UNIVERSIDAD DE ANTIOQUIA

Comparativa de modelos predictivos para identificación de palabras con carga emocional

Autor

Edward Nicolas Montoya Arcila

Universidad de Antioquia
Facultad de ingeniería
Medellín, Colombia
2021

Comparativa de modelos predictivos para identificación de palabras con carga emocional

Edward Nicolas Montoya Arcila

Monografía presentada como requisito parcial para obtener el título de:
Especialista en analítica y ciencia de datos

Asesor
Antonio Jesús Tamayo Herrera

Universidad de Antioquia
Facultad de ingeniería
Medellín, Colombia
2021

Comparativa de modelos predictivos para identificación de palabras con carga emocional

Edward Nicolas Montoya Arcila, Ingeniero electrónico

***Abstract*—Monografía comparativa entre *bag of words*, LSTM, y BERT, mediante un problema de identificación de palabras con carga emocional. En esta monografía se aborda un problema no trivial de una variante de análisis de sentimiento dando un recorrido histórico al estado del arte de las implementaciones más comunes de Machine y Deep Learning para la soluciones de problemas de procesamiento de lenguaje natural. El resultado es una combinación que permite identificar el avance progresivo de las técnicas de análisis de texto y sorpresas que nos recuerdan que más allá de la tecnología implementada, los mejores resultados se obtienen de las implementaciones adecuadas.**

***Index Terms*—Inteligencia, artificial, aprendizaje, comentario, tweet, sentimiento, análisis, etc.**

I. Introducción

EL análisis de sentimientos es una técnica que permite detectar el sentimiento subyacente en un fragmento de texto. Los sentimientos se suelen clasificar como: neutros, positivos y negativos. Existen diferentes maneras de analizar el texto para poder identificar el propósito o sentimiento del mismo. La aplicación más común del análisis de sentimientos consiste en identificar la carga semántica y sintáctica de una oración para, mediante un modelo de *machine learning* o *deep learning*, en otras palabras, se trata de encontrar el sentimiento que expresa dicho texto. El análisis de sentimientos es esencial para múltiples tareas empresariales y retos del nuevo siglo, ya que mediante éste, ha sido posible mejorar en gran medida la atención al cliente y la automatización de procesos donde el lenguaje humano es altamente utilizado. Estos son algunos de los sectores donde es más usado el análisis de sentimientos:

- **Sector comercial** con implementaciones de chatbots, herramientas de comunicación inteligentes y análisis de opiniones.
- **Sector bancario** con implementaciones de análisis social para dirigir campañas de consumo.
- **Sector social** con la implementación de herramientas de perfilamiento.
- **Sector logístico** con la implementación de herramientas de distribución automatizadas basadas en las interacciones humanas.

Uno de los principales medios de información para el análisis de sentimientos son las aplicaciones de *microblogging* como Twitter, Facebook e Instagram. En dichas aplicaciones las personas publican todo tipo de comentarios, recomendaciones, opiniones, charlas e incluso discusiones. Convirtiéndose en un punto critico para todo tipo de empresa, pues con el tiempo los usuarios publican información relevante para sus negocios, permitiendo identificar opiniones políticas, sociales y económicas que pueden dar herramientas de decisión para sus objetivos de mercado.

Para esta monografía se utiliza un corpus proporcionado por la competencia de *Kaggle Tweet sentiment Extraction* [1], que a su vez, fue tomado de *Figure Eight's Data for Everyone platform* [2]. Basado en dicho conjunto de datos, el objetivo de esta monografía es construir diferentes modelos que permitan obtener la palabra o frase que mejor respalda al sentimiento etiquetado para un tweet.

Las aplicaciones del análisis de sentimientos están ampliamente distribuidas por el mundo debido a su capacidad de extraer conocimientos. Los nuevos mercados se apalancan en las interacciones con el fin de entender a millones a usuarios. Sin embargo, este proceso no es sencillo, pues el lenguaje humano es complejo. Enseñar a una máquina a analizar los diversos matices gramaticales, variaciones culturales, jergas y errores ortográficos en las menciones online es un proceso difícil, aún más, si se deben analizar diferentes idiomas donde el contexto y el tono del mensaje pueden afectar todo el sentido de la oración.

El uso de modelos de lenguaje pre-entrenados ha resultado de gran utilidad para enfrentar este tipo de retos, ya sea en sus variantes de extracción de características o de puesta a punto. El enfoque de extracción de características se centra en usar arquitecturas especificas basadas en tareas que buscan adicionar características valiosas para el entrenamiento, y el enfoque de puesta a punto se centra en encontrar los parámetros idóneos para la solución de tareas genéricas. Los dos enfoques comparten el mismo objetivo, y utilizan modelos de lenguaje para aprender representaciones de lenguaje generales. Algunos ejemplos de enfoques de extracción de características son: ELMo[3], Glove[4] y FastText[5]. Y para el enfoque de puesta a punto se tienen: OpenIA[6] y BERT[7]. Para esta monografía se usan las siguientes arquitecturas:

- Bag of words
- Glove + Redes neuronales convolucionales [8]
- Glove + Redes neuronales convolucionales [8]
- Glove + biLSTM [9]
- FastText + biLSTM
- FastText + Redes neuronales recurrentes [10]
- BERT + Redes neuronales recurrentes
- DistilBERT + Question And Answer
- RoBERTa + Redes neuronales recurrentes

El propósito de elegir dichas arquitecturas es dar un recorrido histórico a los diferentes métodos que se han usado con el paso de los años para análisis de sentimientos, de modo, que sea posible identificar el rendimiento que se ha tenido con cada una de las implementaciones mencionadas. En este

compilado hay representaciones de enfoques clásicos como ***Bag of words***, uso de *embeddings* sin contexto como ***FastText*** y ***Glove***, uso de embeddings con contexto como ***BERT*** y diferentes arquitecturas de redes neuronales como: ***biLSTM*** y ***Transformer***.

II. Estado del arte

El procesamiento del lenguaje natural (NPL, por sus siglas en inglés) se refiere a la rama de la informática, y más específicamente, a la rama de la inteligencia artificial o IA, que se ocupa de dar a las computadoras la capacidad de comprender texto y palabras habladas de la misma manera que los seres humanos [11]. Desde 2012 comenzó un proceso de crecimiento tecnológico nunca antes visto en el área de Deep Learning, transformándose en la piedra angular para el desarrollo de modelos de lenguaje de alta calidad, que dieron un nuevo alcance a toda el área de procesamiento de lenguaje natural. En esta sección se incluye un recorrido histórico por los principales modelos de lenguaje más transcendentales hasta la fecha.

