MODELO PARA IDENTIFICACIÓN DE SARCASMO EN TEXTO PARA COMENTARIOS EN VIDEOS DE YOUTUBE EN ESPAÑOL

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA

JUAN JULIO CESAR CAMPAS BUITIMEA

DIRECTOR

DR. SAMUEL GONZÁLEZ LÓPEZ

H. NOGALES, SONORA, MÉXICO. **JUNIO DE 2022.**

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Nogales
División de Estudios de Posgrado e Investigación

Nogales, Sonora, 23/junio/2022
Oficio No. DEPI/177/2022.

REYNALDO GUTIÉRREZ GUTIÉRREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE.

Por este medio le comunicamos a Usted que el trabajo de Tesis denominado: "**MODELO PARA LA DETECCIÓN DE SARCASMO EN TEXTO PARA COMENTARIOS EN VIDEOS DE YOUTUBE EN ESPAÑOL**.", que presentó el alumno **JUAN JULIO CESAR CAMPAS BUITIMEA**, con número de control 18341002, candidato a obtener el grado de **MAESTRO EN SISTEMAS COMPUTACIONALES**, ha sido revisado por los miembros del Comité Tutorial y cubiertas las observaciones realizadas, se Autoriza su Impresión y se Acepta para su Evaluación en la presentación del Examen de Grado.

Agradeciendo de antemano el apoyo brindado al presente, le reitero mi consideración distinguida.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE NOGALES
DIVISIÓN DE ESTUDIOS DE,
POSGRADO E INVESTIGACIÓN

ATENTAMENTE
Excelencia en Educación Tecnológica.
"La Ciencia y la Tecnología para la Liberación del Hombre".

JESUS RAUL CRUZ RENTERIA
REVISOR

LUIS ARTURO MEDINA MUÑOZ
REVISOR

SAMUEL GONZALEZ LOPEZ
REVISOR

Instituto Tecnológico de Nogales
División de Estudios de Posgrado e Investigación

H. Nogales Sonora, 23/junio/2022.
Oficio No. DEPI/178/2022.

JUAN JULIO CESAR CAMPAS BUITIMEA
CANDIDATO A OBTENER EL GRADO DE MAESTRO EN SISTEMAS COMPUTACIONALES
PRESENTE.

De acuerdo con el Reglamento de Titulación del Sistema Nacional de Institutos Tecnológicos de la Secretaría de Educación Pública y habiendo cumplido con todas las indicaciones que el Comité Tutorial realizó, con respecto a su Tesis titulada: "**MODELO PARA LA DETECCIÓN DE SARCASMO EN TEXTO PARA COMENTARIOS EN VIDEOS DE YOUTUBE EN ESPAÑOL**", la División de Estudios de Posgrado e Investigación Autoriza su Impresión.

Agradeciendo de antemano el apoyo brindado al presente, le reitero mi consideración distinguida.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"La Ciencia y la Tecnología para la Liberación del Hombre"®

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE NOGALES
DIVISIÓN DE ESTUDIOS DE,
POSGRADO E INVESTIGACIÓN

REYNALDO GUTIÉRREZ GUTIÉRREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo

RGG/cmmj

MODELO PARA IDENTIFICACIÓN DE SARCASMO EN TEXTO PARA COMENTARIOS EN VIDEOS DE YOUTUBE EN ESPAÑOL

Resumen

En nuestra sociedad actual surgen nuevas fuentes de información y nuevas plataformas para generar opinión, dentro de toda esta información nueva que se crea día con día podemos encontrar opiniones sobre productos, política, noticias, personas famosas etcétera. Desafortunadamente la capacidad de análisis humana de la información que estos comentarios generan tiene limitantes los cuales se pueden superar computacionalmente con el procesamiento de lenguaje natural. Las diferentes opiniones plasmadas en los videos de Youtube, generan gran cantidad de información, comentarios que pueden ser de diferentes tipos. En este proyecto se busca analizar las opiniones del público en videos de autos, específicamente en las reseñas de nuevos modelos para identificar el Sarcasmo. Para ello se plantea el uso de técnicas de minería de datos y procesamiento de lenguaje natural utilizando técnicas de procesamiento como Naive Bayes, Random Forest y Redes Transformer (Deep learning) mostrando los resultados en términos de F-measure.

Abstract

In our current society, new sources of information and new platforms arise to generate opinions. Within all this new information that is created every day, we can find opinions about products, politics, news, famous people, etc. Unfortunately, the human ability to analyze the information generated by these comments has limitations which can be overcome in a certain way with natural language processing.

The different opinions reflected in YouTube videos generate a large amount of information, comments that can be of different types. This project seeks to analyze the opinions of the public in car videos, specifically in the reviews of new models to identify Sarcasm and Cynicism. For this, the use of data mining techniques and natural language processing is proposed.

Agradecimientos

Agradezco profundamente a mis docentes, asesores y compañeros por su apoyo y paciencia, también por su apoyo brindado para la realización de este trabajo. Agradezco también a mis padres por su incondicional apoyo a lo largo de este proceso.

Agradezco a mis amigos por toda su ayuda y palabras de aliento a continuar mi camino para alcanzar mis propósitos. Agradezco al instituto tecnológico de nogales por brindarme la oportunidad de continuar preparándome y alcanzar uno más de mis objetivos.

Índice de contenido

Resumen v
Abstract ix
Agradecimientos x
Índice de contenido viii
Índice de Tablas ix
Índice de figuras x
Capítulo 1: Introducción. 12
Capítulo 2: Marco teórico 22
Capítulo 3: Estado del arte 40
Capítulo 4: El sarcasmo y sus implicaciones 69
Capítulo 5. Metodología 76
Capítulo 6. Experimentación y resultados 93
Capítulo 7. Análisis 103
Capítulo 8: Referencias 119

