos. Si bien, dependiendo del tipo de red, existen ciertas acciones a realizar para que esto no ocurra, lo recomendable es siempre intentar contar con la mayor cantidad de datos posible.

3.5.3. Corpus

Un corpus es un conjunto cerrado de texto o de datos, que están destinados a la investigación científica. Es similiar a un dataset en el sentido de que contiene texto a utilizar en el entrenamiento, pero en este estudio, difiere en que será utilizado en el entrenamiento de la vectorización de palabras.

Como bien se explicó, la vectorización de palabras es parte esencial del trabajo final, dado que permite poder relacionar palabras tanto morfológica como semánticamente por medio de vectores. Permitiendo por ejemplo, saber que las palabras "también", "tmb", "tbn", tienen un significado similar.

Estos algoritmos de PNL realizan sus métodos de aprendizaje por medio de un contexto, es decir, que vectorizan una palabra según el contexto en la cual se encuentran. Por lo que, si una misma palabra la utilizamos en una cantidad mayor de oraciones o frases que mantengan un contexto similar, pero con distintas palabras, permitirá que el desempeño final del algoritmo sea mejor. Un mejor desempeño implica entre otras cosas, que el vector final asociado a la palabra, será representado de una mejor manera en relación a las otras. Con esto, mientras mayor sea el corpus a utilizar, se esperarán por lo tanto mejores resultados.

Dependiendo de las necesidades, un corpus puede estar ligado directamente a una área específica, esto implica que las frases o textos utilizados en este, estén relacionados con esta área, permitiendo así poder representar las palabras claves. Por ejemplo, en este trabajo el área está ligada hacia los servicios de telecomunicaciones en Chile, por lo que es clave poder vectorizar de buena forma, palabras como Wi-fi, Smartphone, mbps, señal, etc. Palabras que pueden incidir de manera directa en la clasificación del comentario.

3.5.4. Limpieza de Texto

Para las máquinas de vectorización, una palabra, es toda aquella que se encuentre separada por dos espacios, es decir que por ejemplo "(palabra)", “esta.también”, “no?” serán consideradas como si fueran una palabra. Esto limita bastante su resultado final, dado que la cantidad total de palabras crece, pero no así la cantidad de veces en que ellas se repiten. “esta” y “también” pueden aparecer muy seguidas dentro de un texto, pero no así “esta.también”.

Para evitar estos problemas, es que todo texto, frase u oración a analizar debe ser pasada por un proceso de limpieza (Llamado en la literatura "Tokenizer"), esto implica, modificar el texto en un formato que sea óptimo para su trabajo.

Las distintas modificaciones que se pueden realizar a un texto, dependerán exclusivamente del problema a enfrentar. Algunas de las modificaciones que se pueden realizar, y que pueden ayudar a un mejor desempeño son listadas a continuación.

1. Separar las palabras de algún signo como exclamación o de puntuación.
2. Cambiar caracteres HTML por su correspondiente caracter. & ? &.
3. Quitar imágenes y URLS. Esto dado que no aportan información relevante a su análisis.

4. Cambiar todas las letras en mayúscula a minúsculas. Para los algoritmos de vectorización, las palabras “Hola” y “hola” son distintas dado que la letra H está en mayúscula en la primera. Para evitar que sean tomadas de manera distintas, todas las letras son llevadas a una u otra forma.

5. Eliminar tíldes y símbolos sobre letras. Al igual que el punto anterior “también” y “tambien”, o “sinvergüenza” y “sinverguenza” son palabras distintas, para unificarlas, es necesario cambiar las letras con tildes, por sin tildes. Esto se debe principalmente a que no todas las personas escriben correctamente y menos en Twitter. Si bien esto debería poder evitarse al utilizar la vectorización de palabras, es mejor poder ayudar de cierta forma a que existan más palabras de un tipo, que muchas distintas.

6. Cambio de palabras. Frases como “yo tambiéeeeeeeeen!!” es posible llevarlas a “yo también !”, independiente del número de letras que se repitan, ayudando así a unificar palabras, que de lo contrario serían imposibles de analizar dada que serían prácticamente únicas.

3.6. Ambiente de Entrenamiento

Los entrenamientos de redes asociadas a Deep Learning, no son fáciles de realizar y su complejidad no tan sólo radica en cómo los modelos son programados, si no además, en el uso de recursos con que se debe contar, para lograr buenos resultados. Las redes neuronales necesitan de una gran capacidad de cómputo, que de no contar con ellos, llevará que el tiempo de entrenamiento crezca exponencialmente.

En cuanto a la programación, existen hoy en día una gran variedad de librerías que permiten entrenar distintos tipos de redes haciendo uso de funciones ya implementadas, por lo que un usuario no tiene la necesidad de implementar toda una red desde cero, si no más bien hacer uso de ellas, y enfocarse en el modelo más que en su programación. Actualmente son 6 las

librerías más utilizadas, Tensorflow [3], Theano [4], Torch [5], Keras [6], Caffe [7] y Deeplearning4j [8]. Cada una de ellas posee distintas ventajas con respecto a sus pares, ya sea por el lenguaje con que deben ser utilizados, la facilidad con que pueden ser implementadas, la rapidez con que trabajan y el soporte que entregan. Las que se destacan más en estos puntos son las 4 primeras nombradas, siendo Keras la más fácil de utilizar, ya que es básicamente un framework que permite utilizar de fondo las librerías de Tensorflow o Theano, dependiendo de lo que desee el usuario, pero haciendo que su implementación sea mucho más sencilla que en ellas dos.

Como se mencionó, el uso de recursos es alto, donde si bien los cálculos no son tan complejos de realizar, el alto número de ellos, hace que el tiempo de entrenamiento crezca enormemente, ya que si el dataset es grande, la red es profunda y con un alto número de neuronas en un alto número de iteraciones, hará que se tengan que realizar millones de cálculos, lo que hace que manejar esta cantidad de cómputos por un procesador tome mucho tiempo. Una enorme mejora en esta área, la cual ayudó también a impulsar el uso de Deep Learning, es la utilización de tarjetas gráficas en el proceso entrenamiento de la red, esto dado la gran cantidad de procesadores que poseen. Esto permite la paralelización de los cálculos, ayudando a reducir significativamente el tiempo de entrenamiento de una red, bajando los tiempos de horas a minutos, incluso de meses a días en casos de entrenamiento de una gran cantidad de imágenes.

[3]Librería matemática de código abierto desarrollada por Google, diseñada para implementar aplicaciones de Aprendizaje Automático

[4]Librería matemática de código abierto desarrollada por la Universidad de Montreal.

[5]Biblioteca de código abierto para Aprendizaje Automático basado en el lenguaje de programación Lua

[6]Librería de código abierto escrita en Python, diseñada para implementar aplicaciones de Redes Neuronales, puede utilizar tanto Tensorflow como Torch, como su sistmea de procesamiento.

[7]Framework (entorno de trabajo) de Deep Learning desarrollado por la universidad de Berkeley, también de código abierto.

[8]Librería de código abierto para implementar sistemas distribuidos de Deep Learning, para ser utilizado con código Java y Scala.

4. CAPÍTULO: METODOLOGÍA

Con las técnicas de categorización de texto presentadas anteriormente, se busca mejorar una de las funciones de la plataforma de la empresa Wholemeaning, la cual consiste en categorizar comentarios provenientes de Twitter, según la emoción expresada, en positivo, neutro o negativo.

Para ello se utilizarán dos vías centrales distintas para resolver el problema planteado. Una se centra en la utilización de una red convolucional junto a la metodología de procesar un texto como si se tratase de una imagen. En la segunda la red utilizada es la red recurrente LSTM, cuya entrada serán los vectores asociados a las palabras del comentario a analizar. En ambos casos, las salidas de la red, serán redes Feedforward cuya capa final será una softmax.

Dado que en un principio no se cuenta con un dataset propio, que permita hacer pruebas de los algoritmos, es que se han utilizado dos datasets distintos que si bien no son en español, estos si están ligados al Análisis de Sentimiento, de ellos se hablará más adelante. El uso de los dos dataset externos permitirá poder validar los modelos de redes neuronales, o sea que logren cumplir la tarea propuesta, luego se realizarán pruebas con datasets extraídos de resultados previos de la empresa, y también de uno hecho específicamente para este trabajo. El análisis en detalle del procedimiento llevado a cabo, se presenta al final del presente capítulo 4.8.

Antes de revisar específicamente la metodología de cada sección, se detallará la precisión alcanzada por la empresa Wholemeaning.

Totales	
Facebook	
Negativos	108.683
Positivos	7.883
Otros	2.575
Sin Etiqueta	515.263
Total	634.404
Twitter	
Negativos	427.291
Positivos	18.192
Otros	11.391
Sin Etiqueta	2.0250.452
Total	2.477.326
Total FB+TW	3.111.730

Tab. 4.1: Resumen de dataset categorizado según algoritmos de la empresa, en redes sociales de Facebook y Twitter.

4.1. Precisión Plataforma empresa Wholemeaning

Los algoritmos encargados de categorizar comentarios provenientes de redes sociales son modelos lingüísticos que han sido desarrollados y mejorados durante los últimos años. Durante la presente sección se analizará cual es su grado de precisión, categorizando comentarios sacados de Twitter, asociados sólo al área de Telecomunicaciones.

El resumen del dataset proporcionado por Wholemeaning para el presente estudio puede observarse en la tabla 4.1, donde se encuentran un total de más de 3 millones de comentarios provenientes de dos redes sociales, tales como Twitter y Facebook, y es que a pesar de que el estudio sólo estará enfocado en una de ellas, al tener una mayor cantidad de texto asociado al área de Telecomunicaciones, puede ayudar significativamente en el entrenamiento de los vectores. En el dataset de Twitter, existen comentarios categorizados de 4 maneras distintas, Negativos y Positivos, Otros (los cuales están asociados a emociones específicas, como sorpresa),y por último sin etiqueta, es decir, comentarios que no tenían características asociadas

	Clase Verdadera		
Clases Analizada	Positivo	Neutros	Negativos
Comentarios Positivos	82 %	11 %	7 %
Comentarios Neutros	3 %	53 %	44 %
Comentarios Negativos	1 %	6 %	93 %

Tab. 4.2: Análisis plataforma BI Wholemeaning comentarios por clase

a las anteriores clases, por lo que comentarios sin ninguna etiqueta serán considerados como clase neutra. La clase neutra, por definición, se trata de aquella que agrupa comentarios que carecen de connotación tanto negativa como positiva (de acuerdo a la sección 3.2), por lo que agrupar todos aquellos comentarios sin etiqueta, no sólo permitirá verificar la suposición de que esto se cumpla, si no además, de observar que tan bien es capaz de capturar los comentarios asociados a las otras dos clases.

Tomando sólo en consideración comentarios de Twitter, se seleccionaron 100 comentarios aleatoriamente de cada tipo de clase, de cada uno de ellos se comparó si realmente pertenecían a dicha categorización o no. El resultado evocará su desempeño con respecto a su precisión en la tarea. La comparación se realizó bajo los criterios sobre que es considerado como un comentario asociado a cierta clase, utilizando los parámetros expuestos en la sección 3.2.

- Comentarios Neutros:

  De las tres clases, los comentarios neutros es el que arroja un mayor porcentaje de error, ya que de los 100 comentarios analizados (Tabla 4.1), un 47 % corresponde a mal etiquetados, específicamente un 44 % pertenece a comentarios negativos un 3 % a positivos. Lo que demuestra que muchos de los comentarios no catalogados, si pertenecen a una clase específica.

  Esto tiene un motivo, y es que al realizar un análisis más profundo sobre el funcionamiento de la plataforma de la empresa, se logra observar que no todos los comentarios asociados a la insatisfacción del cliente, son considerados como una emoción negativa, por lo que puede ocurrir que si bien, estos no son etiquetados como negativos, si hayan

sido etiquetados como insatisfacción.

- Comentarios Positivos:

  Los comentarios positivos tienen un 82 % de acierto (Tabla 4.1), donde el resto se distribuye entre un 7 % de negativos y un 11 % de neutros.

- Comentarios Negativos:

  Comentarios bajo la clase de negativos, son los que tienen un mayor grado de precisión, esto se debe a que mayoritariamente son analizados un porcentaje mayor de esta clase, con respecto a las otras dos. Las personas realizan una mayor cantidad de reclamos, que comentarios asociados a dar agradecimientos o felicitar, dado esto es que su sistema lingüístico ha podido ser más preciso. El porcentaje de precisión es de un 93 %, siendo el resto de un 6 % de neutros y un 1 % de positivos.

  Como se observa en la tabla 4.1, la precisión del sistema es bastante alta si predice que un comentario es negativo, pero al observar los resultados de comentarios neutros, se aprecia que muchos de ellos no son "capturados" de buena manera, haciendo que la cantidad real de comentarios negativos sea mucho más.

4.2. Armado Datasets

Tal como se mencionó anteriormente se utilizaron previamente datasets externos para realizar las primeras pruebas sobre los algoritmos desarrollados con las redes neuronales, de estos se hablará a continuación 4.2.1.