A. Técnicas de representación de texto en espacios vectoriales

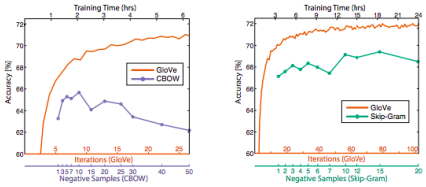
1) Bag of words: Uno de los problemas básicos de NPL es la clasificación de documentos. Un enfoque de *Machine Learning* consiste en entrenar a la computadora para crear reglas utilizando los datos proporcionados, y a su vez, un enfoque simple para la solución de este tipo de problemas es *bag of words*. El modelo de *bag of words* es uno de los métodos de representación más populares para la categorización de objetos. La idea clave es cuantificar las principales características extraídas de cada palabra para luego agruparlas basados en un conjunto de condiciones que varían según el objetivo, para este propósito, generalmente se usa un algoritmo de agrupamiento (por ejemplo, K-medias). Varios estudios han mostrado resultados alentadores de la representación *bag of words* para la categorización de objetos, los estudios teóricos sobre las propiedades del modelo *bag of words* están casi intactos, posiblemente debido a la dificultad que presenta el uso de un proceso de agrupamiento heurístico [12].

2) TF-IDF: Bag Of Words puede funcionar en algunos casos, pero presenta problemas cuando un documento es largo y el recuento de palabras puede afectar la presición del algoritmo. Para resolver este problema, se puede usar un enfoque de recuperación de información llamado TF-IDF (Término Frecuencia - Frecuencia Inversa de Documentos). Este enfoque normaliza el recuento y da como resultado palabras clave que son importantes para el documento. Estos enfoques no tienen en cuenta ninguna relación entre palabra.

3) Word Embedding: Los términos escritos se han codificado tradicionalmente como símbolos discretos que no pueden compararse directamente entre sí (a menos que consideremos la distancia de edición a nivel de carácter). Conceptualmente, esto es equivalente a una codificación *one-hot* en la que cada término se representa como un vector disperso con dimensionalidad igual al tamaño del vocabulario (cada dimensión corresponde a una palabra única). En la codificación *one-hot* para un término dado t, creamos un vector $\vec{0}$ y establecemos el índice correspondiente a t en 1. En tal codificación, todos los términos son ortogonales y equidistantes entre sí, y por lo tanto no hay una manera fácil de reconocer términos similares en el espacio vectorial. Los modelos semánticos del espacio vectorial del lenguaje representan cada palabra con un vector de valor real. Esto está basado en la semántica distribucional, una idea de Firth [13]. Estos vectores se pueden utilizar como características en una variedad de aplicaciones, como la recuperación de información [14], clasificación de documentos [15], respuesta a preguntas [16], reconocimiento de entidad nombrada [17], y análisis sintáctico [18]. La mayoría de los métodos de vectores de palabras se basan en la distancia o ángulo entre pares de vectores de palabras como método principal para evaluar la calidad intrínseca de tal conjunto de representaciones de palabras. Sin embargo con *Glove* se introduce un nuevo esquema de evaluación basado en analogías de palabras que investiga la estructura más fina del espacio vectorial, examinando no la distancia escalar entre vectores de palabras, sino más bien sus diversas dimensiones de diferencia. Este esquema de evaluación favorece los modelos que producen dimensiones de significado, capturando así la idea de agrupamiento múltiple de representaciones distribuidas. *Word Embedding* en general se considera como una de las representaciones más populares para de vocabularios. Es capaz de capturar el contexto de una palabra en un documento, similitud semántica y sintáctica, además de la relación con otras palabras. En este modelo se tiene una lista de valores que representan una palabra. Cada número de la lista significa una medida en una dimensión diferente o un tema diferente. *Word2vec* es un ejemplo que toma una palabra de una oración, luego busca palabras cercanas de la misma oración y se ejecuta a través de una red neuronal para producir los valores de cada categoría para una lista de palabras. En la actualidad uno de los modelos de *embeddings* más utilizados es *word2vec*, un modelo que utiliza una red neuronal de tres capas con una capa oculta para aprender las representaciones de palabras. Se propusieron dos variantes: *skip-gram* predice el contexto circundante dada la palabra actual, mientras que *continues bag of words* predice la palabra actual dado su contexto. Los autores entrenaron *word2vec* en un gran corpus de noticias, y publicaron tanto el software como los *embeddings* resultantes en año 2013 [19].

4) Glove: En 2014, se presento a la comunidad otro enfoque para generar *word embeddings* llamado **GloVe** (Vectores Globales para Representación de Palabras) [4]. En este enfoque se agrega la matriz global de co-ocurrencia palabra-palabra de un corpus para producir word embeddings. El enfoque de **GloVe** es definir el contexto para una palabra i y una palabra j basados en la aparición de estás palabras y en su proximidad a otro conjunto de N paralabras. El vector de codificación contiene la relación de las probabilidades de coexistencia de dos palabras conocidas explícitamente como método basado en recuento. En contraste con el enfoque *skip-gram*, el vector de codificación contiene las probabilidades de su co-ocurrencia que predice el modelo conocido como método basado en predicciones. El método basado en predicciones ha demostrado ser más eficiente, pero los autores de GloVe argumentan que el enfoque basado en conteo captura

la estadística global y puede ser más ventajoso ya que supera el aprendizaje de la representación de palabras en analogía de palabras, similitud de palabras y tareas de NER. En la Figura 1 se muestra el gráfico de la precisión de los enfoques *GloVe* vs *word2vec* utilizando un tamaño de vector de 300 con un tamaño de vocabulario de 400,000 y una ventana de tamaño 10 (número de palabras circundantes) en un corpus de 6 mil millones de tokens.



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Fig. 1: Glove: Global Vectors for Word Representation [4]

B. Arquitecturas de modelos predictivos

1) Redes neuronales recurrentes (RNN): Una red neuronal recurrente (RNN) es un tipo de red neuronal artificial que utiliza datos secuenciales o datos de series de tiempo. Estos algoritmos de aprendizaje profundo se utilizan comúnmente para problemas ordinales o temporales, como la traducción de idiomas, el reconocimiento de voz y los subtítulos de imágenes. Se distinguen por su ”memoria”, ya que toman información de entradas anteriores para influir en la entrada y salida actuales. Mientras que las redes neuronales profundas tradicionales asumen que las entradas y salidas son independientes entre sí, la salida de las redes neuronales recurrentes depende de los elementos anteriores dentro de la secuencia. Si bien los eventos futuros también serían útiles para determinar la salida de una secuencia dada, las redes neuronales recurrentes unidireccionales no pueden tener en cuenta estos eventos en sus predicciones.

2) Long short-term memory (LSTM): Esta es una arquitectura RNN muy popular, fue introducida por Sepp Hochreiter y Juergen Schmidhuber como una solución al problema del *vanishing gradient*. Su investigación [20] abordó el problema de las dependencias a largo plazo. Es decir, si el estado anterior que influye en la predicción actual no es del pasado reciente, es posible que el modelo RNN no pueda predecir con precisión el estado actual. Para remediar esto, las LSTM tienen ”celdas” en las capas ocultas de la red neuronal, que tienen tres compuertas: una compuertas de entrada, una compuertas de salida y una compuertas de olvido como lo muestra la Figura 2. Estas compuertas controlan el flujo de información que se necesita para predecir la salida en la red.