Índice de Tablas

Tabla 3.1 N-grams 27
Tabla 5.1.3 Normalización 81
Tabla 5.1.4 Preprocesamiento de datos en corpus 82
Tabla 6.4.1 Ejemplo de codificación del corpus 98
Tabla 7.1 Distribución de comentarios del corpus por polaridad en la sección de entrenamiento. 104
Tabla 7.2 Distribución de comentarios del conjunto por polaridad en la sección de prueba. 105
Tabla 7.6.1 Matriz de confusión 108
Tabla 7.6.1 Parámetros 108
Tabla 7.6.3 Matriz de confusión 109
Tabla 7.6.4 Parámetros 109
Tabla 7.6.5 Matriz de confusión 110
Tabla 7.6.6 Parámetros 110
Tabla 7.6.7 Matriz de confusión 110
Tabla 7.6.8 Parámetros 110
Tabla 7.6.9 Matriz de confusión 111
Tabla 7.6.10 Parámetros 111
Tabla 7.6.11 Matriz de confusión 111
Tabla 7.6.12 Parámetros 111
Tabla 7.6.13 Matriz de confusión 113
Tabla 7.6.14 Parámetros 113
Tabla 7.6.15 Concentrado de Naive Bayes y BERT 113

Índice de figuras

Figura 3.1 Proceso de categorización del texto. 25
Figura 3.2. Árbol de decisiones 31
Figura 3.3 Hiperplano en un SVM 34
Figura 3.4 Datos inseparables linealmente proyectados en un espacio superior. 35
Figura 3.5 Validación cruzada 5 veces 37
Figura 3.1.1 Historia creada de forma aleatoria 42
Figura 3.1.2 Historia larga creada de forma aleatoria 43
Figura 3.1.2 Estructura básica de una red Transformer 44
Figura 3.1.4 Tokenizacion de posición 45
Figura 3.1.5 Estructura de un codificador 45
Figura 3.1.6 Frase dividida en dos 46
Figura 3.1.7 asociación de frases 46
Figura 3.1.8 Redes neuronales de un codificador 46
Figura 3.1.9 Comparaciones entre salidas 47
Figura 3.1.12 Aplicación de capa SOFTMAX a matriz de puntajes 48
Figura 3.1.13 Matriz de puntajes normalizada 48
Figura 3.1.1 múltiples bloques atencionales 49
Figura 3.1.16 Estructura del bloque residual 50
Figura 3.1.17 Estructura de normalización 51
Figura 3.1.1 Bloque traductor 52
Figura 3.1.19 comparadores de salidas 53
Figura 3.1.20 enmascaramiento 53
Figura 3.1.21 Estructura del bloque atencional 54
Figura 3.1.22 Capa lineal y SOFTMAX 55
Figura 3.2.1 Ejemplo de inferencia 56
Figura 3.2.2 Equivalencia semántica 57
Figura 3.2.3 Pregunta respuesta 57
Figura 3.2.4 Análisis de sentimientos 58
Figura 3.2.5 Aceptabilidad lingüística 58
Figura 3.2.6 Equivalencia semántica 58
Figura 3.2.7 Codificación y decodificación 60
Figura 3.2.10 Dato de entrada 60
Figura 3.2.11 Datos de entrada con tokens de separación e inicio 61
Figura 3.2.12 Representación de endoder 61
Figura 3.2.13 Ejemplo de bidireccionalidad de palabras 62
Figura 3.2.14 Ejercicios de bidireccionalidad 62
Figura 3.2.15 Bidireccionalidad de frases 63
Figura 3.2.16 Ejemplo de inferencia 64
Figura 3.2.17 Equivalencia semántica 65
Figura 3.2.18 Análisis de sentimientos 65
Figura 3.2.19 Aceptabilidad lingüística 66
Figura 5.1 Proceso de experimentación 77

Figura 5.1.1 Modelo de clasificación 1 78
Figura 5.1.2 Representación gráfica del entorno de programación. 79
Figura 5.1.3 Fragmento de código 83
Figura 5.1.4 Fragmento de código 84
Figura 5.1.5 Fragmento de código 85
Figura 5.1.8 Fragmento de código 86
Figura 5.2.1 Modelo de clasificación 2 propuesto 87
Figura 5.2.2 Cargar las dependencias requeridas. 89
Figura 5.2.3 Cargar las dependencias requeridas. 90
Figura 5.2.4 Cargar las dependencias requeridas. 90
Figura 5.2.5 Cargar datos y Codificación de texto 91
Figura 5.2.6 Construir modelo 92
Figura 5.2.8 Construir modelo 93
Figura 6.1 Secuencia lógica general del sistema 95
Figura 6.6 Ilustración del preprocesamiento de corpus. 99
Figura 6.7 Ejemplo de paso de selección de características 100
Figura 7.1 gráfica de distribución del conjunto de entrenamiento 104
Figura 7.7.1 Imagen de palabras que forman partes de las características con 200 muestras sin modificar la frecuencia mínima necesaria 116
Figura 7.7.2 Baja frecuencia mínima. Imagen de palabras que forman parte de las características con 200 muestras modificando la frecuencia mínima necesaria. 116
Figura 7.7.3 Nube de palabras con pocas muestras 117

Capítulo 1: Introducción.

1.1 Antecedentes

Las tecnologías de la información y comunicación avanzan modificando el desarrollo de la sociedad y esto modifica actividades tan cotidianas como comunicación interpersonal adoptando nuevos instrumentos que la facilitan y la modifican al mismo tiempo para dar paso a nuevas formas de interacción. Según el canadiense Marshall McLuhan, se podría considerar a los medios de comunicación como una extensión del cuerpo humano, como si estos fueran una segunda boca o un tercer ojo (McLuhan, 1994), por lo que se puede decir que los teléfonos inteligentes, computadoras o sistemas de cómputo son una extensión del cuerpo humano que nos ayudan a procesar información y a comunicarnos de una manera fácil y efectiva.

McLuhan también adoptó el término Aldea Global, concepto tomado más adelante por el Dr. Néstor Alejandro Ramos, (Ramos, 2019) lo cual describe las consecuencias de la transformación de la cultura material, que principalmente se ven reflejadas en los diferentes medios de comunicación que se usan hoy en día y como se vive día a día con esto. Actualmente es fácil comunicarse con una persona ubicada al otro lado del mundo, leer opiniones y comentarios de personas que usaron o compraron productos o servicios, incluso sin conocer a la gran mayoría de estas personas; pero finalmente existe esa facilidad de comunicación que convertiría al mundo actual en una gran aldea global. En este momento nos encontramos en una cuarta revolución Industrial (Araque González, Gómez Vásquez, Vélez Uribe, & Suárez, 2021), revolución que encuentra a internet y a lo que se conoce como Web 2.0 que desde sus inicios en 2003 el usuario adquiere un nuevo rol, pasando de ser consumidor de contenidos a ser también productor de los mismos como en el caso de los docentes usando las Tecnologías de la información para mejorar sus enseñanzas (Vargas Cárdenas, 2018) , de receptor pasivo a emitir juicios y opiniones. Posteriormente con la creciente moda por el uso de redes sociales, que ha ido en aumento de 2005 a la fecha, se ha reconfigurado nuevamente el rol del usuario promedio permitiendo que cada uno pueda dar sus opiniones, compartir sus pensamientos y sus sentimientos por igual, así como la creación de círculos sociales virtuales con usuarios con quienes se puede tener una mayor afinidad.