Para el análisis final, donde será necesario utilizar comentarios propios del área de trabajo, se presentan 3 datasets distintos, 2 de los cuales serán realizados en consideración con la tabla 4.2. Dicha tabla muestra la cantidad de comentarios que serán finalmente considerados en cada clase, para considerar el tamaño final, se toma como referencia la menor cantidad de

Facebook			
% Correcto	Clase	Cantidad	Cantidad a Utilizar
91 %	Negativo	108.683	7.800
82 %	Positivo	7.883	7.800
54 %	Neutro	515.263	7.800
Twitter			
% Correcto	Clase	Cantidad	Cantidad a Utilizar
81 %	Negativo	427.291	18.000
82 %	Positivo	18.192	18.000
74 %	Neutro	2.020.452	18.000

Tab. 4.3: Tabla resumen, con las cantidades a utilizar en los dataset.

una clase, esto es por que es necesario que la cantidad de ejemplos asociada a cada una de ellas, se mantenga equilibrado para no permitir que la red se especialice en la que posea más ejemplares.

A continuación se detallan cada uno de los datasets.

- Telco Facebook y Twitter: Dataset que toma en consideración comentarios extraídos de ambas redes sociales, dado que los comentarios de Facebook superan la cantidad de 140 caracteres, este dataset no puede ser utilizado para la red convolucional dada su limitante en tamaño. La cantidad total de ejemplos es de 77.400 divididos en 25.800 por cada clase, donde 7.800 provienen de Facebook y los restantes de Twitter.

  Si bien el presente estudio no busca clasificar comentarios provenientes de Facebook, estos fueron tomados en cuenta para poder tener una mayor cantidad de comentarios de entrenamiento al utilizar redes recurrentes, las cuales no se ven tan afectadas por el largo de un texto como las redes convolucionales.

- Telco Twitter: Dataset que sólo tomará en consideración los comentarios relacionados con Twitter, o sea existirán 18.000 ejemplos asociados a cada clase.

- Nuevo Telco Twitter: Dataset creado a partir de los comentarios extraídos desde la

plataforma de Wholemeaning, lo que distingue este de los anteriores, es que fue construido tomando en consideración que todos los ejemplos estarán realmente asociados a su clase correspondiente, lo que genera en teoría, que la precisión del dataset será de un 100 %. Es decir, a partir de los dataset propios de la empresa, se crea uno nuevo, eliminando todo comentario que no corresponde a su clase. Para su generación, se utiliza un programa diseñado para leer comentarios y categorizarlos inmediatamente por medio del criterio de un usuario, por lo que el dataset resultante puede contener algunos errores asociados a la percepción que posea la persona con respecto a la emoción expresada en él.

Existen comentarios cuya connotación tanto negativa como positiva es bastante ambigüa o que generan un amplio margen a la intepretatividad que pueda tener el usuario para su categorización, es por esto, que para minimizar dicho problema, es que estos comentarios no son tomados en consideración para el dataset de entrenamiento. Esto se debe a que la red debe tener en lo posible, los mejores ejemplos asociados a cada tipo de clase, para que genere a partir de ellos buenas predicciones. Por lo mismo que además de categorizar comentarios ya elaborados, es que se han elaborado frases típicas presentes en el área de trabajo (telecomunicaciones).

En total, se desarrolló el dataset, con un total de 1800 comentarios por cada clase.

4.2.1. Uso de datasets externos para prueba

Una forma de determinar que los algoritmos desarrollados realmente funcionen, y no tengan problemas asociados a la calidad de los datasets, es que como medida previa, se utilizarán datasets externos de entrenamiento y de prueba, datasets que de cierta manera entregan fiabilidad de los datos con que se va a trabajar.

Para la red neuronal convolucional, se trabajará con un dataset 100 % proveniente de Twitter, denominado como Sentiment140 (Sentiment 140 [56]), el cual contiene 1.6 millo-

nes de comentarios catalogados como positivos y negativos, cada clase posee el 50 % de los comentarios. Este dataset fue construido utilizando un Clasificador de Máxima Entropía, algoritmo de Machine Learning, cuyo desempeño no es de un 100 % por lo que existirá un error asociado a las categorías asignadas. Aún así es un buen dataset para comenzar a probar la arquitectura de una red Convolucional.

El segundo dataset a utilizar se llama "Large Movie Review Dataset" (gran conjunto de datos de análisis de películas), el cual es el estándar utilizado en investigación para evaluar el desempeño de distintos algoritmos asociados al "Sentiment Analysis". Este dataset contiene 50 mil reviews (Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher. [57]) separados en dos clases, positivos y negativos (Al igual que el anterior), donde existen 25 mil ejemplos para cada una. Cada uno de los reviews fue tomado de alrededor de 30 películas distintas de la página IMDB en el cual, comentarios con arriba o igual a 7 estrellas eran considerados como positivos e iguales o menores a 4 como reviews negativos. Este es uno altamente utilizado, el cual ha sido mejorado al pasar los años, por lo que su confiabilidad en cuanto a que realmente sean de la clase que dice ser, es alta.

4.3. Pre-Procesamiento de Datasets

Para establecer una unificación de ciertas palabras, ayudando a los algoritmos a no trabajar con un gran espectro de distintas palabras, es que se debe realizar un programa que se encargue de limpiar el texto. Esta limpieza debe transformar todos los datasets y corpus a utilizar, las transformaciones deben ser las señaladas en la subsección de "Limpieza de Texto" 3.5.4, a excepción del último punto, el cual por el momento no se considera muy trascendente (Acortar palabras donde se repita más de una vez una letra, ejemplo "Que Genialll!!!' a "que genial !").

Se desarrolla un script [1] en Python encargado de la "tokenización" de los cuerpos de datos, el cual de acuerdo a diversas reglas de expresiones, modificará el texto base y lo llevará a uno acorde a las necesidades, permitiendo que el espectro total de palabras se vea disminuido, y con ello obtener mejores resultados. La confección del script sólo tiene como fin, poder mejorar el desempeño de la red recurrente.

4.4. Vectorización de Palabras

Actualmente existen 3 programas capaces de vectorizar palabras a partir de un corpus de entrenamiento, los 3 son de código abierto, por lo que su uso no se encuentra restringido a una licencia, estos 3 son Word2Vec , FastText y GloVe (J. Pennington, R. Socher and C. D. Manning [45].). Se decidió utilizar los dos primeros, puesto que Word2Vec es uno de los algoritmos más utilizados y con mayor información (GloVe en cuanto a su algoritmo es bastante similar, pero con menor respaldo), FastText por ser el más nuevo, presenta una serie de ventajas sobre los otros dos.

FastText (A. Joulin, E. Grave, P. Bojanowski and T. Mikolov [44].), librería desarrollada por el grupo de investigación de Facebook, es prácticamente una extensión de Word2Vec (T. Mikolov, K. Chen, Greg Corrado and J. Dean [36].), el cual una de sus grandes ventajas:

1. Disminuye considerablemente el tiempo de entrenamiento.
2. Permite ser entrenado en más de un billón de palabras en menos de 10 minutos.
3. además de poder generar mejores vectores asociados a palabras raras, palabras que no se encuentran comúnmente y
4. por último la de poder generar vectores a palabras que se encuentran fuera del vocabulario de entrenamiento

[1] programa que reune una serie de conjuntos de órdenes que es posteriormente ejecutado por lotes o línea a línea, en tiempo real por un intérprete.

Los dos últimos puntos, se debe a que FastText no trata las palabras como un todo, si no como un conjunto de grupos de letras, llamados "n-grams" (o n-gramos), donde n denota la cantidad de letras. Por ejemplo, utilizando un minimo de 2 y un máximo de 6 n-gram, la palabra "sandía" puede ser representada como la suma de vectores de sus n-gram; $[sa, san, sand, sand\acute{i}, sand\acute{i}a, an, and, and\acute{i}, anda, nd, nd\acute{i}, nd\acute{i}a, d\acute{i}]$.

La comparación entre Word2Vec y FastText se realizará usando los dos tipos de algoritmos de vectorización de palabras, Skip-gram y CBOW. Para FastText se utiliza directamente el código previsto por su página (FastText [47]), el cual necesita ciertos requerimientos básicos para su funcionamiento; como un compilador gcc-4.6.3 o más nuevo, Python 2.6 o mayor y extensiones de él como Numpy [2] y Scipy [3]. El uso de Word2Vec es a través de Gensim, un framework de Python que contiene distintas herramientas, entre ellas, Word2Vec, lo que facilita su uso, pero no altera los resultados.

Para todos los casos, ya sea utilizando Skip-gram o CBOW, se crearán vectores de dimensionalidad 300.

4.5. Configuración Ambiente de Trabajo

El ambiente de trabajo es provisto por la empresa, el cual consiste en el arriendo de un servidor de Microsoft Azure llamado "Azure Data Science Virtual Machine" (DSVM [48].). Esta es una máquina virtual que se encuentra en la nube de Microsoft Azure [4], la cual provee todas las herramientas para el estudio de "Data Science" [5]. Muchas de estas herramientas ya se encuentran instaladas, listas para ser usadas.

[2] Biblioteca de funciones matemáticas de alto nivel para Python.

[3] Biblioteca de herramientas y algoritmos matemáticos para Python.

[4] Servicio en la nube que ofrece Infraestructura y Plataforma como servicio, en pocas palabras, cientos de Datacenters con gran cantidad de recursos que pueden ser accedidos desde cualquier parte del mundo.

[5] Métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento desde los datos.

La máquina virtual a utilizar cuenta con las siguientes características y herramientas que ayudan al presente estudio.

- Ubuntu 16.04 LTS: Ubuntu es una distribución del sistema operativo GNU/Linux.
- 56 GB RAM.
- Python 2.7 y 3.5: Lenguaje de programación interpretado.
- Keras ([49]): Es una librería de código abierto escrita en Python, la cual permite trabajar con modelos de Deep Learning sobre distintas librerías matemáticas, como TensorFlow, Theano o DeepLearning4j. Al ser estas librerías de dificil uso, Keras permite trabajarlas de una manera más simple y directa, al simplificar distintas metodologías por medio de distintas funciones, logrando que el código final sea más directo.
- Tensorflow ([50]) : Biblioteca de código abierto desarrollado por Google, diseñado para la computación numérica, específicamente orientado a problemas de Deep Learning. Para realizar los cálculos de manera más rápida, es posible utilizar Tensorflow con una GPU Nvidia.
- CUDA, CUDNN, Nvidia Driver: Dado que el uso de una GPU disminuye considerablemente los tiempos de entrenamiento de una red neuronal, es que la máquina virtual esta provista de drivers instalados para la comunicación de las tarjetas gráficas con otros programas.
- Nvidia Tesla k80: Esta tarjeta está específicamente diseñada para acelerar las aplicaciones de análisis de datos y cálculo científico. Esta optimizada para sistemas de aprendizaje profundo, por lo que su uso es idóneo para el presente trabajo.
  Dentro se sus características se encuentran; Dos GPU; 24 GB de memoria GDDR5; 480 GB/s de ancho de banda de memoria; 4992 núcleos de procesamiento paralelo CUDA, entre otras.

Una de las metas, fuera de los objetivos de este estudio, es la de implementar las redes neuronales dentro de los sistemas de la empresa, dado esto, es que el entrenamiento se ha

realizado utilizando dos librerías. La primera es Keras, con la cual se realizará un primer acercamiento a Deep Learning, en ella serán modelados la mayoría de las redes utilizadas en el presente estudio. La segunda librería, es la de Tensorflow, la cual está diseñada para llevar modelos de Machine Learning a los servicios de las empresas, por lo que el mejor modelo de la presente investigación, será implementado directamente en Tensorflow, modelo con el cual se realizará el análisis final. Si bien Keras permite trabajar con Tensorflow como herramienta matemática, el código final puede ser optimizado más fácilmente cuando se encuentra directamente trabajado con Tensorflow y no a través de Keras.

4.6. Redes Convolucionales: Modelo Propuesto

Para poder implementar una red convolucional para la clasificación de texto, es necesario aplicar ciertas metodologías que permitan tratarlo como si fuera una imagen. Como guía de este proceso, y del modelo de red, es que se ha tomado como referencia un estudio realizado por Xiang Zhang y Yann LeCunn (X. Zhang and Y. LeCun [43]). Este trabajo llamado "Text understanding from Scratch" ("Comprensión de texto desde cero", en su traducción directa), demuestra que es posible clasificar un texto sin tener conocimientos previos de palabras, frases, sentencias o reglas de sintáxis o de semántica, e incluso siendo totalmente indiferente del idioma.

Xiang y Yann proponen una red convolucional cuyas entradas provienen directamente de las letras del texto, haciendo que su forma de entrenar, sea totalmente independiente del idioma. Para lograr esto, y que la red pueda aprender, es necesario tratar las letras de una manera distinta, realizando una transformación, que permita no sólo realizar operaciones matemáticas sobre ellas, si no además adaptar el texto de una manera similar a una imágen, objetos sobre los cuales una red convolucional ha sido adaptada.

Para su transformación se propone una lista de caracteres a utilizar, los cuales permitirán generar vectores de cada letra, asociados a la posición de ella dentro de la lista. En el presente

estudio se proponen dos listas, la primera (Lista 4.1), toma en consideración todas las letras, mayúsculas y minúsculas, con o sin tilde, con números y algunos caracteres especiales. La segunda (Lista 4.2) sólo considera letras minúsculas, sin tilde, además de los números y caracteres especiales. Ambas listas tienen considerada dentro de ellas el espacio vacío.

$$\begin{aligned} &\text{“}aAáÁbBcCdDeEéÉfFgGhHiIíÍjJkKlLmMnNñÑ \\ &oOóÓpPqQrRsStTuUúÚüÜvVwWxXyYzZ01234567 \\ &89:()=*,.-_[]@?/<?\text{”} \end{aligned} \tag{4.1}$$

$$\text{“}abcdefghijklmnopqrstuvwxyz0123456789:()=*,.-_[]@?/<?\text{”} \tag{4.2}$$

En el paper guía sólo proponen un tipo de lista, principalmente porque trabaja con el idioma inglés, pero es interesante poder notar, la influencia de considerar todo el espectro de las distintas letras. Es claro que la inclusión de más letras puede ayudar a capturar más precisamente la comprensión de un comentario, pero a su vez puede necesitar que la cantidad de ejemplos aumente de manera considerable para tener que llegar a ese punto. Una lista más corta en cambio, necesitará de una menor cantidad de ejemplos de entrenamiento para poder categorizar con un alto grado de precisión.