3) ELMo: ELMo es una representación contextualizada profunda de palabras, es capaz de modelar caracteristicas complejas (como sintaxis y semántica), además es capaz de identificar las variaciones linguisticas que puede tener una palabra según en el contexto en el cual sea usada. Al igual que los anteriores casos, ELMo se usa como modelo de lenguaje por su excelente caracterización de secuencias de texto complejas, con la particularidad de ser capaz de generar grandes mejoras al ser usado en una amplia gama

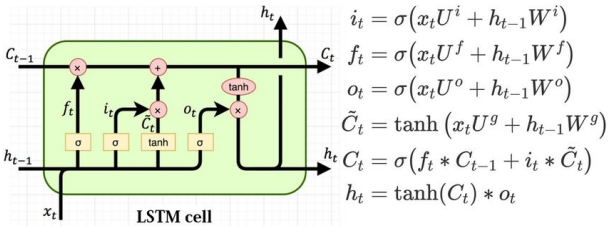


Fig. 2: Configuración interna de una LSTM[21]

de problemas desafiantes de NPL, incluida la respuesta a preguntas, la vinculación textual y el análisis de sentimientos [3]. ELMo se caracteriza por representar cada palabra según el **contexto** en el cual se utiliza (ejm. célula biológica y célula terrorista, tienen diferentes representaciones), ELMo es profundo pues combina todas las capaz de entrenamiento de una red neuronal profunda y finalmente las representaciones de ELMo se basan puramente en caracteres, lo que permite a la red utilizar pistas morfológicas para formar representaciones robustas para tokens fuera del vocabulario que no se ven en el entrenamiento.

En la Tabla I se puede visualizar las mejoras obtenidas mediante los embeddings de ELMo, para algunas de las principales tareas de NLP.

Task	Previous OTA		Traditional Baseline	ELMo baseline	Increase
SQuAD	SAN	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al (2017)	88.6	88.0	88.7 +/- 0.17	0.7 / 5.8%
SRL	He et al (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee ar al (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al (2017)	91.93 +/- 0.19	90.15	99.22 +/- 0.10	2.06 / 21%
Sentiment (5-Class)	McCann et al (2017)	53.7	51.4	54.7 +/- 0.5	3.3 / 6.8%

TABLE I: Mejoras en problemas de NPL

4) Transformer: Las desventajas computacionales provenientes de las redes neuronales recurrentes dieron como resultado en diciembre de 2017 el modelo de arquitectura ***Transformer***, un modelo basado en técnicas de atención (técnicas que permiten modelar las dependencias sin tener en cuenta su distancia entre las secuencias de entrada o salida), mediante el uso de la arquitectura *encoder - decoder*. donde busca codificar información en su vector sobre el contexto relevante de una palabra determinada. En general, el codificador y el decodificador son excelentes para problemas *seq2seq*, donde se mapea una secuencia de entrada a una secuencia de salida. El codificador en el transformador tiene el mecanismo de auto atención para facilitar la conexión residual a la siguiente etapa, el decodificador. La autoatención permite al modelo asociar la palabra a su referencia. Utiliza almacenes de clave-valor donde se puede recuperar consultando la información de su memoria. Auto-atención implica que en cada paso, a medida que se genera el siguiente token, solo hace uso del token anterior. El decodificador también realiza *multi-head attention* sobre la salida de la pila de codificadores. La atención de varios encabezados indica que una búsqueda de consulta de una palabra tiene varios significados, lo que da como resultado más de un valor-clave por consulta.

El ***Transformer*** mejora significativamente las tareas de traducción con su estructura codificador-decodificador. Puede lidiar con las dependencias a largo plazo mejor que las redes neurales recurrentes LSTM. En la Figura 3 se puede visualizar el esquema de la arquitectura.

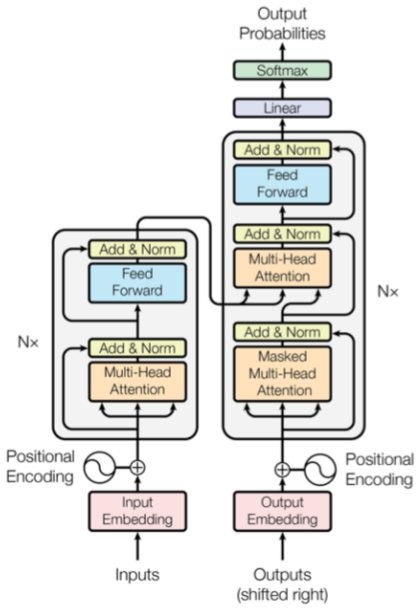


Fig. 3: Modelo de arquitectura Transformer [22]

La arquitectura ***transformer***, es el primer modelo de transducción de secuencia basado enteramente en tensores, reemplazando las capas recurrentes más comúnmente utilizadas en arquitecturas de codificador-decodificador con técnicas de auto-atención. Finalmente, las ventajas de transformer sobre las redes neuronales LSTM son:

- Capturan mayor dependencias entre secuencias que las LSTM.
- Pueden ser usadas por hardware de procesamiento en paralelo como las TPU. Por lo tanto, el aprendizaje es mucho más rápido, sin propagación hacia atrás en el tiempo. El entrenamiento que podría llevar días en tareas de NPL con LSTM se puede realizar en horas con un modelo de solo atención.

5) BERT: BERT es una arquitectura de modelo de lenguaje, que usa el *transformer* como base. En su implementación inicial los tranformer usan dos mecanismos separados de encoder y decoder para tratar de aprender la relación entre las palabras. En BERT se usa solo el encoder porque su obejetivo es ser un modelo de lenguaje. La innovación técnica clave de BERT es la aplicación de la formación bidireccional de Transformer, al modelado del lenguaje. Esto contrasta con los esfuerzos anteriores en los que se analizaba una secuencia de texto de izquierda a derecha o una formación combinada de izquierda a derecha y de derecha a izquierda. Los resultados del artículo muestran que un modelo de lenguaje que se entrena bidireccionalmente puede tener un sentido más profundo del contexto y flujo del lenguaje que los modelos de lenguaje unidireccionales.

C. Trabajos relacionados

Implementaciones similares no son muy comunes en este campo, la identificación de palabras normalmente es un proceso intermedio de otro tipo de análisis de clasificación. Sin embargo, la calidad con la cual se realice y la flexibilidad del modelo para reconocer palabras posiblemente significativas son el pilar funcional de lo que denominamos *sentiment analisys*. En el proceso de mejorar los modelos de análisis de sentimientos algunas investigaciones similares son:

1) Identificación de sentimientos consciente, Yushi Yao, Guangjian Li, 2016: El análisis de sentimientos tradicional a menudo utiliza el diccionario de opiniones para extraer información de opiniones en texto y clasificar documentos. Sin embargo, las palabras y frases informales resultantes en el contenido generado por el usuario requieren un análisis del contexto. Por lo general, una palabra tiene significados especiales en un contexto particular, lo que a su vez modifica la carga emocional que transmite. El modelo de lenguaje distribuido word2vec, en dicho artículo representa un método novedoso de identificación de palabras basado la representación de sentimientos en un contexto particular. El resultado de esta investigación muestra que el modelo final tiene un mejor rendimiento en la representación de las palabras con un significado especial, mientras sigue funcionando bien en la representación de patrones idiomáticos [23].