No cabe duda, las redes sociales han contribuido de gran manera a fomentar el desarrollo de los medios de comunicación electrónicos en su forma escrita, a la par también han producido interés en otras áreas del conocimiento siendo la lingüística computacional una de esas áreas (Arias, 2018).

En el procesamiento de lenguaje natural o también llamado la lingüística computacional, se interesa por el estudio y modelado del lenguaje humano usando métodos computacionales, también podemos ubicar la tarea de minería de opinión y análisis de sentimiento, que cobró un especial interés en la Web 2.0 y las redes sociales por la posibilidad de explorar la inmensa cantidad de información creada todos días por ellas.

La detección de emociones en texto se ha desarrollado con el fin de proporcionar pistas para futuras interacciones entre humanos y computadoras, de modo que las computadoras puedan actuar como actores sociales para lograr interacciones más creíbles (R. Cowie, 2001). Además de la detección de emociones a partir de textos, se ha trabajado para detectar los estados emocionales de los usuarios a partir de fuentes multimodales como el audio, los gestos y la mirada en la última década (Jiménez Carrillo, 2022).

Dada la gran cantidad de comentarios, opiniones y gustos que se expresan diariamente a través de diversos sitios en internet (blogs, foros, Facebook, LinkedIn, YouTube, etc.) resulta muy útil poder extraer de forma automática el sentimiento predominante asociado a una marca, empresa o producto, principalmente para planificar su potenciación si es un comentario positivo o cómo contrarrestarlo si es negativo (Binali, 2010). Por ello se analiza Internet utilizando herramientas en línea pues esta información es espontánea y poco estructurada diferenciándose en su inmediatez de la obtenida a través de encuestas o estudios de mercado (M. Chunling, 2005).

El monitoreo de sentimientos de textos en las redes sociales (que adopta diferentes nombres en inglés como *sentiment analysis*, *opinion mining*, *brand monitoring*, *buzz monitoring*, *online anthropology*, *market influence analytics*, *conversation mining*, *online consumer intelligence*, *user generated content)* (Binali, 2010) es el proceso que determina el tono emocional que hay detrás de una palabras determinadas, si una frase contiene una opinión positiva o negativa sobre un producto, marca, institución, organización, empresa, evento o persona.

1.2 Planteamiento del problema

El análisis de información y el proceso de transformación de la información en conocimiento son capacidades exclusivas de los seres humanos; involucran y mezclan elementos y procesos como conocimientos previos, análisis de contexto, ventajas, desventajas apropiación del conocimiento, entre otros aspectos.

Las redes sociales han cambiado nuestra vida en los últimos años. Por ejemplo, según datos de IAB Spain, en 2021 los españoles utilizaban hasta 5 redes sociales diferentes al mes. El 45% de los usuarios reconoce que estas plataformas influyen directamente en sus decisiones de compra. Además, un 85% de la población española de entre 16 y 70 años de edad usa redes sociales con asiduidad.

Para las empresas, esta realidad social ha transformado en una obligación la puesta en marcha de canales de comunicación y comercialización a través de las redes sociales. Los mismos requieren de un feedback o retroalimentación de los consumidores, mediante comentarios o mensajes que marquen la aceptación o rechazo a cada una de las propuestas, productos o servicios.

A pesar de esto, uno de los grandes cuellos de botella de la comunicación digital para las empresas y organizaciones es el sarcasmo en los mensajes escritos, que dificulta la comprensión de las necesidades de los usuarios en las redes sociales. Se puede considerar que este reto tiene claras aplicaciones comerciales, como la personalización de los sistemas de recomendación o la minería de datos de opinión, que permitiría medir de forma más acertada el sentimiento que genera en el público un producto determinado a partir de las opiniones que se expresan en la red. Por lo tanto, el incorporar la detección de sarcasmo afecta directa o indirectamente el desempeño de los modelos en tareas tan diversas como el análisis de sentimientos, la extracción de opiniones, atención a clientes, segmentación de clientes, análisis de mercados financieros, el comercio electrónico, la gestión de foros, el marketing en línea y evaluación de productos.

Los sentimientos se pueden clasificar en positivos, negativos y neutros. Sin embargo, el lenguaje natural es complicado y ambiguo por lo que enseñar a una máquina a que analice los matices gramaticales, variaciones culturales, jergas, expresiones coloquiales o a distinguir faltas de ortografía, la sinonimia o la polisemia dentro de un contexto que determina el tono de la conversación es francamente difícil.

Por lo anterior el poder contribuir mediante una propuesta que facilite el análisis de las opiniones clasificándolas en una escala que además realice la tarea con buena precisión sería una buena aportación en el campo de estudio.

1.3 Objetivos generales y específicos

General

La creación de un algoritmo que sea capaz de identificar el sarcasmo en comentarios escritos creando un modelo de predicción utilizando técnicas de minería datos como deep learning y Machine learning con el propósito de optimizar el reconocimiento de sentimientos sobre textos en español.

Objetivos específicos

1. Diseñar un método para tokenizar las palabras dentro de un comentario.
2. Crear un corpus que se adapte a los requerimientos de la tarea.
3. Desarrollar un algoritmo que sea capaz de clasificar correctamente un comentario en texto tomando en cuenta su contexto.
4. Desarrollar un sistema que implemente la metodología propuesta.
5. Diseñar un método para eliminar los símbolos que no son relevantes para el proceso del texto.