Por cada letra se asignará un vector donde todas sus componentes serán 0 a excepción de la componente que se encuentra en la misma posición que la letra dentro de la lista, el cual tendrá el valor de 1. Si un caracter no se encuentra en la lista, está tendrá el mismo vector asociado que el espacio vacío. Con esto se crearán matrices de tamaño fijo para cada comentario, por ejemplo, tomando en consideración la lista número dos (4.2), las matrices presentarán un tamaño 140×56 , siendo 140 la cantidad máxima de caracteres dentro de un comentario en Twitter, y 56 la cantidad total de caracteres en la lista.

El modelo de red utilizado es también guíado por el estudio de Xiang y Yann, en él, se

Red Convolucional			
Capa	N° Filtros	Tamaño	Capa Max-Pooling
1	64	(7x7)	(2x2)
2	128	(7x7)	(2x2)
3	128	(5x5)	(2x2)
4	256	(3x3)	Sin Pooling
5	256	(3x3)	Sin Pooling
6	256	(3x3)	Sin Pooling
Red Feedforward			
Capa		N° Neuronas	
7		1024	
8		1024	
9		2 o 3	

Tab. 4.4: Estructura modelo de red convolucional propuesto.

proponen dos modelos dependiendo de la cantidad de palabras del texto a tratar. El modelo a utilizar tiene 9 capas de profundidad, compuestas por 6 capas convolucionales y 3 capas feedforward (En la tabla 4.6 puede observarse con más detalle.). El número de filtros por capa aumenta a medida que la capa está más profunda, pasando de 64 y 128 a 256, esto dado a que no es posible seguir al 100 % lo expuesto en el paper, ya que al tener tantos filtros en la red, genera un problema en la tarjeta de video. Se aumentó el número de filtros en capas posteriores puesto que existe una mayor probabilidad de que en ellas se encuentre la información más concentrada al ya haber pasado por las capas de Max-Pooling, por lo que pueden encontrar una mayor cantidad de patrones. El tamaño de los filtros también viene condicionado con el tamaño de la información, las primeras capas contienen un tamaño mayor, por lo que se necesitan filtros de mayor tamaño, disminuyendo a medida que la capa es más profunda.

La red Feedforward se compone de 3 capas, las dos primeras están compuestas de 1024 neuronas totalmente entrelazadas, para finalizar con una capa Softmax con una cantidad de 2 o 3 neuronas. En esta última capa, se consideraran 2 neuronas al utilizar la red con el dataset Sentiment 140, dado que sólo posee dos clases, y 3 neuronas cuando se trabaje con las 3 clases, positivo, negativo y neutro.

4.7. Redes Recurrentes LSTM: Modelo Propuesto

A diferencia del modelo de red convolucional, el modelo recurrente LSTM no es tan estricto en su estructura, permitiendo no tan sólo realizar pruebas configurando sus parámetros, si no además su estructura. La estructura base es la "Muchos a uno" (Figura 4.1).

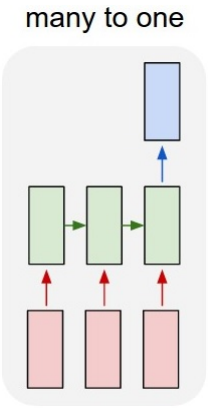


Fig. 4.1: Estructura base para la red Recurrente

Además de utilizar los dos algoritmos de vectorización de palabras, para ambas librerías (Word2Vec y FastText), se añade además una capa capaz de mejorar los vectores durante el proceso de entrenamiento, una capa en Keras llamada Embedding, esta capa puede permitir mejorar los resultados finales.

4.8. Procedimiento

Hasta el momento se han detallado cada uno de los puntos que serán realizados en la obtención de un modelo predictivo, como lo son el armado de datasets, la vectorización y la limpieza de texto. Es por eso que en la presente sección se explicará el procedimiento que se llevará a cabo, que permita obtener las dos herramientas más importantes en el presente estudio, estas son el dataset de entrenamiento, y el modelo de Deep Learning a utilizar.

El trabajo se divide en 3 etapas, representadas por el diagrama de la figura 4.2. En él, se

representa el uso de los datasets como elipses de color gris, los rectángulos azules (de puntas redondeadas) representan los distintos modelos de Deep Learning utilizados y por último los rombos de color naranja simbolizan los dataset de pruebas, realizados específicamente para el presente trabajo, detallados en la sección de armado de datasets 4.2.

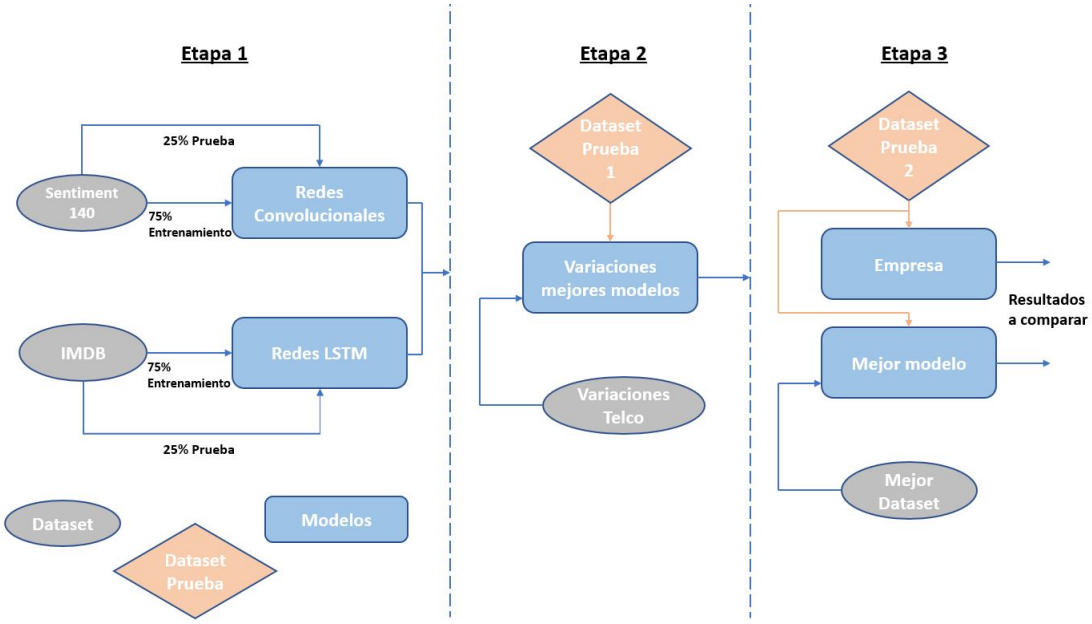


Fig. 4.2: Etapas de desarrollo.

Cada una de las etapas, será detallada a continuación.

4.8.1. Etapa 1

La primera etapa busca encontrar los mejores modelos para cada uno de los tipos de redes a utilizar. Para eliminar el factor proveniente de la calidad de los dataset de entrenamiento, es que para esta etapa, se utilizan datasets externos, donde si bien no presentan un 100 % de confiabilidad de acuerdo a que todos los comentarios estén realmente asociados a su clase verdadera, si servirán para poder considerar los mejores modelos, a partir de la gran cantidad de datos a entrenar.

Las redes convolucionales se analizará con el dataset Sentiment140, mientras que la red recurrente LSTM se entrenará utilizando el dataset asociado a reviews de películas, el dataset de IMDB. Para cada tipo de red, se modificarán ya sean parámetros internos, como estructura de las redes para observar el impacto que tienen ellas en sus resultados de predicción. Para realizar las correspondientes pruebas, es que el 25 % de los datos de cada uno de los datasets, será utilizado para verificar el porcentaje de acierto en sus predicciones.

Parámetros que pueden variar en cada uno de los modelos de red, pueden ser el número de neuronas por capa, cantidad total de capas, factor de aprendizaje η, tamaño de los batchs de entrenamiento, etc. Además de la utilización de distintos métodos de vectorización de palabras o la variación de tamaño de diccionario de letras en el caso de las redes convolucionales.

La obtención del o de los mejores modelos, se consigue simplemente realizando una comparación en los porcentajes de acierto de cada una de las redes, en cuanto a la categorización de los comentarios de acuerdo a su clase. El o los mejores modelos serán considerados en la etapa 2.

4.8.2. Etapa 2

La segunda etapa está dedicada a analizar el impácto que tienen distintos tipos de datasets sobre el desempeño que puede tener un modelo de una red. Para esto es que se utilizan los 3 tipos distintos de datasets presentados en 4.2.

El o los distintos modelos obtenidos en la etapa 1, que sobresalen en cuanto a la acertividad que logran en sus predicciones, son entrenados nuevamente utilizando estos nuevos tipos de dataset, pero a diferencia de la etapa anterior, estos no son probados sobre los mismos, si no sobre unos elaborados con el propósito de realizar comparaciones equitativas a cada modelo. Este nuevo dataset de prueba consta de 100 comentarios por cada clase, y fueron recopilados utilizando la plataforma de la empresa, pero sin utilizar un filtro de sentimiento. El único filtro utilizado para la obtención de los comentarios, es la de fecha, lo que permite

sólo obtener comentarios que fueron publicados a partir de cierto día. Con esto se evita que cada comentario seleccionado, se encuentre en alguno de los datasets utilizados para el entrenamiento.

La comparación finalmente es realizada, al igual que en la etapa 1, en referencia al porcentaje alcanzado en la categorización correcta de comentarios. De estos resultados se obtendrá tanto el mejor dataset para entrenar, como el mejor modelo de red a utilizar. Esta combinación será considerada en la última etapa de estudio.

4.8.3. Etapa 3

Después de ambas pruebas se tendrá una base final, que considerará el mejor modelo junto al mejor dataset de entrenamiento. Esta etapa considera una prueba final, donde se realiza una comparación en cuanto a las predicciones del modelo obtenido versus la plataforma BI de la empresa Wholemeaning.

Para ello se ha elaborado un dataset final de prueba, que contiene 200 comentarios por clase, comentarios que al igual que la etapa 2, no han sido vistos por ninguno de los modelos. La elaboración de este, fue realizada por una persona del área de lingüistas de la empresa, donde cada uno de los comentarios pertenece al área de Telecomunicaciones y fueron realizados durante el mes de Noviembre del 2017. Con esto último se evita, que los comentarios de pruebas no se encuentren dentro del dataset de entrenamiento del modelo de Deep Learning.

Para hacer una comparación más justa, es que se ha modificado la forma de considerar los comentarios etiquetados por la plataforma de Wholemeaning , esto porque su analizador de sentimiento no siempre cataloga la insatisfacción de un cliente como comentario negativo. Por lo que se ha decidido adaptar e incluir los comentarios que son categorizados como “insatisfacción” por la plataforma, como comentarios negativos, lo mismo para catalogados como “satisfacción” serán incluidos como positivos. Esto permite mejorar el desempeño de la herramienta de la empresa, para realizar una mejor comparación.

El análisis final, a diferencia de las anteriores etapas, se realizará no sólo comparando el porcentaje de acierto que tiene por clase, si no también por medio de un análisis de curva ROC (Receiver operating characteristics o Característica Operativa del Receptor). La interpretación que se logra a través de ROC es la representación de la razón de verdaderos positivos (VPR = razón de Verdaderos Positivos) frente a la razón de falsos positivos (FPR = Razón de Falsos Positivos). La curva ROC permite comparar distintos modelosen base a las razones VPR y FPR logrando plasmar a través de ellos, no tan sólo el porcentaje de acierto, si no además el grado de confiabilidad que posee el modelo al momento de predecir cierta clase, o sea que tan probable es de que una clase ya predicha, pertenezca realmente a ella y no sea de otra.

Esta prueba podrá determinar si finalmente el método propuesto a través de la utilización de modelos de Deep Learning, ayudado con procesamiento natural del lenguaje y de la lingüística computacional obtiene mejores resultados que los proporcionados por el mejor modelo asociado al área de Telecomunicaciones, proveído por la empresa Wholemeaning.

5. CAPÍTULO: ANÁLISIS DE RESULTADOS

5.1. Datos Utilizados

Como se mencionó en la tabla 4.1, el corpus utilizado para el entrenamiento de vectores, consta de alrededor de 3.1 millones de comentarios, de los cuales cerca de 600 mil corresponden a Facebook, y el resto a Twitter.

Antes de analizar el corpus completo, este fue pre-procesado, utilizando un script realizado en Python que permite limpiar texto según las reglas expuestas en la sección 3.5.4. Esto permite que el texto presente palabras más homogéneas y que la utilización de un punto inmediatamente después de una palabra, no sea considerada como una palabra distinta, por ejemplo “saludos.” y “saludos” sería tomada como dos palabras distintas cuando se realice la transformación hacia vectores. Si bien ambos vectores se encontrarán cercanos en el espacio, es mejor permitir que exista una mayor cantidad de una misma palabra para que su vector asociado sea más fiel a lo que representa.