2) Identificación de palabras de sentimiento mediante un modelo de optimización con regularización L1, Zhi-Hong Deng, Hongliang Yu, Yunlun Yang, 2016: La identificación de palabras de sentimiento es un trabajo fundamental en numerosas aplicaciones de análisis de sentimientos como la minería de opiniones, la búsqueda de titulares y la clasificación de textos. En este artículo se propone un modelo de optimización con regularización L1, denominado ISOMER, capaz de identificar palabras con carga emocional de un corpus. El modelo puede emplear tanto palabras semilla como documentos con etiquetas de sentimiento, diferente de la mayoría de las investigaciones existentes que adoptan solo palabras semilla. La penalización L1 en la función objetivo produce una solución ya que la mayoría de las palabras candidatas no tienen sentimiento. Los experimentos en los conjuntos de datos reales muestran que ISOMER supera a los enfoques clásicos y que el léxico aprendido por ISOMER se puede adaptarse eficazmente al análisis de sentimientos a nivel de documento [24].

3) Identificación de palabras de sentimiento con factores contextuales de sentimiento, Liang, Jiguang, Zhou, Xiaofei, 2015: El libro *Identificación de palabras de sentimiento con factores contextuales de sentimiento* es uno de los principales referentes de la literatura de identificación de palabras con carga emocional. En el libro, la identificación de palabras con carga emocional se considera como un problema de factorización matricial y se proponen diferentes modelos para su solución. En lugar de palabras clave, se aprovecha la coincidencia de sentimientos y la coherencia de los mismos para modelar. Amplios estudios experimentales en tres conjuntos de datos del mundo real demuestran que los modelos propuestos superan muchos de los enfoques de vanguardia [25].

III. METODOLOGÍA

En esta sección se describe el proceso planteado para cada uno de los enfoques o arquitecturas usados en la solución del

problema de identificación de palabras con carga emocional. El problema consiste encontrar las palabras que contienen la carga emocional de un *Tweet*, a partir del texto y una etiqueta que indica el sentimiento del mismo. Para esta monografía se usa un conjunto de datos titulado ***Análisis de sentimiento: tweets de emoción en texto con etiquetas de sentimiento existentes***, dicho conjunto de datos está bajo la licencia internacional 4.0 de creative commons.
El proceso de implementación de cada una de las arquitecturas propuestas se realizó mediante un flujo estandarizado de procesamiento de datos para Machine/Deep Learning. En la Figura 4 se puede visualizar el proceso.

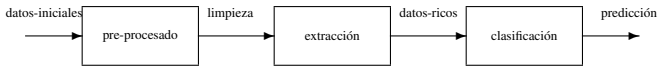


Fig. 4: Procesamiento de datos

A. Configuración de ambiente

El conjunto de datos elegido para este proyecto es una base de datos de Tweets artificial en inglés. Ésta se encuentra en un archivo separado por comas (CSV) y contiene las siguientes columnas: Identificador único por tweet (textId), Texto del tweet (text), Sentimiento del tweet (sentiment) y texto que soporta el sentimiento (selected_text). Un ejemplo del corpus del conjunto de datos se puede visualizar en la Tabla II

text	sentiment	selected_text
I'd have responded	neutral	I'd have responded
Sooo SAD I will miss you	negative	Sooo SAD
the baby are fun	positive	fun

TABLE II: Fragmento del conjunto de datos

La implementación práctica de la monografía usa Python como lenguaje de programación. Y para las arquitecturas de Deep Leraning, manipular, limpiar y procesar los datos se usan las siguientes librerias:

- **Numpy:** Es el paquete fundamental para la computación científica en Python.
- **Pandas:** Herramienta OpenSource para análisis y manipulación de datos.
- **matplotlib:** Librería para crear visualizaciones estáticas, animadas e interactivas en Python.
- **seaborn:** Librería para crear visualizaciones basada en Matplotlib.
- **plotly:** Librería para crear visualizaciones que facilita la manipulación de nubes de palabras.
- **nltk:** Librería de lenguage natural en Python
- **tensorflow:** Es un *framework* de código abierto para el aprendizaje automático y cálculos sobre datos descentralizados.
- **Keras:** Es una API diseñada para seres humanos, no para máquinas. Keras sigue las mejores prácticas para reducir la carga cognitiva.

B. Preprocesamiento

El procesado de los datos o limpieza se realiza con el objetivo de generar datos sin ruido que puedan alterar el comportamiento del algoritmo final. Para este caso, algunos de los principales factores de ruido son: Uso de enlaces a contenido multimedia, signos de puntuación, uso de mayúsculas, emoticones y elementos que no pertenecen a la oración. El preprocesamiento de los datos es una parte esencial pues debe garantizar una mínima calidad en los datos de entrada para que el modelo pueda interpretar correctamente la información que será procesada y pueda generar predicciones de valor. El código fuente usado para limpiar el texto de los Tweets, se encuentra disponible en el repositorio de GitHub del presente proyecto[1].

C. Análisis exploratorio

El análisis exploratorio o EDA por sus siglas en inglés, se refiere al proceso crítico de descubriendo de los datos. Implica realizar una investigación exhaustiva de la naturaleza de los mismos con el fin de identificar patrones y anomalías. Además de probar hipótesis y comprobar supuestos que permitan tener un resumen idóneo con estadísticas y gratificas que faciliten el entendimiento del conjunto. Para comenzar con un análisis exploratorio es necesario tener claro los siguientes elementos: **conjunto de datos**, **proceso de limpieza** y **métrica de evaluación**.

Para los tweets, el principio del análisis exploratorio consiste en identificar la distribución de los sentimientos, de manera que se puedan distribuir en tres grupos fundamentales: neutros, positivos y negativos. Posteriormente se procede a obtener el valor de la métrica de evaluación que para este caso es el índice de jaccard. Se eligió dicha métrica porque permite identificar el nivel de similitud o diversidad de dos conjuntos de datos, lo que para el caso de los tweets es el objetivo principal. Los conjuntos serán respectivamente el texto del tweet y el texto seleccionado. Otras variables relevantes de una primera aproximación del EDA son: número de palabras en el texto, número de palabras en el texto seleccionado y diferencia de palabras. Con una primera aproximación a la estructura de los tweets, comienza un análisis mas detallado, identificando la distribución de los tweets basado en el número de palabras junto con una aproximación Kernel de las mismas. Finalmente, se procede a realizar un conteo de palabras por grupo, de tal forma que sea posible identificar las palabras más usadas por sentimiento. Es de resaltar que se deben eliminar las palabras sin contenido semántico ***STOP WORDS***.

[1] https://github.com/NicolasMontoya/monografia

D. Arquitecturas propuestas

Una vez se finaliza el análisis exploratorio de los datos, se da paso al diseño de arquitecturas. El objetivo del diseño de arquitecturas es realizar un repaso histórico de los diferentes métodos que se han usado para abordar el problema de análisis de lenguaje natural y comparar los diferentes resultados base de cada una de las arquitecturas ante un problema que combina análisis de sentimientos e identificación de palabras. Es de resaltar que este trabajo no buscar entrar en detalle de ajustes finos para cada una de las arquitecturas, ni tampoco implementar versiones de alto costo computacional de cada una de ellas. Por el contrario, el análisis se centra en la identificación de palabras que expresan un sentimiento para cada una de las arquitecturas, con una implementación estándar de su tecnología.