1.4 Justificación

Una arquitectura basada en híbridos para la detección de emociones ha sido propuesta y validada con resultados experimentales que sugieren que la información semántica y sintáctica puede mejorar enormemente la exactitud de la predicción. Existen investigaciones que se enfocan en determinar la portabilidad entre dominios y el desarrollo de heurísticas que se pueden implementar en la detección de emociones para reducir el sesgo de clasificación (Binali, 2010).

El problema es que a pesar de que los métodos basados en el aprendizaje pueden determinar automáticamente las probabilidades entre las características y las emociones, los métodos basados en el aprendizaje todavía necesitan palabras clave y supervisión humana, pero sólo en la forma de características. El contexto también es un factor importante a la hora de identificar el verdadero significado de los textos.

Comparado con la investigación de detección de emociones en otros campos, la detección de emociones en texto todavía no está madura y requiere mejoras para ser montado como práctico. Estas mejoras incluyen una mejor comprensión acerca de vocabularios recién evolucionados (modismos y métodos de detección que permitan más categorías de emociones). Existen aún preguntas sin respuesta y territorios inexplorados (como la neurociencia afectiva) que presentan desafíos y oportunidades que pueden sustentar la investigación durante varias décadas (Zloteanu, 2016).

Una de las principales industrias del mundo es la automotriz (Dirección General de Industrias Pesadas y de Alta Tecnología , 2012), México tiene plantas de manufactura de vehículos las cuales surten a norte, centro y Sudamérica, Marcas como BMW, Mercedes Benz, AUDI, FORD, GM, HONDA, TOYOTA y MAZDA tienen plantas en México (Carrillo, 2013) y de ahí la importancia de saber el estado actual del mercado en la región con el fin de tener un diagnóstico del mismo, conocer sus gustos y preferencias, que características son del agrado del comprador latino y que características no lo son (Noguez, 2016). La opinión de cada posible consumidor es importante para aquellos que deseen vender autos. La finalidad que se persigue al adquirir el conocimiento es dar un avance en el estudio de la inteligencia artificial en idioma español dado que la información y desarrollo de la misma es escasa a comparación con el idioma inglés. El mayor beneficiario de este estudio será la industria automotriz y el campo de aplicación para los resultados obtenidos es en el marketing inteligente enfocado al sector automotriz.

El conocimiento que se generará no solo será valioso para las marcas automotrices, comprender el sarcasmo y el cinismo en el texto permitirá obtener una imagen precisa de lo que se solicita (Paula Carvalho, 2009) . En la atención médica, la detección de sarcasmos podrá funcionar como detección de lesiones cerebrales en una etapa temprana. En una investigación realizada en el *University College* de Londres, pudieron demostrar que las personas con lesiones cerebrales tenían problemas de comprensión del sarcasmo en comparación con otro grupo el cual podía detectar el sarcasmo con facilidad (Channon, 2004) . Otra área importante es la de evaluar si una amenaza es real o no, las consecuencias podrían ser devastadoras si se toman en serio algunos mensajes sarcásticos o cínicos. P.ej. la humorística cuenta de twitter *dailykisev* tuiteó "¡Y 12 bombarderos parten ahora ... a Lituania!" después de que resultó que Lituania le dio a Rusia cero puntos por su desempeño en el Festival de la Canción de Eurovisión 2015. El servicio secreto de EE. UU también dijo que están empezando a trabajar en un detector de sarcasmo para *twitter*. A pesar de que los expertos en el área expresan su escepticismo hacia la idea, es considerable que al menos dos personas hayan sido detenidas falsamente como consecuencia de publicaciones sarcásticas durante los últimos cinco años (Silver, 2014).

1.5 Alcances y límites

Se ha tomado la decisión de tomar como tópico principal las reseñas de automóviles en internet para el clasificador ya que el juicio que se debe tomar en ciertas características para evaluar si un automóvil es o no bueno son subjetivas y por tanto es un campo fértil para que gente con puntos diferentes expresen sus opiniones de manera creativa y libre.

Se ha decidido crear un clasificador de texto en español excluyendo todos los demás idiomas para concentrar esfuerzos en analizar los comportamientos de la gente de habla hispana en países como México, España y Argentina. Los comentarios y en general toda la información recabada y usada para el proyecto proviene única y exclusivamente de Youtube ya que es la plataforma que mejor se adapta al escrutinio de sus comentarios para ser analizados y a su vez para servir como prueba para el clasificador.

Para propósitos de esta tesis la magnitud del problema implica a todas las empresas que tengan negocios en países de habla hispana, debido a que el dinero que se puede invertir en investigación, desarrollo y publicidad puede tener un mal enfoque si estos sistemas fallan, por lo tanto la distribución geográfica puede cubrir todo Sudamérica, centro América, México y España pero debido a la cantidad de jergas con diferentes palabras de cada región se decidió delimitar la distribución a solo México, concretamente a las empresas automotrices que venden en este país.

1.6 Estructura de la tesis

Muestra la información general del proyecto: Planteamiento, objetivos generales y específicos, justificación, alcances y límites. Posteriormente se muestra toda la información relacionada con los retos al trabajar con redes sociales virtuales y niveles de lenguaje natural, seguido de esto se presenta el estado del arte con respecto al análisis automático de opiniones en redes sociales. En capítulos posteriores se presenta la aproximación para el análisis automático de opiniones en redes sociales para después llegar a describir la implementación del sistema y los recursos adicionales utilizados, posteriormente se presentan las pruebas realizadas al sistema y resultados obtenidos y para el final se enlistan las conclusiones y plantea el trabajo a realizarse a futuro.

Capítulo 2: Marco teórico

El sarcasmo como fenómeno lingüístico ha sido ampliamente estudiado, (Wilson, 2006), el sarcasmo surge de la disparidad situacional. La relación entre la incongruencia del contexto y el procesamiento del sarcasmo (por humanos) se ha estudiado (Pexman S. L., 2003). También se han investigado varias propiedades del sarcasmo.

Campbell y Katz afirman que el sarcasmo se produce en diferentes dimensiones, a saber, expectativas fallidas, falta de sinceridad pragmática, tensión negativa, presencia de una víctima y componentes estilísticos como las palabras de emoción (Katz J. D., 2012). Eisterhold observa que el sarcasmo puede identificarse según la declaración que precede y sigue a la declaración sarcástica (Jodi Eisterhold, 2006). Esto es particularmente cierto en los casos en que la incongruencia no se expresa dentro del propio texto sarcástico.