El corpus consta de 527.605 palabras diferentes, esto sin considerar que existen en él, palabras que significan lo mismo, pero se encuentran escritas de distinta forma. Si bien la cantidad es alta, existe un pequeño problema con respecto a que existen en él, una gran cantidad de palabras que sólo se repiten 1 vez en el corpus, el total es de 263.987 palabras, lo que representa aproximadamente un 50 % del total de palabras diferentes. Ahora esta cantidad disminuirá aún más, dado que los algoritmos de vectorización, para que la representación vectorial de una palabra tenga una buena representación, necesitará que esta se repita más de una cierta cantidad de veces durante el corpus, por lo que si se toma como referencia un

Palabra	**Tipo**
#esdomingo	Hashtag
#muchovalor	Hashtag
13d	Fecha
04/04/2015	Fecha
19-20	Fecha
ajjasa	Risa
@luis190167	Cuenta
saklsajsjajskakajs	Risa
@gabytaquezada	Cuenta
89625915	n° Teléfono
quishera	Falta Ortográfica
222xxx935	n° Teléfono
@jaratv	Cuenta
vasilemos	Falta Ortográfica
stf4759631y	Indefinido
nº9xxxx891	n° Teléfono
nº712xxxx33	n° Teléfono
#needsolution	Hashtag

Tab. 5.1: Palabras únicas dentro del Corpus.

mínimo de 5 veces, el total de palabras distintas disminuye a 129.640 lo que representa un 24.57 % sobre el total.

La existencia de palabras únicas dentro del corpus, se debe a diversos motivos, algunas de ellas puede apreciarse en la tabla 5.1. Normalmente se tratan de palabras que pertenecen a un nombre de cuenta, a caracterización de una risa, números telefónicos y ruts, estos últimos se debe, a que al ser comentarios del área de Telecomunicaciones, algunas empresas ayudan a sus clientes por medio de la red social, por lo que algunos usuarios entregan dicha información para obtener alguna ayuda. Una forma de mitigar esto, podría ser la de mejorar el script asociado a la limpieza, el cual podría, transformar por ejemplo los distintos ruts a una palabra en común, como rut o números de teléfono a $ntelefono$, cosa de poder agrupar conceptos similares bajo una misma palabra, permitiendo no tan sólo disminuir la cantidad de palabras únicas, si no además ayudar a los programas a utilizar, que tengan información sobre qué es lo que se está analizando, permitiendo quizás mejorar su desempeño. Puede

11 Palabras/Caracteres más repetidos.	
Palabra	Cantidad
,	1.800.405
de	1.660.809
!	1.310.692
y	1.260.372
que	1.120.546
el	1.073.964
no	938.891
la	893.947
?	865.371
a	824.923
en	767.673

Tab. 5.2: 11 primeras Palabras/Caracteres más repetidos.

ocurrir también, que dichos conceptos no aporten información relevante a la red, a la hora de realizar su predicción.

5.1.1. Ley de Zipf

Una manera de verificar la distribución de las palabras dentro del corpus, es a través de la ley de Zipf, la cual fue formulada en la década de 1940 por George Kingsley Zipf, lingüista de la Universidad de Harvard. La ley de Zipf dice que en general, la mayoría de las veces en un texto cualquiera, independiente del idioma, la segunda palabra más usada aparecerá la mitad de veces que la palabra más usada, la tercera palabra será un tercio de la primera, y así sucesivamente (Tabla 5.1). Incluso tiene vigencia no solamente en el lenguaje en general, sino que puede ser observada en una obra de un escritor en particular, donde el uso de las diez primeras palabras más utilizadas llega a ser alrededor de un 25 % del texto.

Fig. 5.1: Representación Vectorial

En la tabla 5.1 se puede observar un pequeño listado de las palabras y caracteres más utilizados en el corpus, considerando solamente las palabras y ordenándolas según la frecuencia de ocurrencia se obtiene la figura 5.2, que se asemeja bastante a la ley de Zipf (Primeras 100 palabras). Dado que todo lo que se encuentre entre dos espacios se considera como palabra (en este contexto), los caracteres también se encuentran en el gráfico.

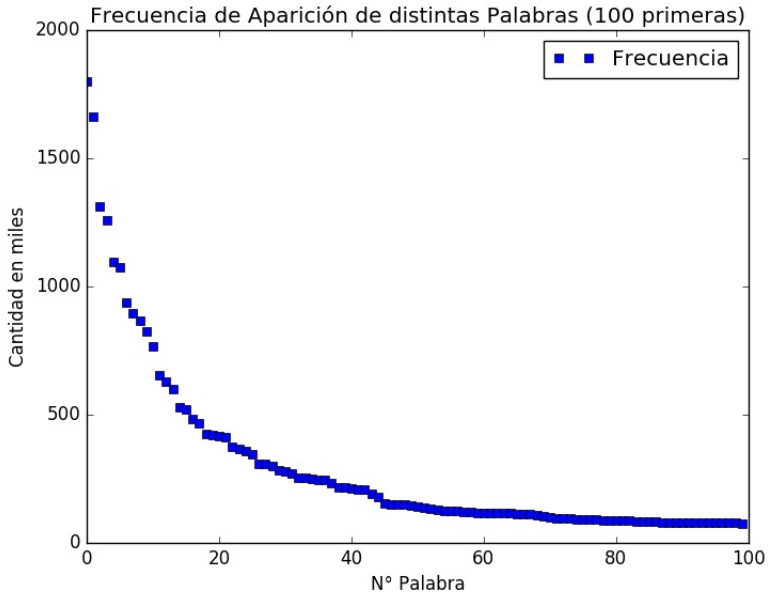


Fig. 5.2: Gráfico de frecuencia de las primeras 100 palabras en el corpus.

Las palabras más utilizadas por los hispanohablantes en orden descendente, son; $de, la, que, el, en, y, a, los, se, del$, esto según el listado de frecuencias del CREA (Corpus de Referencia

del Español Actual [51]), que elabora desde 1996 la Real Academia Española (RAE). Si bien no se encuentran en el mismo orden, 7 de esas 10 palabras se encuentran en la tabla 5.1, lo que demuestra un cierto grado de validez del corpus utilizado. La palabra "no" también se encuentra en este listado, lo que puede demostrar en cierta medida, que la mayoría de los comentarios presentes en el corpus tienden a estar cargados con frases negativas, lo que queda de manifiesto al observar la tabla 4.2 del capitulo 4, donde los comentarios negativos, según el dataset provisto representan casi un 20 % mientras los positivos no alcanzan el 1 %.

5.2. Vectorización de Palabras

La vectorización es realizada utilizando FastText y Word2Vec, en ambos, se generaron vectores utilizando los dos tipos de algoritmos, Skip-Gram y CBOW. Además, para cada entrenamiento se utilizaron los mismos parámetros, estos fueron los siguientes:

- Épocas: 10 épocas de entrenamiento.
- Dimensión: Se crean vectores de dimension 300.
- Tamaño Ventana: 7. El tamaño de ventana, implica la cantidad de palabras que serán tomadas en cuenta, al momento del entrenamiento. Estas palabras son las que son utilizadas como contexto sobre la palabra objetivo.
- Cuenta Mínima: 5. Es la cantidad de veces como mínimo que debe aparecer la palabra dentro del corpus, para que esta sea tomada en consideración, y se genere un vector asociado a ella. Si la palabra se encuentra en una menor cantidad, no tendrá un vector asociado.

Para analizar previamente los resultados, es que se ha utilizado una librería llamada ANNOY (Approximate Neares Neighbors [52]), librería desarrollada por Spotify [1] para recomendar música similar a los gustos del usuario, esto por medio de representación vectorial de las canciones. En resumen, ANNOY es capaz de buscar los n vectores más cercanos a un

[1] Aplicación multiplataforma empleada para la reproducción de música vía streaming.

Comparación con palabra objetivo “Santiago”				
Word2Vec			FastText	
Palabras	Distancia		Palabras	Distancias
stgo	0.68896		centro	0.81609
concepcion	0.97124		metropolitan	0.88227
temuco	0.97462		metropolitano	0.96617
talca	0.98417		metropolis	0.95275
chillan	0.98970		region	0.96978
antofagasta	0.99925		santigo	0.99315
conce	1.00370		stg	1.00511
iquique	1.01986		santa	1.07441
chimkowe	1.02207		stgo-lo	1.07564
santigo	1.02517		yungay	1.07963

Tab. 5.3: Comparación “Santiago” CBOW Word2Vec - FastText

vector específico.

Utilizando ANNOY se desarrolla un script en Python, el cual es capaz de encontrar los 10 vectores más cercano a un vector objetivo, este vector representa a una palabra en particular, por lo que realmente estaremos buscando las 10 palabras más cercanas a una en particular. Las siguientes pruebas fueron realizadas comparando los resultados de la vectorización de CBOW usando ambos programas.

En la tabla 5.2 se observa las 10 palabras más cercanas a $santiago$, por cada palabra se muestra la distancia euclidiana entre ambos vectores (palabras), mientras menor sea esta distancia, más similares serán las palabras. Dentro de los primeros 10 de Word2Vec, aparecen nombres de otras ciudades y algunas otras formas que se utilizaron para referirse a $santiago$ como por ejemplo $stgo$. Por su lado FastText, su lista se basa en distintas formas de referirse a $Santiago$ más que en nombrar otras ciudades.

En la tabla 5.2, se muestran las 10 palabras más cercanas a $movistar$, por su parte Word2Vec (lista izquierda) relaciona otras compañías del mismo rubro, tales como $entel$, $movistar$, wom, $claro$ a diferencia de FastText (lista derecha), que nuevamente relaciona

Comparación con palabra objetivo “Movistar”				
Word2Vec			FastText	
Palabras	Distancia		Palabras	Distancias
entel	0.69527		movistat	0.70464
@movistarchile	0.74492		movista	0.74782
claro	0.75091		movistsr	0.76536
vtr	0.80607		movistarchile	0.77762
@clarochile_cl	0.85828		movistal	0.78808
mi	0.89214		movistarsin	0.79725
pero	0.89368		vovistar	0.79732
...	0.89612		ovistar	0.81086
wom	0.90184		@3gmovistarchile	0.82784
no	0.90454		movistr	0.84802

Tab. 5.4: Comparación “Movistar” CBOW Word2Vec - FastText

distintas formas de decir Movistar, donde la mayoría de ellas son palabras con faltas ortográficas.

Por último, se analiza la palabra $conectar$, la cual se aprecia en la tabla 5.2. Nuevamente Word2Vec (lista izquierda) relaciona la palabra con sinónimos, como por ejemplo $navegar, acceder, cargar$, que a diferencia de FastText (lista derecha), que a pesar de tener algunas palabras de uso similar (aunque sean antónimos), como $desconectar, reconectar$, se relaciona más con palabras con faltas ortográficas que se asemejan a la palabra objetivo.

Es difícil poder a partir de este análisis, concluir cual de ellos es mejor, puesto que ambos cumplen con la función principal. Aunque dentro de las 10 primeras palabras en FastText sólo existan variaciones en la forma de escribir la palabra objetivo, estas poseen mucha menor distancia que las presentadas por Word2Vec (Por ejemplo con “Movistar” y “Conectar”), lo que implica que pudo relacionar de una mejor manera de que en realidad apuntan a una misma palabra. Por su lado Word2Vec representa de mejor manera palabras totalmente distintas bajo un mismo concepto, como el caso de “Movistar” donde pudo relacionar nombres de compañías del mismo rubro, pero, puede ser que FastText también haya obtenido aquellos resultados, sólo que no se encontraban dentro de la lista de las más cercanas.

Comparación con palabra objetivo “Conectar”				
Word2Vec			FastText	
Palabras	Distancia		Palabras	Distancias
conectarme	0.75625		conectarm	0.60347
conectarse	0.87338		conectarlo	0.63777
conecta	0.93168		conectarla	0.65517
navegar	0.93222		conectarle	0.65601
acceder	0.93605		desconectar	0.66239
entrar	0.94846		conectare	0.72805
usar	0.95467		reconectar	0.72818
conecte	0.96647		conectaron	0.74308
conecto	1.00169		conectarse	0.74706
cargar	1.00656		conectara	0.75462

Tab. 5.5: Comparación “Conectar” CBOW Word2Vec - FastText

La decisión sobre cual algoritmo utilizar en este estudio, no es posible deducirlo de estos resultados, por lo que el método a utilizar, para corroborar que uno presenta mejores resultados que otro, es la de entrenar un mismo modelo, con las diferentes vectorizaciones de palabras obtenidos.

5.3. Gráficos de Resultados

Los siguientes resultados, fueron obtenidos realizando el procedimiento elaborado en la sección 4.8, la cual involucra tres etapas de desarrollo, las dos primeras ligadas a encontrar la mejor combinación entre modelo y dataset de entrenamiento, para posteriormente, en la última, realizar la comparación final entre la solución propuesta y la plataforma de la empresa Wholemeaning.

5.3.1. Resultados: Etapa 1

Para la obtención del modelo de red convolucional, es que se han variado tanto los parámetros internos como la estructura de la red. Esta última modificación, no genera cambios

significativos en el resultado de acierto en la fase de prueba de cada modelo.

La estructura base, es la presentada en la sección 4.6, mientras que los parámetros propuestos, son; Un Dropout de 0 % para la red Feedforward, utilización de SGD (Gradiente Estocástico descendiente) con un Learning Rate de 0.001. Mientras que la función de activación de cada capa convolucional es una Relu.