E. Bag of Words + carga emocional

La primer arquitectura propuesta es uno de los métodos clásicos de procesamiento de lenguaje natural, su implementación toma relevancia en el inicio del *Machine Learning* y tiene cientos de implementaciones posibles. El algoritmo usado para analizar el conjunto de datos mediante bag of words fue tomado de Kaggle [26]. Su implementación se divide en dos algoritmos, uno encargado de obtener los pesos de cada palabra y otro encargado de identificar el texto seleccionado basado en la unión de palabras con mayor carga sentimental:

La descripción del algoritmo para obtener los diccionarios de pesos es la siguiente:

1) Encontrar todas las palabras i en los Tweets pertenecientes a la clase j, donde j es cada uno de los posibles sentimientos (positivo, negativo y neutro).
2) Calcular el número de tweets denominado n_{ij} en la clase j que contienen la palabra i.
3) Siendo dj el número de tweets de la clase j, calcular la proporción de tweets en la clase j que contienen la palabra i.
$$n_{ij} = \frac{n_{ij}}{d_j} \tag{1}$$
4) Se calculan los pesos asignados a cada palabra dentro de cada clase.
$$w_{ij} = p_{ij} - \sum_{k \neq j} p_{ik} \tag{2}$$

La descripción del algoritmo de selección del texto es la siguiente:

1) Identificar el sentimiento del Tweet
2) Si el sentimiento es neutral retornar todo el texto. NOTA: La razón de este paso será explicada en los resultados obtenidos en el análisis exploratorio.
3) Se calcula la sumatoria de carga emocional según el sentimiento del Tweet mediante el diccionario de pesos.
4) Se retorna el subset con mayor carga emocional.

En resumen, este método propone obtener diccionarios que relacionen una palabra con un ”nivel” de carga emocional según la ocurrencia de una palabra en los Tweets de un sentimiento. Posteriormente dividir cada una de frases en combinaciones de n-palabras consecutivas, tantas como combinaciones se puedan construir con la frase. Un ejemplo de esto para la frase ”La vida es bella” sería: un primer conjunto estaría conformado por ”La”, el segundo sería ”La vida”, el tercero sería ”La vida es” y finalmente el último sería ”La vida es bella”. Como último paso se calcula la carga emocional general para cada uno de los conjuntos de palabras donde se define el texto seleccionado como aquel con mayor carga emocional de un sentimiento particular.

F. Embeddings (Glove/FastText) + Red neuronal convolucional (1D)

La arquitectura presentada en la Figura 5 es una combinación de embeddings con una capa convolucional de una dimensión, en esta arquitectura el tweet y su sentimiento se dan como entrada, y el modelo devuelve el índice de inicio y fin del texto seleccionado. Es de resaltar que se usaron dos tipos diferentes de embeddings para identificar posibles variaciones semánticas a favor de uno u otro. Los embeddings usados fueron Glove y FastText, ninguno fue reentrenado por la red pues la capacidad de procesamiento disponible era superada por los aproximadamente 10 millones de parámetros que resultan de usar la capa de embeddings. Las capas de la arquitectura son:

1) **Capa de entrada:** El sentimiento y el tweet se tokenizan para alimentar como entrada. El percentil 99,9 de los tweets tenía menos de 32 tokens. Por lo tanto, sequenc_length se eligió como 33 (uno extra para el sentimiento).
2) **Capa de embeddings:** Esta capa tiene los vectores (Glove, FastText) previamente entrenados de dimensión 100 para Glove y dimensión 300 para FastText, que convierten cada token en un vector de su correspondiente dimensión.
3) **Capa conv1d:** 6 capas convolucionales de una dimensión con tamaño de kernel 2 y stride = 1.
4) **Capa aleatoriedad:** Evita el sobreajuste mediante la inyección de aleatoriedad del 20 por ciento a la salida del sistema. Es de resaltar que para redimensionar correctamente los datos después de pasar por la capa convolucional es necesario usar la utilidad Flatten.
5) **Capa de salida:** Dos capas densas de 32 neuronas con función de activación soft-max. Una encargada de predecir el índice de inicio y otra encargada de predecir el índice final del texto seleccionado.

La función de pérdida utilizada fue CategoricalCrossentropy y como optimizador se usa Adam con tasa de aprendizaje predeterminada. La implementación detallada puede ser consultada en Kaggle [27] [28]. Este modelo sirve como una implementación inicial para tener un punto de referencia sobre el rendimiento en aplicaciones de Deep Learning para este tipo de problemas.

G. Embeddings (Glove/FastText) + BiLSTM

La arquitectura presentada en la Figura 6 es una combinación de embeddings con redes neurales recurrentes del tipo

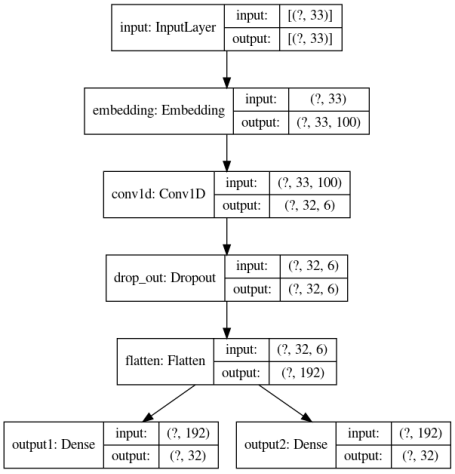


Fig. 5: Embeddings + Red neuronal convolucional (1D)

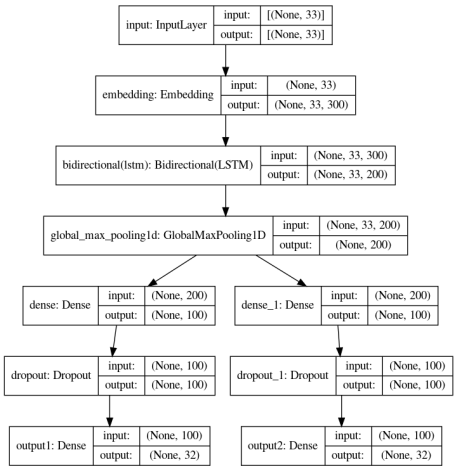


Fig. 6: Embeddings + LSTM

BiLSTM. Al igual que en la anterior arquitectura se usan los embeddings Glove y FastText sin reentrenar. Las capas de la arquitectura son:

1) **Capa de entrada:** El sentimiento y el tweet se tokenizan para alimentar como entrada. El percentil 99,9 de los tweets tenía menos de 32 tokens. Por lo tanto, sequenc_length se eligió como 33 (uno extra para el sentimiento).
2) **Capa de embeddings:** Esta capa tiene los vectores (Glove, FastText) previamente entrenados de dimensión 100 que convierten cada token en un vector de dimensión 100.
3) **Capa Bireccional con LSTM:** Capa con red neural recurrente LSTM con propagación bidireccional. Esta diseñada con 100 neuronas, retorno de secuencias y dropout del 20 por ciento.
4) **Capa post-procesado:** Se utiliza MaxPooling con el objetivo de obtener los valores más representativos en la red. Su inclusión se debe principalmente a un proceso heurístico de ensayo error.
5) **Capa densa:** Reduce el tamaño de la salida y genera un primer intento en ajustar de manera precisa los resultados finales. Usa como función de activación relu.
6) **Capa aleatoriedad:** Evita el sobreajuste mediante la inyección de aleatoriedad del 20 por ciento a la salida del sistema.
7) **Capa de salida:** Dos capas densas de 32 neuronas con función de activación soft-max. Una encargada de predecir el índice de inicio y otra encargada de predecir el índice final del texto seleccionado.