Las investigaciones han demostrado que los niños en una edad temprana pueden entender el sarcasmo. En un experimento llevado a cabo en la Universidad de Calgary, se demostró que los niños de 5 años podían captar fácilmente formas simples de expresiones (Pexman P. M., 2013). Pero, aunque la mayoría de las personas son hábiles para detectar el cinismo en situaciones del mundo real, el cinismo en el texto es algo diferente. A diferencia de la detección de temas. Al separar artículos de diferentes deportes, la detección del cinismo se considera una tarea muy difícil. En el caso de clasificar deportes, cada deporte (tema) normalmente tiene su propio vocabulario que es poco probable que aparezca en cualquier otro documento que no sea ese deporte. Este vocabulario es a menudo suficiente para hacer un clasificador preciso. Sin embargo, la detección del cinismo se considera una tarea muy difícil debido a que aparece en diversas formas y en todo tipo de contextos El cinismo a menudo tiene interpretaciones ambiguas y, a menudo, expresa lo contrario a lo que se dice literalmente (Butler, 1953). Otra razón por la que el cinismo podría ser más difícil de detectar en un texto escrito que en una conversación hablada es debido a que esa información contenida en el tono de la voz o las expresiones faciales que pueden ser importantes para entender, se pierden. Otra teoría más es que lleva más tiempo escribir en una computadora que hablar cara a cara, y que las personas usan ese tiempo extra para escribir un sarcasmo o cinismo más complejo que el que usan normalmente (Chin, 2011).

La ironía es una forma sofisticada de uso del lenguaje que reconoce una brecha entre el significado deseado y el significado literal de las palabras. Aunque es un fenómeno lingüístico ampliamente estudiado, no parece existir una definición clara (Filatova, 2012). La ironía se puede dividir en dos grandes categorías: ironía situacional y verbal. El primero es, por ejemplo, una compañía de cigarrillos que tiene carteles de no fumar en el vestíbulo. La última, que se considera en esta tesis de maestría, se define comúnmente como "decir lo contrario de lo que quieres decir" (Butler, 1953), donde se supone que la diferencia entre el dicho y el significado es clara. Otras definiciones establecen que es cualquier forma de negación sin ningún marcador (Giora, 1995), otra dice que la ironía viola la máxima de no decir lo que crees que es falso [Grice, 1975]. Otra definición más es que una expresión tiene que ser ecoica para ser juzgada como irónica (Sperber, 1995).

1. "¡Gracias, Janet Jackson por otro año más de rock clásico del Super Bowl!"

2. "Está con su otra mujer: la Xbox 360. Son las 4:30 tonto. Claro que puedo dormir con del ruido de las armas.

3. "WOW, siento tu interés ...?"

4. "[I*] Love The Cover*"

Estos ejemplos demuestran diferentes situaciones de ironía y sarcasmo en diferentes textos. Los ejemplos (1), (2) y (3) son tweets de Twitter y el ejemplo (4) se toma de una revisión de Amazon. En el ejemplo (1) se refiere a un rendimiento muy pobre en el super bowl y se refiere al desempeño vergonzoso de Janet Jackson el año anterior. El ejemplo (2) consiste en tres oraciones que son sarcásticas por sí mismas, y cuando se combinan, el sarcasmo se vuelve obvio. En el ejemplo (3), una indicación de ironía es que "WOW" está escrito con letras mayúsculas en combinación con los puntos suspensivos al final de la oración. El ejemplo de Amazon (4) también podría tomarse de una reseña de libro positiva. Sin embargo, esto se ha tomado de una revisión titulada "No juzgues el libro en su portada", y con este título revela un tono sarcástico.

Es obvio que no existe una regla o algoritmo simple que pueda capturar la ironía o el sarcasmo. En este estudio veremos varios aspectos (sentimientos, vocabulario, lingüística, puntuación, etc.) que pueden desempeñar un papel en la formación de contenido irónico y sarcástico. El tema de la mayoría de las publicaciones en los últimos años se ha movido hacia

el enfoque cognitivo para detectar mentiras o sarcasmo , principalmente debido a la facilidad de manipulación experimental de la carga cognitiva en los escenarios de interrogación (Sporer, 2016).Sin embargo, es prematuro limitar los horizontes y alejarse del enfoque basado en las emociones para detectar el sarcasmo, ya que todavía tiene mucho que informarnos sobre el proceso de los juicios de veracidad y el papel que desempeñan las emociones en el sarcasmo.

La detección computacional del sarcasmo es un área de investigación relativamente reciente. El trabajo inicial sobre detección de sarcasmo investiga el papel de las características léxicas y pragmáticas. Tepperman presenta el reconocimiento de sarcasmo en el habla utilizando señales prosódicas, espectrales (tono promedio, pendiente de tono, etc.) y contextuales (risas o respuestas a preguntas) (Joseph Tepperman, 2006). Carvalho usa características lingüísticas simples como interjección, nombres cambiados, etc. para la detección de ironía (Paula Carvalho, 2009). Davidov entrena un clasificador de sarcasmo con características sintácticas y basadas en patrones (Dmitry Davidov, 2010). Gonzalez e Ibanez estudian el papel de los anagramas y los emoticones en la detección del sarcasmo (Roberto Gonzalez-Ibanez, 2011). Liebrecht utiliza un conjunto de datos de tweets holandeses que contienen hashtags relacionados con sarcasmos e implementan un clasificador para predecir el sarcasmo (CC Liebrecht, 2013).

Riloff afirma que el sarcasmo es un contraste entre la palabra de sentimiento positivo y una situación negativa. Implementan un sistema basado en reglas que utiliza arreglos de frases verbales positivas y situaciones negativas extraídas de un corpus de tweets sarcásticos (Ellen Riloff, 2013). Ramteke presenta un enfoque novedoso para detectar frustraciones: el fenómeno en el que el sentimiento en las principales partes del texto se invierte con el sentimiento en partes más pequeñas y concluyentes (Ankit Ramteke, 2013).