Como se mencionó en las secciones anteriores, el código implementado utiliza Keras como la librería de implementación de modelos de Deep Learning, utilizando Tensorflow como su motor de procesamiento. Al ser una gran cantidad de comentarios, es que se diseño un pre-procesamiento por medio de Python, que es capaz de separar los comentarios por medio de un Batch de tamaño 100, es decir por cada Batch de entrenamiento, se ingresarán 100 comentarios de Twitter.

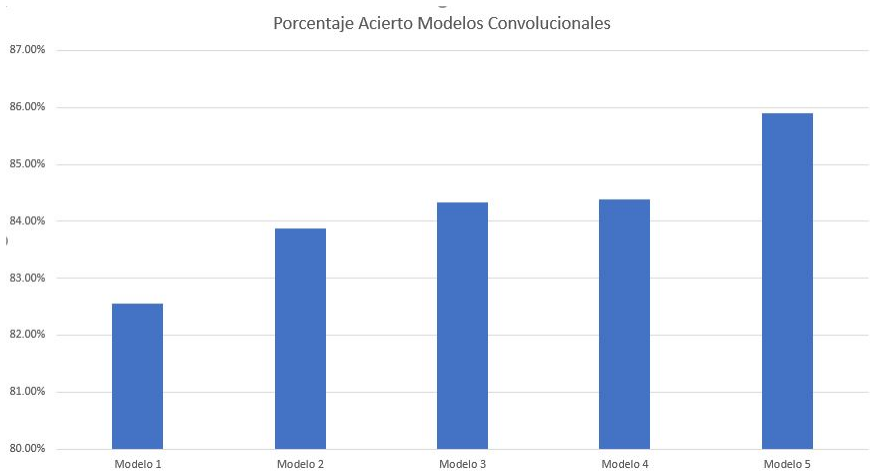


Fig. 5.3: Resumen Comparación asertividad modelos convolucionales

Como se aprecia en la figura 5.3, 5 fueron los modelos que representan distintas variaciones tanto en su estructura como en los valores de sus parámetros (datallados en la tabla 5.3.1). Como se detallará a continuación, la poca variación de sus resultados a partir de modificaciones en la estructura de la red convolucional, ha llevado a que se consideren sólo estos

Modelo	Característica
Modelo 1	Modelo con parámetros y estructura inicial
Modelo 2	Estructura Inicial, 4ta, 5ta 6ta capa Conv, con doble de filtros.
Modelo 3	Estructura Inicial, Dropout 75 %
Modelo 4	Estructura Inicial, Dropout 50 %
Modelo 5	Estructura Inicial, Dropout 50 %, RMSPROP

Tab. 5.6: Tipos de modelo al realizar variaciones en la estructura y parámetros internos de la Red Convolucional.

5 modelos.

Dentro de las modificaciones de la estructura interna de un modelo, se encuentran; número de capas Convolucionales, número de capas de Pooling, número de filtros por capas, número de capas de red Feedforward, tamaño de filtros, tamaño de Pooling. El análisis en detalle de la modificación de cada factor no será expuesto en este estudio, básicamente porque su impacto real en el porcentaje de acierto de los distintos modelos, no afecta en gran medida. Esto, siempre y cuando las variaciones realizadas en él, no hacen aumentar el nivel de acierto de la red inicial junto a los valores de sus parámetros. Al tratarse de un dataset de entrenamiento con 1.6 millones de Tweets (Dataset Sentiment140), es que la modificación de la estructura, si afecta en el tiempo total de entrenamiento, donde si bien, una estructura más simple, es decir con una menor profundidad (menor cantidad de capas) logra disminuir considerablemente el tiempo de entrenamienta, esta al tener un desempeño peor que la estructura propuesta, no se ha tomado en consideración. Una modificación que si impactó en el porcentaje de acierto, es el aumento significativo de filtros en las capas más profundas de convolución (modelo 2), que si bien aumentó el porcentaje final del modelo 1 (Figura 5.3), también lo hizo en su tiempo de entrenamiento. Mientras la estructura inicial demora cerca de 1 hora en procesar todos los comentarios durante una época de entrenamiento, el Modelo 2 lo realiza en casi una hora y media.

Los parámetros internos de la red, que fueron modificados, son el porcentaje de Dropout que posee la red de clasificación, el porcentaje de Learning Rate y la función de activación,

Modelo	Característica
Modelo 1	Sin Embedding Usando Word2Vec
Modelo 2	Sin Embedding Usando FastText
Modelo 3	Con Embedding Usando Word2Vec
Modelo 4	Con Embedding Usando FastText
Modelo 5	Con Embedding Sin Vectorizacion
Modelo 6	Sin Embedding Usando Word2Vec - MaxLen = 100
Modelo 7	Sin Embedding Usando Word2Vec - MaxLen = 200
Modelo 8	Sin Embedding Usando Word2Vec - MaxLen = 300
Modelo 9	Sin Embedding Usando Word2Vec - MaxLen = 400
Modelo 10	Sin Embedding Usando Word2Vec - MaxLen = 450
Modelo 11	Sin Embedding Usando Word2Vec - CommentTtraining = 12500
Modelo 12	Sin Embedding Usando Word2Vec - CommentTtraining = 25000
Modelo 13	Sin Embedding Usando Word2Vec - batch size = 64
Modelo 14	Sin Embedding Usando Word2Vec - batch size = 32

Tab. 5.7: Tipos de modelo al realizar variaciones el tipo de vectorización y la cantidad de comentarios.

permutándola también con una función Sigmoid. El porcentaje Dropout afectó de manera positiva en el resultado final de la red, permitiendo que esta suba en su porcentaje de acierto, el iniciar la red con un 0 % no permite que esta mejoren todas las neuronas, por lo que el aumento de este porcentaje, mejora en su entrenamiento, reflejándose esto en su respuesta al dataset de prueba. Esto se puede observar en los modelos 3,4 y 5, donde a pesar que el modelo 3 tiene un porcentaje mayor de Dropout, este no mejora el resultado de uno que sólo tiene un 50 %, como lo es el modelo 4. Manteniendo aquel porcentaje, una mejora considerable se alcanzó al utilizar RMSPROP como optimizador de la red, a diferencia de SGD, que es utilizado en el modelo 1.

A diferencia de lo ocurrido con las redes convolucionales, las redes recurrentes LSTM, si presentan una variación más fuerte al realizar variaciones en su estructura y en sus parámetros. Es por eso que se presentará una mayor cantidad de distintos modelos asociados a este tipo de redes.

La tabla 5.3.1 se observan las primeras 14 variaciones, a un modelo base, el cual se trata

de un modelo bidireccional LSTM de 2 capas de 300 células LSTM. Los modelos 1 hasta 5 buscan comparar algoritmos de vectorización de palabras. Modelos 1, 2 y 3 utilizan el método CBOW para su vectorización, incluyendo además en el 3, 4 y 5, una capa inicial llamada capa Embedding, capa que se encuentra disponible en Keras, que permite facilitar el uso de estas redes sobre texto, permitiendo asociar cada palabra a un determinado vector, e ir mejorando sus valores por cada batch de entrenamiento, por lo que no sólo los parámetros internos de cada célula LSTM son actualizados, si no también los vectores asociados a cada palabra. Esto en teoría ayuda a mejorar el desempeño de la red. A diferencia del modelo 5, donde la capa Embedding inicializa los vectores, el modelo 3 y 4 toma como vectores iniciales, aquellos obtenidos por los dos métodos diferentes, como lo son Word2Vec y FastText.

Al estar utilizando el dataset de reviews de IMDB, se presentan una gran variedad de tamaños en los comentarios, algunos alcanzando casi las 2000 palabras. Es por esto, que los modelos desde el 6 hasta el 10 presentan variaciones del total máximo de palabras a considerar, por ejemplo, el modelo 8 sólo considerará las primeras 300 palabras de un comentario, cualquiera que tenga una cifra mayor a esta, será cortado, dejando una porción de este sin analizar. Modelo 11 y 12, varían la cantidad de comentarios totales de entrenamiento, y por último, el 13 y 14, varían la cantidad de comentarios que serán utilizados por cada batch de entrenamiento.

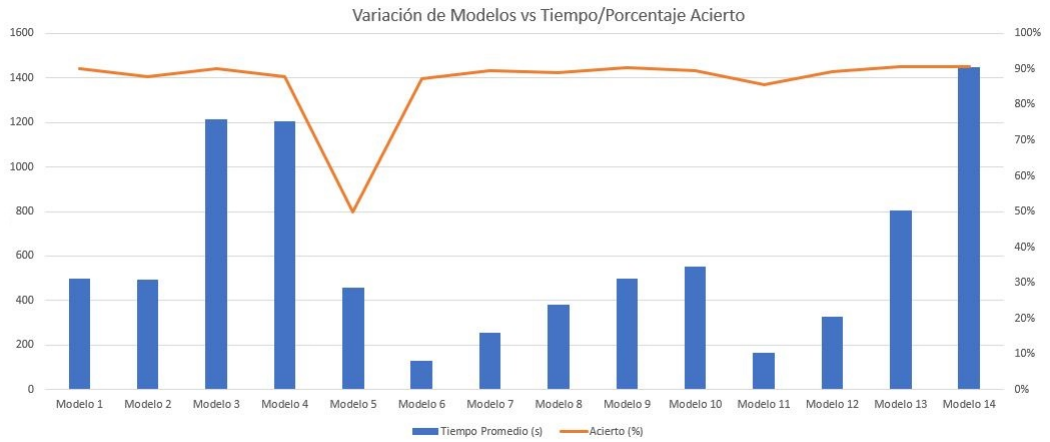


Fig. 5.4: Resumen Comparación

Al observar los resultados de los primeros 14 modelos en la gráfica de la figura 5.4, se aprecia de los primeros 4 modelos no existe diferencia significativa al momento de analizar su porcentaje de acierto, pero si en el tiempo que toma por Epoch entrenar la red, incrementándose casi tres veces el tiempo total al incluir una capa de Embedding. Un caso distinto se observa en el modelo 5, que al no inicializar los valores de los vectores, estos no tienen un buen desempeño, no permitiendo que la red aprenda, presentando con ello el peor resultado. Los modelos del 6 al 10 tampoco presentan gran diferencia en el porcentaje alcanzado, pero si en el tiempo que toma en su entrenamiento por Epoch, aumentando mientras más palabras sean consideradas, esto tiene sentido, ya que el manejo interno de la red será mayor.

Los modelos 11 y 12 presentan una leve diferencia, y es que al sólo considerar 12500 comentarios para la etapa de entrenamiento, han bajado su porcentaje de acierto llegando casi a un 88 %, mientras que considerando el doble de ellos, llega arriba de un 90 %. Por último, la modificación del tamaño de Batch, al disminuirlo desde 128 hasta 64 para el modelo 13 y 32 para el modelo 14 no afectaron en mayor medida en el porcentaje alcanzado por ellas, pero si en los tiempos de procesamiento, alcanzando este último el mayor tiempo de entrenamiento, llegando a los 1400 segundos por Epoch.

Modelo	Característica
Modelo 1	Modelo con Word2Vec CBOW
Modelo 2	Modelo con Word2Vec Skip-Gram
Modelo 3	Modelo con FastText CBOW
Modelo 4	Modelo con FastText Skip-Gram
Modelo 5	Modelo con 2 capas 100 células c/u.
Modelo 6	Modelo con 1 capa 200 células.
Modelo 7	Modelo con 1 capa 400 células.
Modelo 8	Modelo con 2 capas 100 células c/u.
Modelo 9	Modelo con 2 capas 200 células c/u.
Modelo 10	Modelo con 3 capas 100 células c/u.
Modelo 11	Modelo con 3 capas 200 células c/u.
Modelo 12	Modelo con 3 capas 400 células c/u.
Modelo 13	Modelo con 3 capas 400,200,100 células c/u.
Modelo 14	Modelo 2 capas 200 células c/u. RMSprop
Modelo 15	Modelo 2 capas 200 células c/u. SGD
Modelo 16	Modelo 2 capas 200 células c/u. ADAGRAD
Modelo 17	Modelo 2 capas 200 células c/u. ADAMAX
Modelo 18	Modelo 2 capas 200 células c/u. ADADELTA
Modelo 19	Modelo 2 capas 200 células c/u. NADAM

Tab. 5.8: Tipos de modelo al realizar variaciones sobre su estructura.

Los modelos anteriores buscaban modificar parámetros externos a la red, como la capa de Embedding, el largo de los comentarios de entrenamiento o la cantidad de comentarios por Batch. Los siguientes 19 modelos varían tanto estructuras internas de la red, como su número de capas o de células LSTM, como el optimizador utilizado. Estos modelos se pueden observar en la tabla 5.3.1, los primeros 4 varían tanto la librería como el método de vectorización. Los modelos desde el 5 hasta el 13 varían la estructura de la red, aumentando capas y células. Por último los modelos desde el 13 hasta el 19 varían el método de optimización de la red, manteniendo la estructura constante.