La función de pérdida utilizada fue CategoricalCrossentropy y como optimizador se usa Adam con tasa de aprendizaje predeterminada. La implementación detallada puede ser consultada en [29] [30]. Este modelo sirve como una aproximación más precisa a los temas más modernos que se usan para enfrentar los problemas de *natural languaje processing*.

H. DistilBERT enfoque Question and Answer

Las arquitecturas basadas en BERT transformaron el mundo del procesamiento de lenguaje natural al proporcionar herramientas de aprendizaje por transferencia. Técnicas como Question And Answers (seq2seq) se han visto muy favorecidas y tienden a tener buenos resultados en problemas de selección, por lo que resulta de utilidad como punto de comparación en esta monografía. En este caso la arquitectura implementa la versión proveída por la librería simpletransformers de Question And Answer [31], junto con un modelo de la implementación DistilBERT preentrenado con el dataset SQuAD [32]. Se elige DistilBERT como variación de BERT porque DistilBERT es un modelo pequeño, rápido, económico y ligero entrenado mediante destilación de la base BERT. Tiene un 40% menos de parámetros que bert-base-uncased, funciona un 60% más rápido y conserva más del 95% del rendimiento de BERT según lo medido en el punto de referencia de comprensión del lenguaje GLUE. En la Figura 7 se puede visualizar una aproximación propuesta por Vivekjadhavr [33], un competidor de la competencia de Kaggle para el enfoque QA. Cabe resaltar que la librería simpletransformers debería tener una implementación similar.

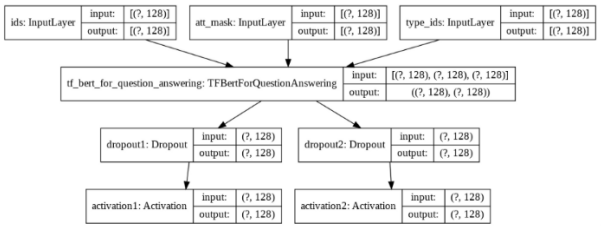


Fig. 7: BERT QA model architecture

La implementación detallada puede ser consultada en Kaggle [34].

I. RoBERTa enfoque Seq2Label

Roberta se basa en el modelo de BERT. Esta variación de BERT se entreno con conjuntos de datos más grandes, secuencias más largas y mini lotes (batch) más grandes. Se entrenó solo en el modelo de lenguaje enmascarado sin la predicción de la siguiente oración. Debido a todos estos factores, el modelo de Roberta supera al modelo de BERT, pero implica un

mayor gasto computacional. En la Figura 8 se puede visualizar el esquema de la arquitectura de RoBERTa implementada en TensorFlow. Las capas que forman la arquitectura son:

1) **Capa de entrada:** Como entrada al sistema se usan tres vectores. El primero denominado input_ids el cual contiene el texto del tweet y el sentimiento que lo identifica concatenados mediante tokens que indican el inicio y finalización de cada elemento. El segundo elemento de entrada es la mascara de atención, esta capa evita el mecanismo de atención en elementos sin importancia (da prioridad a la entrada y evita la atención en los tokens de apoyo). El elemento final para este caso no se usa pues el enfoque definido es seq2label, en caso de implementar un enfoque de Question And Answers el tercer vector define el tipo de token según su posición.
2) **Capa roBERTa:** Aquí se encuentra el modelo de lenguaje.
3) **Capa aleatoriedad:** Evita el sobreajuste mediante la inyección de aleatoriedad del 20 por ciento a la salida del sistema.
4) **Capa conv1d:** 768 capas convolucionales de una dimensión con tamaño de kernel 2 y stride = 1.
5) **Capa aleatoriedad:** Evita el sobreajuste mediante la inyección de aleatoriedad del 20 por ciento a la salida del sistema.
6) **Capa de salida:** Capas densas de 128 neuronas con función de activación softmax. Una encargada de predecir el índice de inicio y otra encargada de predecir el índice final del texto seleccionado.

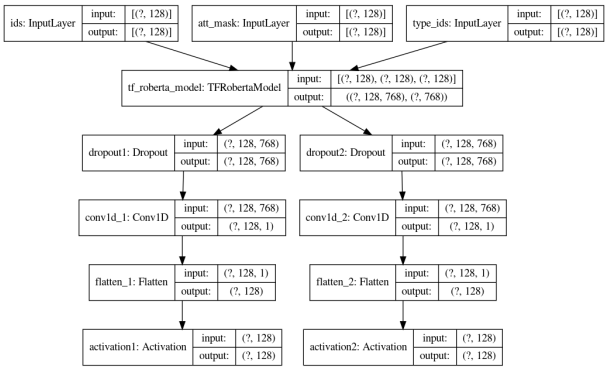


Fig. 8: Arquitectura RoBERTa tensorflow

La implementación detallada puede ser consultada en Kaggle [35].

IV. Resultados y análisis

Los primeros resultados relevantes provienen del análisis exploratorio del conjunto de datos. En la Figura 9 se puede apreciar la distribución de los tweets según su sentimiento. La figura refleja una distribución variada con una superioridad de los datos de sentimiento neutral.

A. *Análisis exploratorio*

En la Figura 10 se puede apreciar con mayor claridad la distribución porcentual de los datos de entrenamiento.

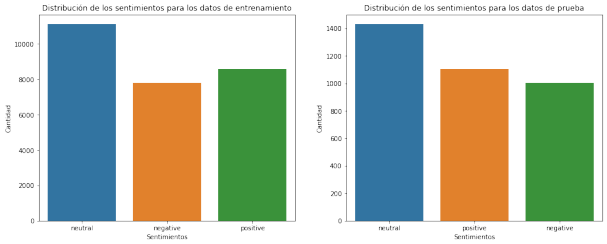


Fig. 9: Distribución de los Tweets según sentimientos

La distribución des-balanceada de los datos indica que son propensos a sesgos por lo que es útil implementar técnicas de estratificación que mantengan lo más equilibrado posible el entrenamiento del modelo.

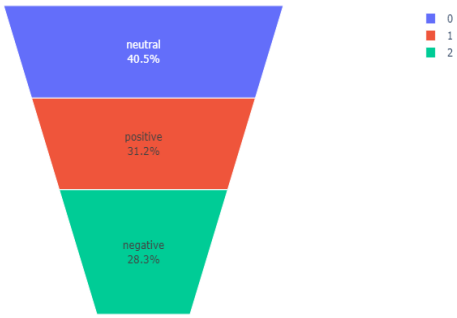


Fig. 10: Distribución porcentual de los Tweets según sentimiento

El proceso de ingeniería de características para los datos implica extraer la cantidad de palabras por tweet, la cantidad de palabras seleccionadas, la diferencia entre la cantidad de palabras totales y la cantidad de palabras seleccionadas, y finalmente el índice de Jaccard para cada uno de los registros. En las Tablas III, IV, V y VI se aprecia el resumen de las características obtenidas de las meta-variables distribuidas según su sentimiento.