La categorización del texto es un área que se superpone tanto con el procesamiento del lenguaje natural (NLP) como con el aprendizaje automático. La tarea es ordenar automáticamente los documentos en algunas categorías predefinidas. El texto en sí mismo no es susceptible de clasificación, y el primer paso en la categorización del texto es procesar y

extraer las características del texto. Este capítulo está dedicado a explicar los métodos que se pueden usar en el proceso de categorización del texto. Como se muestra en la figura 3.1, en el primer paso, describimos cómo se prepara el texto para extraer características, que es único para los problemas de categorización del texto. Los pasos restantes son comunes para muchos tipos de problemas de aprendizaje automático.

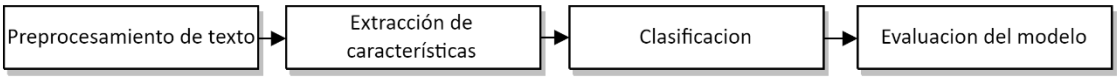


Figura 3.1 Proceso de categorización del texto.

2.1 Ingeniería de características

Crear características para el modelo es quizás la parte más pesada e importante y probablemente la razón por la cual un modelo tiene éxito o falla. A menos que los datos tengan muchas independencias se correlacionen bien, la ingeniería con los datos es necesaria. Esta parte del proceso a veces se denomina "arte negro", ya que cada problema de clasificación requiere un conjunto diferente de características, y ser creativo al extraer características puede ser tan valioso como elegir el mejor clasificador (Domingos, 2012). El tipo de funciones que se pueden extraer depende en gran medida de cómo se haya procesado el texto. Un método simple para clasificar texto es usar conteos para cada palabra que aparece en el corpus. Sin embargo, antes de hacerlo, podemos mejorar el texto utilizando los métodos presentados en esta sección.

2.2 Tokenización

Un token es una secuencia de caracteres que queremos tratar como un grupo. La descomposición de un texto en tokens permite crear recuentos de tokens, que se pueden usar como funciones. Un token podría ser un párrafo, una oración, etc., pero comúnmente las palabras se eligen como tokens en la categorización del texto. Ejemplo de diferentes tokenizadores de la biblioteca NLTK se pueden encontrar en su página de demostración.

2.3 Derivación y Lematización (*Stemming and Lemmatization)*

La derivación, junto con la lematización, es un método común en la NLP. El objetivo del método es reducir la cantidad de palabras involucradas eliminando el sufijo de la palabra para recuperar una "forma base" de la palabra. La derivación se basa en la idea de que las palabras con la misma raíz tienen un significado cercano y que la NLP se mejorará si esas diferentes palabras se pueden agrupar en un solo término (Porter, 1980), por lo tanto, en Aprendizaje automático, el número de características puede ser reducido si se usan vástagos en lugar de las palabras originales. En el ejemplo de las palabras:

Discutir

- Argumentó
- Argumentos
- Discutiendo

Un algoritmo de derivación podría identificar los sufijos "ando", "endo" y quitar las palabras a la raíz "argumentar". Como lo demuestra el ejemplo, una raíz no necesariamente tiene que ser una palabra. El algoritmo de eliminación de sufijos utilizado para proporcionar este ejemplo es el Porter Stemmer, de la biblioteca de Python NLTK, basado en el algoritmo desarrollado por Martin Porter (Porter, 1980).

En lingüística, un lema es la forma canónica de una palabra. Por ejemplo, el verbo "trabajar" puede aparecer como "trabajando", "trabajado". Su forma canónica "trabajo" es un lema, y el método de conversión de una palabra a su lema se conoce como lematización. Una lematización transformaría, por ejemplo, la palabra "ratones" en el lema "ratón". La lematización se considera más compleja que la derivación, ya que los algoritmos de lematización deben analizar primero un texto para identificar la parte del habla de las palabras, debido a este hecho, la lematización no se considerará en este informe.

2.4 Etiquetado de parte del discurso (POS) *Part Of Speech*

Otra herramienta que se puede utilizar para la extracción de características es un etiquetador de POS. Es una de las muchas herramientas interesantes que se pueden usar en el procesamiento del lenguaje natural. Los marcadores POS asignan a cada token su clase de palabra, es decir, etiqueta el token con su parte del habla. Un beneficio de usar un etiquetador de POS es distinguir los homónimos, que se muestran en el siguiente ejemplo en inglés.

1. Put/VB it/PRP back/RB.
2. I/PRP hurt/VBP my/PRP$ back/NN.

Aquí ¨Back¨ está etiquetado como un adverbio (RB) en la primera oración y como un sustantivo (NN) en la segunda. El etiquetador POS utilizado para este ejemplo se basa en un etiquetador simple basado en reglas desarrollado por Eric Brill en 1992. El algoritmo del etiquetador de Brill tiene dos pasos. En el primer paso, a cada palabra se le asigna una etiqueta estimada por un gran corpus etiquetado manualmente. Además, si aparecen palabras nuevas que no existen en el corpus etiquetado, se asignan de acuerdo con los supuestos: Si la primera letra es mayúscula, se le asigna una etiqueta de nombre. De lo contrario, se le asigna una etiqueta de la clase con palabras que generalmente terminan con las mismas tres letras. En el segundo paso, examina las etiquetas utilizando un conjunto de reglas que examinan el orden de las etiquetas (Brill, 1992).

2.5 N-grams

Los patrones pueden crearse concatenando tokens adyacentes en *n-grams* donde n = {1, 2, 3 ...}. Para el caso simple donde n es igual a uno también se llama *n-grams*. Un ejemplo de cómo se construyen los n-gramas se explica en la tabla 3.1 con la oración de ejemplo: "Nicolas adora jugar futbol".

n-grams.	Secuencia.	Longitud
1	“Nicolas", "Adora" "jugar”, “futbol"	4
2	" Nicolas adora ", adora jugar", " jugar futbol"	3
3	" Nicolas adora jugar ", " adora jugar futbol "	2

Tabla 3.1 N-grams

Tienden a aparecer en texto con respecto a otras palabras. Por lo general, los n-gramas nunca son más largos que n = 3. Es probable que los valores mayores creen "también" patrones complejos que rara vez coinciden.

2.5.1 Selección de características

El objetivo con la selección de características es seleccionar un subconjunto para usar en la clasificación. En los datos de texto siempre habrá características que son irrelevantes y redundantes. Las características redundantes son aquellas que no aportan nada o muy poco para distinguir las clases entre sí. Al hacer esto, la dimensionalidad se reducirá de modo que la cantidad de datos a procesar sea menor y esto ahorrará tiempo al realizar la clasificación. Otro beneficio, para algunos algoritmos de clasificación, es que disminuimos el riesgo de un ajuste excesivo de los datos.