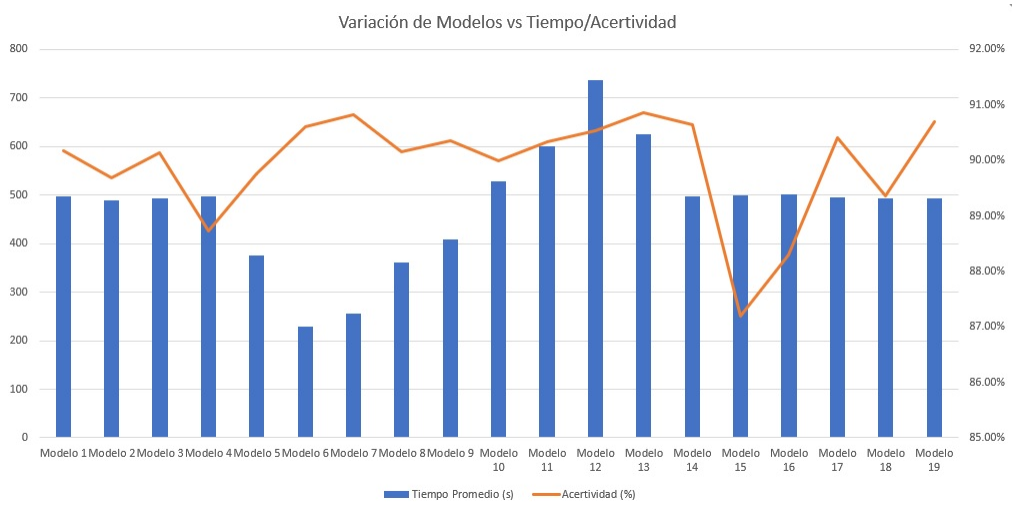


Fig. 5.5: Resumen Comparación

Los resultados obtenidos por estas últimas 19 variaciones, se puede observar el el gráfico de la figura 5.5. Los primeros 4 no presentan diferencia en el tiempo de entrenamiento, pero si en el porcentaje alcanzado de acierto, siendo el método de CBOW el que mejores resultados presenta, tanto para la librería FastText como Word2Vec. De los siguientes modelos, el 7 junto al 13 son los que obtienen un mejor resultado, pero es el primero de ellos, quien tiene un menor tiempo de entrenamiento. Tanto el aumento de capas como de células aumenta el tiempo final pero no logran mejorar el desempeño de la red, manteniendo el porcentaje alrededor de un 90 %.

Dentro de la variación de optimizadores presentes entre los modelos 14 y 19, el RMSPROP junto al NADAM son los que presentan mejores resultados, no existiendo una diferencia significativa en el tiempo de procesamiento. El de peores resultados es el modelo 15 con SGD, junto al modelo 16 y 18 con AGRADAD y ADADELTA respectivamente.

Tal parece ser que las redes LSTM tienen un funcionamiento muy bueno en el análisis de texto, y que sus variaciones si bien afectan variadamente en sus tiempos de procesamiento, estos no presentan grandes cambios en el resultado final de aciertos, incluso superando enormemente los alcanzados por las redes Convolucionales.

Finalmente los modelos seleccionados para la etapa 2 son el modelo 5 de redes convolucionales, es decir con su estrcutura propuesta en la tabla 4.6, pero con un valor de Dropout de 50 % para su capa de clasificación (red Feedforward) y un optimizador RMSPROP. Y finalmente el modelo para la red recurrente, es uno de una capa de 400 células LSTM, utilizando la vectorización de la librería Word2Vec con el método CBOW y utilizando el optimizador RMSPROP.

5.3.2. Resultados: Etapa 2

2 fueron los modelos obtenidos en la etapa previa, pero para realizar un mejor análisis en la presente etapa, es que se han realizado variaciones a estos.

Como se mencionó anteriormente en los modelos propuestos, la red convolucional se entrenará con dos diccionarios distintos, uno teniendo en cuenta los tildes y letras mayúsculas, y otro sin considerar ninguno de ellos, o sea sólo letras en minúscula y sin tildes. También se tendrán la variante por cada modelo, de que un modelo esté compuesto por dos redes distintas, es decir, una que se especialice en predecir si un comentario es positivo o no positivo y otro si es negativo o no negativo. La respuesta de ambas determinará finalmente la respuesta final del modelo. Si una predice positivo y la otra no negativo, determinará que la respuesta

del modelo será positivo,y dado que la mayoría de los comentarios que son catalogados como positivos y negativos a la vez, tienen atribuciones negativas, serán considerados como de clase negativa. Esto último se debe a que este tipo de comentarios, que cae dentro de ambas clases, mayoritariamente son comentarios de tipo sarcásticos, donde a pesar de expresar palabras positivas, su transfondo es negativo.

Los distintos datasets a analizar son los propuestos en sección 4.2. El primero de ellos, el denominado "'Telco Facebook + Twitter" será analizado sólo a través de las redes recurrentes, esto se debe a que al considerar comentarios provenientes de Facebook, sus comentarios pueden tener un largo mayor a los 140 caracteres considerados en las redes convolucionales. Por su lado, dataset "Telco Twitter" es analizado por 4 modelos, de las redes convolucionales no se consideraron la variación de utilizar dos redes en conjunto, dado que no presentaron un gran cambio en sus resultados. En cambio estos si son utilizados con el dataset "Nuevo Telco Twitter", dataset realizado específicamente para el presente estudio. En este último todos los tipos de modelos son puestos a prueba.

Modelo	Dataset	Características	Negativos	Neutros	Positivos	Total	Outliers/Sarcasmo	Total Con Sarcasmo
1	Telco FB + Twitter	1 Modelo LSTM de 3 Clases	40%	88%	66%	65%	10%	61%
2	Telco FB + Twitter	2 Modelos LSTM 2 Clases	34%	96%	63%	64%	12%	61%
3	Telco Twitter	1 Modelo Convolucional 3 Clases Dict. Corto	32%	91%	68%	64%	15%	60%
4	Telco Twitter	1 Modelo Convolucional 3 Clases Dict. Largo	62%	80%	70%	71%	14%	67%
5	Telco Twitter	1 Modelos LSTM de 3 Clases	44%	90%	62%	65%	14%	62%
6	Telco Twitter	2 Modelos LSTM 2 Clases	35%	91%	65%	64%	14%	60%
7	Nuevo Telco Twitter	1 Modelo Convolucional 3 Clases Dict. Corto	86%	82%	67%	78%	31%	75%
8	Nuevo Telco Twitter	2 Modelos Convolucionales 2 Clases Dict. Corto	88%	83%	65%	79%	40%	76%
9	Nuevo Telco Twitter	1 Modelo Convolucional 3 Clases Dict. Largo	65%	68%	81%	71%	13%	67%
10	Nuevo Telco Twitter	2 Modelos Convolucionales 2 Clases Dict. Largo	66%	70%	81%	72%	39%	70%
11	Nuevo Telco Twitter	1 Modelo LSTM de 3 Clases	90%	85%	92%	89%	44%	86%
12	Nuevo Telco Twitter	2 Modelos LSTM 2 Clases	94%	87%	92%	91%	61%	89%
13	Nuevo Telco Twitter (50%-50%)	2 Modelos LSTM 2 Clases	95%	88%	78%	87%	91%	87%
14	Sin Dataset	Plataforma BI Wholemeaning	97%	76%	69%	81%	65%	80%
15	Sin Dataset	Azure Sentiment Analysis	59%	53%	89%	67%	14%	63%

Tab. 5.9: Tabla resumen de comparación entre distintos dataset y modelos.

Además es que se realizó una comparación previa junto a la plataforma de categorización de la empresa y a la plataforma de "Sentiment Analysis" de Azure (Azure Sentiment Analysis [58]). Esto es para tener una comparación previa antes de decidir el modelo que será utilizado en la etapa final.

En la tabla 5.9 se puede apreciar los resultados de cada uno de los modelos al utilizar los distintos tipos de datasets. Es interesante notar que ninguno de los modelos entrenados con los datasets proveídos por la empresa, los cuales estaban etiquetados según su sentimiento (sólo positivo o negativo), presentan buenos resultados en la categorización en las clases de neutros y positivos pero no en negativos. Esto se puede deber a que muchos de los comentarios que son catalogados como insatisfacción no son catalogados como negativos, quedando estos sin categorización, y posteriormente tomados en cuenta de manera errónea como comentarios neutros. Dado esto es que quizás las redes no posean tantos comentarios catalogados como negativos, que permitan que ella aprenda a diferenciarlos de los neutros, haciendo que finalmente sean catalogados como neutros.

Una gran diferencia marca cada uno de los modelos entrenados con el dataset "Nuevo Telco Twitter", que a pesar de no poseer una gran cifra de comentarios de entrenamiento, como si lo tienen los datasets anteriormente utilizados, si obtiene un muy buen resultado y una mejora considerable con los anteriores modelos. Esta diferencia se marca aún más al comparar los resultados de las redes convolucionales con las recurrentes LSTM, siendo está última la que mejores resultados obtiene.

Tomando sólo en consideración los resultados obtenidos al utilizar el dataset especialmente confeccionado para el presente estudio, es que existe una diferencia apreciable en los modelos convolucionales al utilizar un diccionario corto, es decir sin tomar en consideración la mayúsculas ni los tildes de las palabras. Estas presentan un mejor resultado, donde en promedio aumentan en un 9 % su porcentaje de acierto, y si la comparación se realiza entre un modelo de una red de 3 clases, contra un modelo de dos redes de 2 clases (que categoriza 2 clases sólamente, positivo - no positivo o negativo - no negativo), estos últimos tienen una mejora de 1 % en su porcentaje de acierto total.

Esto igualmente ocurre con los modelos recurrentes LSTM, ya que los modelos que tie-

nen 2 redes que sólo predicen 2 clases, presentan un mejor desempeño a la hora de etiquetar comentarios, aumentando en un dos porciento el total, al realizar la comparación entre el modelo 11 y 12. Los modelos desde el 7 hasta el 12 fueron entrenados con una proporción de 33 %-66 %, es decir que por ejemplo, si se quere determinar las clases "Negativo" y "No Negativo", se utilizaron los 1800 comentarios categorizados como negativos, más los 3600 comentarios categorizados como no negativos, siendo esta la suma entre positivos y neutros. El modelo 13 intenta variar este porcentaje, tomando sólo 900 de positivos y neutros para establecer la clase no-negativos, lo que no logra aumentar el porcentaje global de acierto, disminuyéndolo hasta un 87 %, aunque este último logró aumentar su porcentaje en los comentarios tanto neutros como negativos.

Por último también se realizó una prueba con un pequeño dataset de 25 comentarios asociados a sarcasmo, un par de comentarios ejemplo puede observarse en 5.1 y 5.2.

$$@miclaroCL\ Me\ queda\ claro\ entonces\ que\ ustedes\ privilegian\ losclientes\ nuevos\ que\ los\ clientes\ antiguos,\ excelente\ la\ retencion\ que\ tienen \tag{5.1}$$

$$@AyudaMovistarCL\ la\ ra@@!\ Es\ maravilloso\ que\ respondan\ 2\ dias\ despues. \tag{5.2}$$

Donde si bien no es una prueba directamente relacionada con el propósito de esta etapa, si demuestra una gran mejora de los modelos a partir de 2 redes neuronales con respecto a los que poseen sólo una red que clasifica las 3 clases. Esto se debe principalmente que al tener redes especializadas en detectar tanto comentarios negativos como positivos, un comentario sarcástico será detectado por ambas redes, y como se mencionó anteriormente que los comentarios etiquetados por ambas clases pertenecen en su gran mayoría a la clase negativa, estos son finalmente etiquetados en dicha clase. Esto no ocurre con los modelos que poseen una clase, ya que ella está diseñada a que prediga sólo una clase, siendo la clase positiva la

más mencionada en esta prueba, seguida por la neutra.

Finalmente al realizar una comparación con los resultados obtenidos con la plataforma BI de la empresa y la plataforma de Azure, los modelos propuestos presentan un mejor desempeño. Azure si bien tiene un buen desempeño en la categorización de clases positivas, estas no se ven reflejadas en las clases tanto negativa como neutra, esto se debe mayoritariamente (en teoría) a que al ser una plataforma diseñada con un idioma español estándar, esta no es capaz de capturar expresiones típicas del español chileno. La plataforma de la empresa Wholemeaning posee un muy buen desempeño en la categorización de comentarios negativos, esto a diferencia de los datasets anteriores, se incluyeron todos los comentrios catalogados como insatisfacción, que son la mayoría, como comentarios negativos, llegando a alcanzar el 97 % de acierto, aún así, en un análisis global, este no supera al alcanzado por el modelo 12, siendo este el que mejor resultados entrega, por lo que la etapa final, tendrá en consideración la combinación de factores del modelo 12, es decir, se utilizará el dataset "Nuevo Telco Twitter" junto a un modelo que posee dos redes neuronales recurrentes LSTM.

5.3.3. Resultados: Etapa 3 y Final

La etapa final, es una comparación final entre la solución propuesta y la plataforma de la empresa Wholemeaning. Para ello es que se diseñó un dataset final de prueba con 600 comentarios en total, 200 para cada clase. Donde cada uno de los comentarios fueron realizados a partir del 1 de noviembre del 2017, lo que implica que se encuentran totalmente fuera del dataset utilizado para entrenar el modelo de Deep Learning.