Se puede concluir que la distribución para los tweets negativos y positivos es similar. Además el texto seleccionado en los tweets neutrales es considerablemente más grande, que en los demás casos. En la Tabla III se puede apreciar que la media del índice de Jaccard para los Tweets neutrales es muy cercana a uno, lo que indica que en la mayoría de los casos el texto seleccionado, es el mismo texto del Tweet.

Sentimiento	Media	50%	75%
Negativo	0.338613	0.2	0.5
Neutral	0.976445	1	1
Positivo	0.314372	0.1666	0.4285

TABLE III: Descripción del indice de Jaccard según sentimiento

En los sentimientos positivos y negativos, la distribución kernel del indice de Jaccard muestra dos núcleos principales de concentración de la palabras. El nucleo cercano a 1, nos

Sentimiento	Media	50%	75%
Negativo	3.957975	2	5
Neutral	12.069533	11	17
Positivo	3.519343	2	4

TABLE IV: Descripción del número total de palabras seleccionadas según sentimiento

Sentimiento	Media	50%	75%
Negativo	13.473204	13	19
Neutral	12.343888	11	18
Positivo	13.109881	12	18

TABLE V: Descripción del número total de palabras según sentimiento

da indicios que los tweets con muy pocas palabras siempre tienen el mismo número de palabras seleccionadas. Lo cual resulta bastante útil para la clasificación.

Es interesante analizar la aproximación kernel del índice de Jaccard y de la diferencia de palabras. Pues, junto con los datos obtenidos anteriormente de la distribución de palabras seleccionadas y la distribución de palabras totales, nos pueden dar una idea del comportamiento del indice de Jaccard esperado (métrica de validación) según el sentimiento.

B. Análisis de palabras por sentimiento

El proceso de análisis de sentimiento por palabra se realiza mediante un conteo general en los diferentes Tweets del corpus. La primera aproximación se realiza sin eliminar las palabras sin contenido semántico, en la Tabla VII se puede visualizar el resultado de dicho conteo. Tal como se podría esperar las palabras con mayor uso en el corpus corresponden a conectores y preposiciones. En la tabla VIII se puede visualizar el resultado final del conteo de palabras en el corpus después de la eliminación de las palabras sin contenido semántico. Es de resaltar que las palabras obtenidas son muy variadas y dan indicios de un corpus rico en vocabulario, es decir, mayor complejidad para el modelo. Mientras más diverso sea el compendio de palabras con valor semántico, más complejo para el modelo es poder encontrar todas las posibles combinaciones de contexto que le dan carga emocional a una palabra. Sin embargo, identificar la complejidad de las palabras de una manera tan general no útil dado que no existe una estructuraciones de los datos que nos permita tener ventajas a la hora de clasificar.

Para comprender a profundidad la naturaleza de la distribución de palabras referente al sentimiento que representan, es necesario dividir los Tweets del corpus según su sentimiento y posteriormente realizar un nuevo conteo de palabras. Resulta interesante que algunas palabras se repiten entre las Tablas X XI y IX. Este comportamiento a simple vista puede parecer un error, sin embargo, solo es una muestra de lo nutrido que se encuentra el conjunto de datos. El objetivo final de esta sección toma sentido cuando descubrimos que las palabras que convencionalmente entendemos como buenas también son las palabras que más se usan en los Tweet positivos y viceversa. Finalmente es posible llegar a la conclusión que mientras el texto ingresado no tenga elementos de muy contradictorios,

Sentimiento	Media	50%	75%
Negativo	9.515229	4	15
Neutral	0.274355	0	0
Positivo	9.590538	4	15

TABLE VI: Descripción de la diferencia entra el número total de palabras y el número total de palabras seleccionadas según el sentimiento

el modelo puede aprovechar la estructura semántica del texto para dar un resultado relativamente acertado.

	palabra	contador
0	i	7200
1	to	5305
2	the	4590
3	a	3538
4	my	2783

TABLE VII: Tabla de conteo de palabras

	palabra	contador
0	good	1251
1	day	1058
2	love	909
3	happy	852
4	like	774

TABLE VIII: Tabla de conteo de palabras después de limpieza de palabras sin contenido semántico

	Palabra	Contador
0	miss	358
1	sad	343
2	sorry	300
3	bad	246

TABLE IX: Tabla de conteo de palabras con sentimiento negativo

C. Resultados EDA

Es posible visualizar que las palabras seleccionadas para sustentar el sentimiento, siempre se encuentran dentro del texto principal. En las palabras seleccionadas con sentimiento neutro existe una tendencia clara a que la cantidad de palabras seleccionadas tengan un tamaño igual al tamaño del texto principal. También es de resaltar que las palabras seleccionadas siempre son menores o iguales en cantidad a las palabras que contiene el texto principal del tweet. La distribución de la cantidad de palabras de los tweet con sentimiento positivo y negativo no son fácilmente segmentables. Dichas distribuciones se concentran principalmente en dos núcleos, aquellos registros que contiene todas las palabras del texto y los registros que contienen aproximadamente una cuarta parte de las palabras del texto.

D. Rendimiento de los modelos

El rendimiento obtenido por cada uno de los modelos se calculo mediante la expresión de la Figura 11.

En resumen dicha expresión matemática busca recorrer cada uno de los Tweets para obtener el promedio del índice de Jaccard entre el texto total y el texto seleccionado.

	Palabra	Contador
0	good	826
1	happy	730
2	love	697
3	day	456

TABLE X: Tabla de conteo de palabras con sentimiento positivo

	Palabra	Contador
0	get	612
1	go	569
2	day	492
3	dont	482

TABLE XI: Tabla de conteo de palabras con sentimiento neutral

Los elementos recopilados en la Tabla XII son el resultado de la métrica de evaluación por cada una de las arquitecturas propuestas: *bag of words*, Red convolucional1D + *Glove*, Red convolucional1D + *Embeddings*, LSTM + *Glove*, LSTM + *Embeddings*, DistilBERT y RoBERTa. El primer resultado relevante consiste en el excelente desempeño que posee bag of words en rendimiento/velocidad respecto a los demás modelos, superando por un 6% tanto a las redes neuronales convolucionales como a las redes neuronales recurrentes. Este comportamiento posiblemente se deba a la falta de ajuste fino de las redes neuronales. Sin embargo, podemos concluir que con un uso adecuado de técnicas de clásicas es posible desarrollar modelos competentes con mucho menos consumo de recursos computacionales.
Los mejores resultados en la comparativa son los obtenidos en la arquitectura BERT, tanto en su implementación de *question and answer*, como en su implementación de seq2label. El resultado obtenido de la comparativa da una idea del rendimiento que pueden alcanzar implementaciones de técnicas de atención en problemas de procesamiento de lenguaje natural. Sin embargo, también es visible que presenta falencias respecto a la cantidad de recursos computacionales necesarios para generar un modelo. Por lo cual en problemas donde los recursos computaciones son limitados las técnicas clásicas podrían ser una mejor alternativa dado su relación rendimiento/costo. Para el caso del problema presentado en este artículo la diferencia entre la métrica de BERT y la métrica de *bag of words* es de aproximadamente del 4 %.