2.6 Chi-cuadrado

Las estadísticas de Chi cuadrado son una medida para investigar la independencia entre variables estocásticas. Si las variables son similares, el chi-cuadrado produce un resultado bajo y, a la inversa, un resultado alto si las variables difieren mucho. Definición: Si las variables independientes x_i están normalmente distribuidas con media μ_i y varianza σ^2_i, entonces chi-cuadrado se define como en (1):

$$X^2 = \frac{(x_1 i \mu_1)^2}{\sigma_1^2} + \frac{(x_2 - \mu_2)^2}{\sigma_2^2} + \cdots + \frac{(x_\nu - \mu_\nu)^2}{\sigma_\nu^2} = \sum_{i=1}^{\nu} \frac{(x_i - \mu_i)^2}{\sigma_i^2} \quad (1)$$

Normalmente, las estadísticas de chi cuadrado se usan para estimar cómo encajan las buenas distribuciones. En este caso, la estadística de chi cuadrado se usa como una medida para encontrar el opuesto al filtrar las palabras en un texto que tiene distribuciones similares para ambas clases, por lo que es irrelevante para la clasificación (Pedregosa, 2011).

Ganancia de información: Para comprender la ganancia de información, necesitamos explicar qué es la entropía. La entropía es una medida estadística que puede interpretarse como la (*im*) pureza de colección o cuerpo (Mitchell, 1977) y se define como en (2):

$$H(p) = -\sum_{x \in X} p(x) \log p(x). \quad (2)$$

La ganancia de información se puede utilizar para seleccionar una secuencia de atributos para reducir la clase de una instancia de la manera más corta. En términos de entropía, la ganancia de información es (3).

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{3}$$

Al combinar las ecuaciones obtenemos la ecuación (4) para obtener información.

$$IG(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x_i, y_i) \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}. \tag{4}$$

En lugar de simplemente comparar dos atributos arbitrarios, la ganancia de información se puede generalizar para medir la disminución de la entropía comparando un atributo con un conjunto de atributos. Esto es preferible cuando se trata de la categorización de texto (Nicolosi, 2008). La ganancia de información se define en ese caso como en (5).

$$IG(t) = \sum_{c \in \{c_k, !c_k\}} \sum_{t \in \{t_i, !t_i\}} P(t,c) \log \frac{p(t,c)}{p(t)p(c)}. \tag{5}$$

2.7 clasificación

Cuando se construye un modelo para la clasificación de texto, es importante preguntar cuántos datos están disponibles. ¿mucho? ¿poco? ¿ninguno? Uno de los problemas más difíciles en la clasificación es conseguir suficientes datos de entrenamiento. Si hay disponible una gran cantidad de datos, entonces no es tan obvio qué clasificador usar y una idea podría ser elegir un clasificador con buena escalabilidad (Banko, 2001). Sin embargo, dependiendo de cómo estén disponibles los datos, hay tres métodos de aprendizaje diferentes.

2.8 Aprendizaje supervisado

El aprendizaje supervisado se basa en el aprendizaje de un modelo dado un conjunto de datos correctamente clasificados. El objetivo del aprendizaje supervisado es entrenar un modelo para reconocer atributos discriminantes (en la literatura estadística, el aprendizaje supervisado a veces se conoce como aprendizaje discriminante) en los datos (Michie, 1994). Digamos que queremos construir un modelo que pueda separar artículos de noticias sobre el

fútbol de la religión. Con un conjunto dado de datos etiquetados, el modelo puede ser entrenado para, por ejemplo, aprenda que algunas palabras (atributos) se usan solo, o más frecuentemente, en una de las clases. El modelo podría haber aprendido que los artículos que contienen palabras como "árbitro", "jugador" o "portero" es más probable que sea un artículo sobre el deporte. A la inversa, es más probable que aparezcan palabras como "dios", "iglesia" e "islam" en un artículo sobre religión. Se puede predecir que los nuevos artículos de noticias invisibles sean artículos sobre el fútbol o la religión, según la frecuencia de sus palabras.

2.9 Aprendizaje sin supervisión

Los datos etiquetados no siempre están disponibles y en esos casos el aprendizaje supervisado no es posible. El aprendizaje no supervisado es un método para encontrar patrones en los datos sin ninguna información sobre la medida de resultado de los datos. Es difícil validar los resultados obtenidos de una clasificación sin supervisión ya que la medida de resultado es desconocida. La fuerza del aprendizaje no supervisado reside en encontrar asociaciones en los datos y mostrar cómo se organiza (Mitchell, 1977).

2.10 Semi supervisado

El aprendizaje semi supervisado se basa en el hecho de que los datos etiquetados pueden ser difíciles de obtener y los datos no etiquetados son baratos. La idea es combinar un pequeño conjunto de datos etiquetados y expandirlos utilizando datos no etiquetados con la ayuda de aprendizaje no supervisado (Zhu, 2007). El resultado es entonces un gran conjunto de datos etiquetados, que tal vez contengan algo de ruido, que se pueden usar para la clasificación supervisada. Para obtener una alta precisión, una buena idea es probar diferentes clasificadores en sus datos. En el resto de esta sección presentamos cuatro clasificadores que se han utilizado durante este proyecto, cada uno de ellos se utiliza para el aprendizaje supervisado.

2.11 Árbol de decisiones

El aprendizaje basado en árboles se puede aplicar tanto a problemas de clasificación como de regresión. El procedimiento principal es dividir recursivamente los nodos de datos {xi} en dos nuevos nodos (*Nxi*≥*t*, *Nxj*≤*t*) por una dimensión de la característica *Xj* y un umbral t. El algoritmo para construir un árbol de decisión es básicamente dos pasos:

1. Divida el espacio de la característica X_1, X_2, ..., X_p en J regiones distintas y no superpuestas, R_1, R_2, ..., R_J.

2. Todas las observaciones que caen en la región R_J se predicen de la misma manera, que es simplemente la mayoría (media para la regresión) de los valores objetivo de las instancias de entrenamiento en R_J.