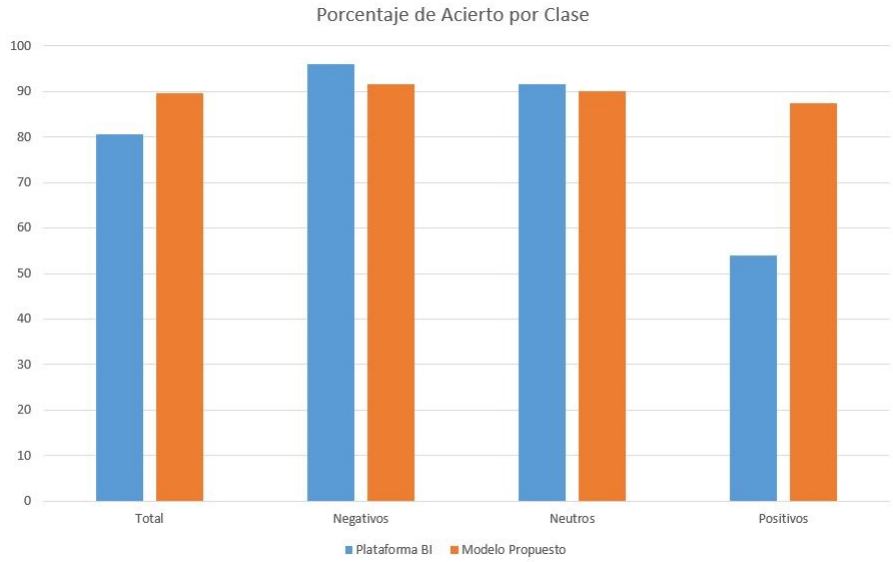


Fig. 5.6: Comparación final entre modelos.

El gráfico de la figura 5.6 muestra el resultado comparativo a partir de la predicción realizado por ambas soluciones. Se puede apreciar en él, que tanto en la clasificación de comentarios neutros como negativos la plataforma de la empresa tiene un mejor resultado, alcanzando un 96 % de precisión en negativos y un 91.5 % en neutros, mientras que el modelo propuesto alcanza un 91.5 % y un 90 % respectivamente. La diferencia se marca en la clase positiva, donde el modelo realizado con una red recurrente LSTM alcanza el 87.5 % de acierto versus sólo un 54 %. En total, se obtiene un resultado global de un 89.6 % versus un 80.5 % de la plataforma.

Si bien este es una comparación válida, ya que es realizada tomando en consideración los comentarios bien catalogados, esto no logra demostrar bien la diferencia existente entre ambas soluciones. Esto se debe básicamente a que es posible que el alto porcentaje de la predicción de una clase, sea a base de fallar muchas otras. Es por eso, que para realizar una buena comparación de modelos predictivos, es necesario hacer un análisis de curva ROC.

Se denomina curva ROC (de su acrónimo Receiver Operating Characteristic o Caracterís-

tica Operativa del Receptor), dado que representa distintas respuesta de un modelo predictivo al variar parámetros internos de él, lo que forma una curva en su gráfica final. En este análisis final, sólo se se presentarán puntos asociados a dicha curva, es decir no se realizarán variaciones en los parámetros para cambiar el comportamiento del modelo. La gracia de utilizar este método, es que no sólo toma en consideración el nivel de precisión que se tiene, si no además, que tan confiable es la predicción que realiza, o en otras palabras, si un modelo predice tal clase, que tan seguro es que realmente pertenezca a ella.

538	Deep L.	Negativo	Neutro	Positivo	% Correcto	% Incorrecto
	Negativo	183	11	3	92.89	7.11
	Neutro	15	180	22	82.95	17.05
	Positivo	2	9	175	94.09	5.91

483	Plat. Bi.	Negativo	Neutro	Positivo	% Correcto	% Incorrecto
	Negativo	192	11	27	83.48	16.52
	Neutro	8	183	65	71.48	28.52
	Positivo	0	6	108	94.74	5.26

Tab. 5.10: Matriz de confusión entre modelo de Deep Learning y Plataforma BI. Parte superior matriz de modelo Deep Learning, en verde indica las predicciones correctas, siendo un total 538 los comentarios bien etiquetados. Parte inferior resultados plataforma de la empresa, teniendo un total de 483 comentarios bien etiquetados.

La construcción de las tablas 5.11, 5.12 y 5.13, son realizadas con la ayuda de una matriz de confusión, la cual puede apreciarse en las tablas 5.10. Donde por ejemplo, para la obtención de los Valores Positivos asociados a la clase $Negativa$ del modelo propuesto, se tiene que tomar en consideración sólo los datos donde la red predijo negativo, y el comentario realmente era negativo, por ende VP tiene un valor de 183. Para los $FalsosPositivos$, estos se consideran todo aque que el modelo predijo como $Negativo$ pero no lo era, es decir $11 + 3 = 14$. Para los $FalsosNegativos$, se asocian la cantidad de veces que la red dijo que no era $Negativo$ pero que realmente si pertenecían, es decir $15 + 2 = 17$. Para los $VerdaderosNegativos$ se suman todos los datos que donde la red predijo que no eran negativos y que realmente no pertenecían a dicha clase, esto suma $180 + 22 + 9 + 175 = 386$.

Tab. 5.11: Análisis clase negativa

Clase - Negativo		Categorización Real	
		Positivo	Negativo
Categorización de Modelo Propuesto	Positivo	VP = 183	FP = 14
	Negativo	FN = 17	VN = 386
	Sensibilidad	0.915	
	Especificidad	0.965	
Categorización Plataforma Empresa	Positivo	VP = 192	FP = 38
	Negativo	FN = 8	VN = 362
	Sensibilidad	0.96	
	Especificidad	0.905	

Con esto ya es posible obtener los factores de $Sensibilidad$, factor asociado al porcentaje de acierto que tuvo el modelo (fórmula 5.3) y la $Especificidad$, este último asociado a que tan confiable es el modelo al predecir una cierta clase (fórmula 5.4).

$$Sensibilidad = VP/(VP + FN) \tag{5.3}$$

$$Especificidad = VN/(FP + VN) \tag{5.4}$$

La tabla 5.11 muestra los cálculos realizados, en él es posible apreciar que la razón de éxitos (VPR o Sensibilidad) es menor el del modelo propuesto al compararlo con la plataforma, caso contrario ocurre con su especificidad. Esto implica que si bien el modelo de la empresa tuvo un mayor porcentaje de acierto en la clase negativa, predijo muchos comentarios como negativos, siendo que pertenecían a otra clase. Esto se observa en los Falsos Positivos, siendo un total de 38, es decir, 38 veces el modelo predijo un comentario como negativo, siendo que estos eran o neutros o positivos. Caso contrario en el modelo propuesto, donde sólo existen 14 comentarios mal predichos, resultando esto en una especificidad mayor.

Lo anterior también vuelve a repetirse al analizar los comentarios asociados a la clase $Neutra$. En la tabla 5.12, se observa el resultado asociado a ella, donde nuevamente el total de falsos positivos es mucho mayor en la plataforma de la empresa que en el modelo propues-

Tab. 5.12: Análisis clase neutra.

Clase - Neutra		Categorización Real	
		Positivo	Negativo
Categorización de Modelo Propuesto	Positivo	VP = 180	FP = 37
	Negativo	FN = 20	VN = 363
	Sensibilidad	0.9	
	Especificidad	0.9075	
Categorización Plataforma Empresa	Positivo	VP = 183	FP = 73
	Negativo	FN = 17	VN = 327
	Sensibilidad	0.915	
	Especificidad	0.8175	

Tab. 5.13: Análisis clase positiva

Clase - Positivo		Categorización Real	
		Positivo	Negativo
Categorización de Modelo Propuesto	Positivo	VP = 175	FP = 11
	Negativo	FN = 25	VN = 389
	Sensibilidad	0.875	
	Especificidad	0.9725	
Categorización Plataforma Empresa	Positivo	VP = 108	FP = 6
	Negativo	FN = 92	VN = 394
	Sensibilidad	0.54	
	Especificidad	0.985	

to. Y al igual que en la clase negativa, el total de $FalsosNegativos$ es mayor en el modelo propuesto, mostrando nuevamente que la plataforma tiene un mayor porcentaje de acierto, aún así, su $Especificidad$ es menor que la del modelo propuesto.

A diferencia de las clases anteriores, en los resultados asociados a la clase positiva (tabla 5.13) se presenta con los datos al revés, es decir, la plataforma presenta una mejor $Especificidad$ al equivocarse muy poco al catalogar un comentario como positivo. Pero esto se ve compensado al predecir una mayor cantidad de comentarios correctamente.

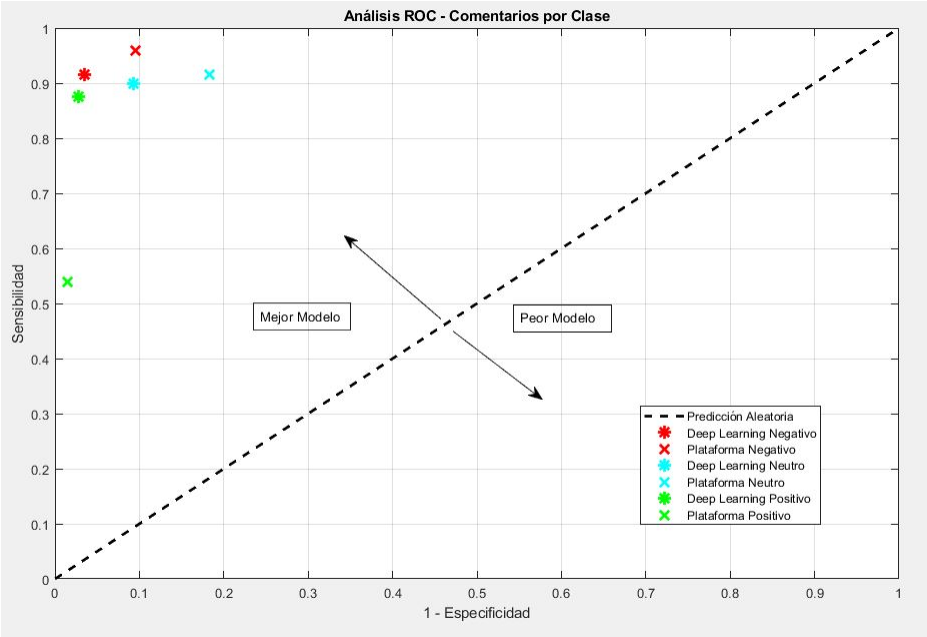


Fig. 5.7: Análisis ROC por cada clase, entre modelo propuesto (Deep Learning) y Plataforma de la Empresa Wholemeaning. Mientras más hacia la esquina superior izquierda se encuentre el punto, esta asociado a un mejor modelo predictivo.

Finalmente, a partir de los datos obtenidos del análisis, es que se gráfica en una curva ROC los resultados finales, los cuales pueden ser apreciados en la figura 5.7. En ella es posible comparar por Modelo-Clase, cuales de las soluciones presenta un mejor desempeño global. Mientras más cargado hacia la esquina superior izquierda se encuentre un punto, indicará que es un mejor modelo predictivo. Su eje y indica la precisión de sus aciertos (su sensibilidad), mientras que el eje x indica la confiabilidad de su predicción por medio de $1 - Especificidad$.

Como es posible apreciar en la figura 5.7, el modelo propuesto ofrece un mejor desempeño en la categorización de comentarios que la herramienta de la empresa Wholemeaning, indicando que a pesar de que no logra superar la precisión alcanzada por dos de las clases, esta solución presenta un mayor grado de confianza al momento de su predicción, ya que su porcentaje de error asociado a una predicción ya hecha, es menor que la plataforma.

6. CAPÍTULO: CONCLUSIONES

Mediante el desarrollo del presente trabajo, se ha podido observar el real alcance que ha tenido durante el último tiempo el desarrollo de la inteligencia artificial, específicamente el área de Deep Learning. Y es que los resultados obtenidos por medio del uso de redes recurrentes LSTM han demostrado funcionar de una manera tal, que a partir de un pequeño dataset de entrenamiento, se ha podido mejorar el desempeño de herramientas que han sido desarrolladas durante años, como es el caso de la plataforma de la empresa Wholemeaning.

La motivación del presente estudio nace sobre la necesidad de poder mejorar el sistema de categorización de sentimiento presente en la empresa Wholemeaning, mediante el uso de algoritmos del procesamiento natural del lenguaje. La imposibilidad que se tiene de poder mejorar el porcentaje de acierto y su alcance, a llevado a probar estas nuevas metodologías y estudiar una posible futura implementación. El foco principal, se centraba en la utilización de 2 redes neuronales profundas distintas, las convolucionales y las recurrentes LSTM. Siendo cada una de ellas ayudada por distintas metodologías de procesamiento de texto, permitiendo así que el manejo por parte de estos algoritmos tengan una mayor y mejor incidencia en sus resultados.

Con la motivación de poder probar estas metodologías en el ámbito chileno, haciendo especial hincapié en la forma que tienen los chilenos de expresarse, es que ha intentado adaptar estas nuevas tecnologías a un contexto nacional. Una de ellas es la vectorización de texto, la cual permitió de gran forma poder representar matemáticamente a través de vectores, las palabras de uso regular en el idioma español chileno, cuyos resultados demuestran el gran potencial que tienen actualmente. Las pruebas realizadas en el presente estudio han dado pie

incluso, a que sean considerados como una herramienta dentro de la plataforma de la empresa, permitiendo así a los lingüistas buscar palabras similares a una palabra objetivo.

Si bien los algoritmos implementados para la vectorización (Word2Vec y FastText) demostraron resultados distintos, no fue posible determinar cual de ellos era mejor, esto en gran medida dado que, ambos resultados eran muy buenos pero de manera distinta. Word2Vec por su lado, lograba relacionar entidades, como lo fue por ejemplo con la palabra "Movistar", la cual la relacionó con palabras de empresas del mismo rubro, como "Entel" o "Claro", mientras que FastText la relacionaba con variaciones de la misma palabra (faltas ortográficas), esto dado la forma en que tiene esta librería de trabajar. Aunque ambos resultados distaban muchos entre si, la única forma de poder corroborar que uno de ellos funciona mejor que el otro, es la de comparar los modelos de redes recurrentes con cada uno de los métodos, y observar sus resultados. Al realizar esa comparación, el método de CBOW de Word2Vec logró una ligera mejora en contra de CBOW de FastText, y ambos métodos superaron a los vectores entrenados con el método de Skip-Gram.