Modelo	Resultado
RoBERTa	0.7079
DistilBERT	0.6951
Bag of words	0.6682
Conv1D + FastText	0.6182
BiLSTM+ FastText	0.6061
BiLSTM+ Glove	0.5964
Conv1D + Glove	0.5890

TABLE XII: Tabla de resultados índice de Jacard por modelo

Para finalizar se concluye que el avance histórico de las técnicas de procesamiento de lenguaje natural ha sido considerable conforme pasan los años y su resultado puede apreciarse en BERT, una de las técnicas más eficientes a la hora de enfrentar problemas que impliquen procesamiento de texto.

$$\frac{1}{n}\sum_{i=1}^{n} jaccard(gt_i dt_i)$$

Fig. 11: Expresión matemática de evaluación

Sin embargo, los resultados de la monografía también dejan entrever que el avance de la tecnología no exime de realizar análisis profundos a la naturaleza de los datos de un problema, pues la solución siempre pasa por encontrar la técnica correcta para un problema concreto.

V. Conclusiones

En el desarrollo de la monografía se logró encontrar un conjunto de datos idóneo para las necesidades de predicción de la comparativa. A su vez se implementaron técnicas de limpieza de texto efectivas que generaron un conjunto de datos estandarizado. El análisis y limpieza de datos dieron paso a cuatro enfoques principales que representan la evolución histórica de las técnicas procesamiento de lenguaje natural: *Bag of words*, redes neuronales convolucionales (Conv1D), redes neuronales recurrentes (biLSTM) y mecanismos de atención (BERT). El enfoque clásico de *Bag of words* y las redes neuronales convolucionales no implicaron grandes gastos computaciones, mientras los modelos de Deep Learning más avanzados implicaron un alto costo computacional el cual puede apreciarse en los *notebook* referenciados anteriormente.

En su mayoría las técnicas utilizadas presentaron resultados aceptables que concuerdan con su avance tecnológico en el paso del tiempo. Sin embargo, *Bag of words* presentó un resultado inesperado, dado que su rendimiento es uno de los mejores, su costo computacional es reducido y puede considerarse el método más antiguo para enfrentar este tipo de problemas, dentro de las técnicas trabajadas en la comparativa. Las técnicas clásicas no deben descartarse de plano por su antigüedad, como lo muestra la comparativa, dado que según la naturaleza de los datos pueden presentar una relación costo/rendimiento ideal para algunos problemas de procesamiento de lenguaje natural.

References

[1] *Tweet sentiment extraction*. [Online]. Available: https://www.kaggle.com/c/tweet-sentiment-extraction.

[2] *Datasets resource center*, Mar. 2021. [Online]. Available: https://appen.com/open-source-datasets/.

[3] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[5] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[6] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://www.aclweb.org/anthology/N19-1423.

[8] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, Berlin, Heidelberg: Springer-Verlag, 1999, p. 319, ISBN: 3540667229.

[9] Z. Huang, W. Xu, and K. Yu, *Bidirectional lstm-crf models for sequence tagging*, 2015. arXiv: 1508.01991 [cs.CL].

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart and J. L. Mcclelland, Eds., Cambridge, MA: MIT Press, 1986, pp. 318–362.

[11] I. C. Education, *What is natural language processing?* [Online]. Available: https://www.ibm.com/cloud/learn/natural-language-processing.

[12] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, pp. 43–52, Dec. 2010. DOI: 10.1007/s13042-010-0001-0.

[13] H. Widdowson, "J.r. firth, 1957, papers in linguistics 1934–51," *International Journal of Applied Linguistics*, vol. 17, pp. 402–413, Oct. 2007. DOI: 10.1111/j.1473-4192.2007.00164.x.

[14] T. Zhao, Q. Cao, and Q. Sun, "An improved approach to traceability recovery based on word embeddings," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, 2017, pp. 81–89. DOI: 10.1109/APSEC.2017.14.

[15] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002, ISSN: 0360-0300. DOI: 10.1145/505282.505283. [Online]. Available: https://doi.org/10.1145/505282.505283.

[16] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03, Toronto, Canada: Association for Computing Machinery, 2003, pp. 41–47, ISBN: 1581136463. DOI: 10.1145/860435.860445. [Online]. Available: https://doi.org/10.1145/860435.860445.

[17] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03, Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 142–147. DOI: 10.3115/1119176.1119195. [Online]. Available: https://doi.org/10.3115/1119176.1119195.

[18] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 455–465. [Online]. Available: https://www.aclweb.org/anthology/P13-1045.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13, Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] S. Varsamopoulos, K. Bertels, and C. Almudever, *Designing neural network based decoders for surface codes*, Nov. 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[23] Y. Yao and G. Li, *Context-aware sentiment word identification: Sentiword2vec*, 2016. arXiv: 1612.03769 [cs.CL].

[24] Z.-H. Deng, H. Yu, and Y. Yang, "Identifying sentiment words using an optimization model with l 1 regularization," Feb. 2016.

[25] J. Liang, X. Zhou, Y. Hu, L. Guo, and S. Bai, "Sentiment word identification with sentiment contextual factors," in *Web Technologies and Applications*, R. Cheng, B. Cui, Z. Zhang, R. Cai, and J. Xu, Eds., Cham: Springer International Publishing, 2015, pp. 215–226, ISBN: 978-3-319-25255-1.

[26] N. Koprowicz, *A simple solution using only word counts*, Agu 2020. [Online]. Available: https://www.kaggle.com/nkoprowicz/a-simple-solution-using-only-word-counts/notebook.

[27] E. Montoya, *Glove + rn*, Enero 2021. [Online]. Available: https : / / www . kaggle . com / nietzs96 / glove - convolutional.

[28] ——, *Fasttext + convolutional*, Enero 2021. [Online]. Available: https://www.kaggle.com/nietzs96/fasttext-convolutional.

[29] ——, *Glove + bilstm*, Enero 2021. [Online]. Available: https://www.kaggle.com/nietzs96/glove-bilstm.

[30] ——, *Fasttext + bilstm*, Enero 2021. [Online]. Available: https://www.kaggle.com/nietzs96/fasttext-bilstm.

[31] T. Rajapakse, *Question answering model*, Dec. 2020. [Online]. Available: https://simpletransformers.ai/docs/qa-model/.

[32] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: https : / / www . aclweb . org / anthology/D16-1264.

[33] Vivekjadhavr, *Tweet sentiment extraction*, Jan. 2021. [Online]. Available: https://vivekjadhavr.medium.com/tweet-sentiment-extraction-6cdf7a136fc3.

[34] E. Montoya, *Distilbert question and answer - preentrenado con el dataset stanford question answering dataset*, Enero 2021. [Online]. Available: https://www.kaggle.com/nietzs96/monograf-a-distilbert.

[35] ——, *Roberta - modelo huggingface preentrenado tf*, Enero 2021. [Online]. Available: https://www.kaggle.com/nietzs96/monograf-a-roberta.