La calidad de los datos utilizados es otro punto a considerar, ya que quedó demostrado que la calidad de ellos está por sobre su cantidad. Esto no tan sólo radica en que la utilización de un dataset esté debidamente confeccionado, si no además de la manipulación que se tiene de él al momento de pre-procesarlo. Es decir, generar un formato tal que permita una mejor manipulación por parte de la red, intentando generar pocas variaciones de una misma palabra (al descartar mayúsculas o tildes), aumentando con ello su representatividad, como lo es con el caso de la vectorización.

En cuanto a los modelos, el método implementado con redes convolucionales no demostraron ser rival de las recurrentes LSTM. Si bien en algunos casos alcanzó casi el 80 % de precisión en la predicción de 3 clases, esta no es competencia para el resultado de la otra alternativa. Y es que la representatividad de letras por medio de diccionario de letras, no parece ser una vía optima a seguir, dado lo poco representativa que se vuelve la palabra. Quizás exis-

tan otras metodologías que capturen de mejor manera una palabra y que logre representarla de tal manera que se asimile a una imagen, para que pueda ser utilizada con redes convolucionales. Claro está que la comparación se esta realizando con el quizás, el mejor método que se puede aplicar sobre texto, por lo que aún así, las redes convolucionales generaron resultados bastante aceptables como modelo predictivo.

Por el contrario, las redes recurrentes LSTM mostraron ser una solución idónea para este tipo de trabajos. Aún a pesar del pequeño tamaño del dataset utilizado para entrenamiento, los resultados son muy buenos, incluso casi alcanzando un porcentaje de acierto de un 90 % en la prueba final. Si bien se realizaron una gran variedad de modificaciones tanto a la estructura de la red como a los parámetros internos de ella, en cada uno de sus análisis mostraron tener buenos resultados, teniendo incluso altos porcentajes de acierto al cortar los comentarios de entrenamiento. Esto no hace más que demostrar el potencial tremendo que tienen este tipos de redes en el ámbito de categorización de texto, donde a partir de un puñado de datos, le es posible extraer las características necesarias que le permitan diferenciar unos de otros.

Una modificación que marcó una diferencia en cuanto a resultados, fue la de considerar dos redes dentro de un mismo modelo predictivo, donde cada una de las redes se especializaba en detectar sólo 2 tipos de clases, siendo estas “Negativos y no Negativos” y “Positivos y no Positivos”. La diferencia en sus resultados alcanza incluso el 2 % en algunos casos, permitiendo además mejorar su desempeño en comentarios caracterizados por ser sarcásticos, ya que a diferencia de una red que debe predecir entre 3 clases, estas puede etiquetar un comentario de dos maneras distintas, haciendo que si uno es etiquetado como “Positivo” y “Negativo” a la vez, permita ser reconocido como un comentario sarcástico y por ende ser categorizado como “Negativo”.

El modelo y método propuesto como solución final, tiene aún mucho por ser mejorado, aún así los resultados muestran una vía alternativa a las soluciones ya implementadas, una vía capaz de ser auto mejorada con el tiempo.

7. CAPÍTULO: POSIBLES MEJORAS

A lo largo del trabajo, se ha demostrado que por sobre la cantidad de datos, su calidad afecta de una mejor y más directa manera en los resultados. Una buena base de los datos mejora considerablemente el porcentaje de acierto de las redes. Esto se demostró al utilizar distintos datasets en la etapa 2 de trabajo, donde datasets que no tenían un porcentaje alto de confiabilidad, en cuanto a que no todos los comentarios presentes en una clase en particular realmente pertenencian a ella, perjudican al entrenamiento de la red, no permitiendo que esta realice buenas predicciones.

Por su lado, la red recurrente LSTM demostró ser una gran herramienta para poder categorizar texto, prácticamente independiente de la estructura que se elija, esta tendrá buenos resultados. Por lo que más que encontrar un modelo más adecuado, la forma de mejorar este sistema, es la de mejorar la calidad de los datos con los que se está trabajando. Por lo que para mejorar el sistema utilizado, se propone lo siguiente.

- Aumentar Corpus: Si bien el corpus utilizado en este estudio considera más de 3 millones de comentarios provenientes de Facebook y Twitter, este sólo pertenece a un área de trabajo de la empresa Wholemeaning, que sólo considera comentarios provenientes del área de Telecomunicaciones, es decir de todas las empresas de ese rubro, tales como WOM, VTR, ENTEL, etc. Por lo que la vectorización de palabra que resulta al utilizar este corpus, estará muy ligado a conceptos que son normalmente utilizados en esta área, tales como; Conexión, Celular, Señal, Red, etc. Por lo que si existiese palabras no tan comunes presentes en los comentarios, y que influyan en la emoción que expresa el mismo, no podrán ser tomadas realmente en consideración, dado que el vector asociada

a ella simplemente no existirá, o no estará lo suficientemente entrenado para permitir a la red realizar una buena predicción.

Por lo que es necesario no tan sólo aumentar el corpus, si no además, incluir en él áreas diferentes, para establecer mejores relaciones entre los vectores de palabras. Mejorando la calidad de los vectores, permitirá a la red poder tomar mejores decisiones.

- Aumentar Dataset: El mejor resultado de entrenamiento fue logrado por medio de un dataset construido con el único propósito de entrenar los modelos, este fue cuidadosamente construido, cerciorándose que cada comentario realmente perteneciera a la clase seleccionada, es decir, no existe ningun comentario positivo o neutro, dentro de la clase negativa, esto para ayudar a la red a que aprenda realmente que es un comentario negativo. Hacer eso produjo un gran impacto en el resultado final, pero aún así es posible mejorarlo. Para ello, es necesario aumentar considerablemente el tamaño de este dataset, esto permitirá que la probabilidad de encontrar comentarios que expresen una misma idea, pero de manera distinta, aumente. Mientras mayor sea la variedad de comentarios, mejor podrá "comprender" la red de que por ejemplo, quejarse sobre un producto, puede ser expresado de muchas formas distintas, pero que realmente todas esa formas, pertenecen a una misma clase.

  En este trabajo sólo se consideró un dataset del área de Telecomunicaciones, ampliarlo con más comentarios de esta misma área, mejorará sin duda su performance, pero una posible mejora, sea la de añadir ejemplos pertenecientes a otras áreas. La inclusión de ejemplos con nuevos conceptos, nuevas palabras o frases puede ayudar a la red a no tan sólo comprender que hace que un comentario presente insatisfacción con respecto a una empresa, si no a "entender" que hace que un comentario exprese una emoción negativa. Lo que se lograría con ello, es que la red pueda trabajar bien bajo cualquier ámbito de trabajo, pero al realizar esto, e intentar expandir su área de trabajo, también podría bajar su desempeño en un área determinada, lo que en cierta medida estaría sacrificando su precisión en un ámbito específico, para poder trabajar en todos a la vez.

- Retro-alimentar la red: Cada vez que exista una predicción equivocada del modelo, el comentario asociado puede ser posteriormente almacenado, para que la próxima vez que se sobre-entrene la red, el comentario sea incluido en el dataset de entrenamiento, esto ayudará a que la red, vaya mejorando sus predicciones con el paso del tiempo. Incluso sobre-entrenar la red con comentarios que fueron bien categorizados, ayudará a que ella se pueda perfeccionar, y entregar en un futuro, resultados con un mayor orden de seguridad. Esto último implica que por ejemplo, si predice que tal comentario (positivo) tiene una probabilidad de un 60 % de ser de la clase positiva, si se entrena con dicho comentario, esta probabilidad pueda subir a un 80 %.

Bibliografía

[1] Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 79?86.

[2] Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 417-424.

[3] Mosh Koppel and Jonathan Schler. The Importance of Neutral Examples for Learning Sentiment. 2006.

[4] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. 2005

[5] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

[6] Shiyang Wen and Xiaojun Wan. Emotion Classification in Microblog Texts Using Class Sequential Rules. 2014.

[7] Walaa Medhat, Ahmed Hassan, Hoda Korashy. Sentiment analysis algorithms and applications: A survey. 2014

[8] Casey Whitelaw, Navendy Garg and Shlomo Argamon. Using Appraisal Groups for Sentiment Analysis. 2005

[9] George A. Miller, Richard Beckwith, Christiane Fellbau, Derek Gross and Katherine Miller. Introduction to WordNet: An On-line Lexical Database. 1993.

[10] Saif Mohammad, Cody Dunne and BOnnie Dorr. Generating High-Coverage Semantic Orientation LexiconsFrom Overtly Marked Words and a Thesaurus. 2009.

[11] Isa Maks and Piek Vossen. A verb lexicon model for deep sentiment analysis and opinion mining applications. 2012.

[12] Hanhoon Kang, Seong Joon Yoo .Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews. 2012.

[13] Yung-Ming Li and Tsung-Ying Li. Deriving Marketing Intelligence over Microblogs. 2011.

[14] Warren S. McCulloch and Walter H. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, Vol. 5, p.115-133, 1943.

[15] Carlos Periñán Pascual. En defensa del Procesamiento del Lenguaje Natural Fundamentado en la Lingüística Teórica. 2012.

[16] F.Rosenblatt. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. Vol. 65, No 6, 1958.

[17] Marvin Minsky and Seymour Papert. A Review of "Perceptrons: An Introduction to Computational Geometry". 1969.

[18] David E.Rumelhart, Geoffrey E. Hinton and ROnald J. Williams. Learning Representations by Back-Propagating Errors. 1986.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Suskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine LEarning Research 15, 2014.

[20] Andrew Y. Ng. Preventing .^{o}verfitting.^{o}f Cross-Validation Data.

[21] Gavin C. Cawley and Nicola L. C. Talbot. On Over-fitting in Model Selection and Subsquent Selection Bias in Performance Evaluation. 2010.

[22] Rich Caruana, Steve Lawrence and Lee Giles. Overfitting in Neural Nets. Backpropagation, Conjugate Gradient, and Early Stopping.

[23] Tim Salimans and Diederik P. Kingma . Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. 2016

[24] Derrick Nguyen and Bernard Widrow. Imrpoving the learning Speed of 2-Layer Neural NEtworks by Choosing Initial Values of the Adaptative Weights.

[25] Raja Giryes, Guillermo Sapiro and Alex M. Bronstein. Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy?. 2015.

[26] D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. 2013.

[27] R. Collobert,J. Weston. Natural Language Processing (almost) from Scratch. Marzo 2011.

[28] George A. Miller, Walter G. Charles. Contextual correlates of semantic similarity. Language and Cognitive Processes, 1991, Vol. 6, p.1-28.

[29] Peter F Brown, Robert L Mercer, Vincent J. Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural. Computational Linguistics, 18(4).

[30] Randall P. Ellis and Pooja G. Mookim. K-Fold Cross-Validation is Superior to Split Sample Validation for Risk Adjustment Models. Junio 2013.

[31] How the backpropagation algorithm works. neuralnetworksanddeeplearning.com/chap2.html. Detalle sobre el algoritmo de back-propagation en redes FeedForward.

[32] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. 2012

[33] Eric Greenstein and Daniel Penner. Japanese-to-English Machine Translation Using Recurrent Neural Networks. 2015.

[34] Pengfei Liu, Xipeng Qiu and Xuanjing Huang. Recurrent Neural Network for Text Classification with Multi-Task Learning.

[35] Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton . Speech Recognition with Deep Recurrent Neural Networks. 2013.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. Enero 2013

[37] G.E. Hinton, J.L. McClelland, D.E. Rumelhart. Distributed representations. In: Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations, MIT Press, 1986.

[38] H. Schwenk. Continuous space language models. Computer Speech and Language, vol. 21, 2007.

[39] Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[40] Razvan Pascanu, Tomas Mikolov and Yoshua Bengio. On the difficulty of training recurrent neural networks.

[41] Sepp Hochreiter and Jurgen Schmidhuber. Long-Short Term Memory

[42] Shi Yan. https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714 .

[43] Xiang Zhang and Yann LeCun. Text Understanding from Scratch. 2015

[44] Armand Joulin, Edouard Grave, Piotr Bojanowski and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. 2016.

[45] Jeffrey Pennington, Richard Socher and Christopher D. Manning. GloVe: Global Vectors for Word Representation. 2014

[46] https://radimrehurek.com/gensim/models/word2vec.html

[47] https://fasttext.cc/docs/en/support.html

[48] https://docs.microsoft.com/en-us/azure/machine-learning/data-science-virtual-machine/overview

[49] https://keras.io/

[50] https://www.tensorflow.org/

[51] Listado de frecuencias del CREA. (Corpus de Referencia del Español Actual). http://corpus.rae.es/lfrecuencias.html

[52] https://github.com/spotify/annoy

[53] Estadística de Tweets por día. http://www.internetlivestats.com/twitter-statistics/

[54] Estudio de uso de Twitter en Chile. www.analitic.cl

[55] Estudio de Zendesk Çustomer Service And Business Results: A Survaey Of Customer Service From Mid-Size Companies". 2013

[56] Dataset Twitter. http://help.sentiment140.com/for-students/

[57] Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher. Learning Word Vectors for Sentiment Analysis. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. 2011

[58] Azure Sentiment Analysis. https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/