Digamos que después del paso 1 obtenemos dos regiones R_1 y R_2 y que la mayoría de las instancias de entrenamiento en R_1 son irónicas. Entonces, todos los nuevos ejemplos $x \in R_1$ serán predichos como irónicos.

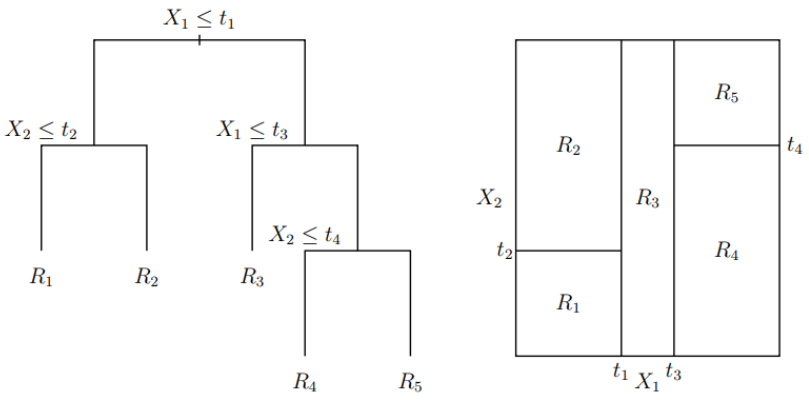


Figura 3.2. Árbol de decisiones

El espacio de características dividido en regiones Ri por divisiones tj. A la izquierda en la figura está la imagen del árbol de decisión. Todas las nuevas instancias x ∈ Ri serán predichas de la misma manera.

En la Figura 3.2 se muestra un ejemplo de cómo crece el árbol con cada división y cómo se divide el espacio de la característica en regiones. Aplicamos la división recursiva utilizando dos características X1 y X2. Primero dividimos X1 = t1. Luego dividimos la región X1 ≤ t1 en X2 = t2. Después de eso, dividimos la región X1> t1 en X1 = t3. Finalmente, dividimos la región X1> t3 en X2 = t4. De esto obtenemos cinco regiones.

Algunos métodos diferentes para medir la impureza en el nodo y evaluar dónde realizar la siguiente división son la impureza y entropía de Gini y el error de clasificación errónea

$$\sum_{k=1}^{K} p_{mk}(1 - p_{mk}). \tag{6}$$

$$-\sum_{k=1}^{K} p_{mk} \log p_m k \tag{7}$$

$$1 - \max_k p_{mk} \tag{8}$$

Para una clasificación binaria ($K = 2$) las ecuaciones (6), (7) y (8) se colapsan para

$$\text{Gini: } 2p(1-p) \tag{9}$$

$$\text{Entropy: } -p\log(p) - (1-p)\log(1-p) \tag{10}$$

$$\text{Misclassification: } 1 - \max(p, 1-p) \tag{11}$$

Un enfoque común es seguir dividiendo el árbol hasta que alcance algunos criterios de detención definidos, p. Ej. Cuando todos los nudos de la hoja son puros se produce una clasificación perfecta en los datos de entrenamiento, pero es muy probable que se adapte excesivamente y se comporte mal en los datos nuevos que no se ven. Otro enfoque es dejar de construir el árbol cuando una división no mejora la clasificación por algún valor de umbral. Este enfoque generará un árbol más pequeño, pero es miope ya que una mala división puede ser seguida por una buena división que separa las clases de manera efectiva. Un mejor enfoque es generar un árbol grande T_0 y luego podarlo, es decir, eliminar los nodos de la hoja, a un subárbol más pequeño T. La idea es mantener la estructura del árbol básico, pero eliminar las divisiones innecesarias que pueden causar un exceso de ajuste. Los subárboles candidatos se pueden seleccionar utilizando el criterio de enlace más débil definido como

$$C_\alpha = \sum_{m=1}^{|T|} Q_m(T) + \alpha|T| \tag{12}$$

Donde α es un parámetro de ajuste, cada uno correspondiente a un subárbol, que representa una compensación entre la complejidad del árbol y qué tan bien el árbol se ajusta

a los datos. Un valor de $\alpha = 0$ corresponde al árbol original T_0. Q_m *(T)* es cualquiera de las medidas de impureza en las ecuaciones (9), (10) y (11). El objetivo es, para cada α, encontrar un subárbol que minimice la Ecuación (12). A continuación, un ejemplo de cómo hacer crecer un árbol y seleccionar un subárbol adecuado.

1. Use la división binaria recursiva para hacer crecer un árbol grande en los datos de entrenamiento, deteniéndose solo

cuando cada nodo terminal tiene menos de un número mínimo de observaciones.

2. Aplique la poda de enlace más débil al árbol grande para obtener una secuencia de los mejores

subárboles, en función de α.

3. Utilice la validación cruzada de *K-fold* para elegir α. Es decir, dividir las observaciones de entrenamiento.

en K pliegues. Para cada $k = 1, ..., K$:

(a) Repita los Pasos 1 y 2 en todos los datos de entrenamiento, excepto en el *kth kth*.

(b) Evalúe el error de predicción en los datos en el pliegue *k-out* izquierdo como una función

de α. Promedie los resultados para cada valor de α, y elija α para minimizar el

error medio

4. Devuelva el subárbol del Paso 2 que corresponde al valor elegido de α

2.12 Máquinas de vectores de soporte (SVM) s

Los SVM son un conjunto de métodos de aprendizaje no supervisados para aprendizaje automático que pueden utilizarse para la clasificación, detección de anomalías y regresión. Una de las ventajas de SVM es que son muy eficientes cuando sus datos son de alta dimensión y también son efectivos en los casos en que la cantidad de dimensiones es mayor que la cantidad de muestras. Los SVM son especialmente buenos para resolver la categorización de texto e hipertexto ya que la aplicación disminuye el uso de ocasiones de entrenamiento de etiquetas (Witten, 2005). Las desventajas con SVM son que no proporcionan una estimación de la probabilidad. Para obtener estos se puede realizar un cálculo de validación cruzada. En los casos en que el número de muestras es significativamente menor que el número de características es probable que el método dé un resultado pobre. El SVM clasifica las categorías al dividirlas con un hiperplano. Un hiperplano es un subespacio de una dimensión menor que el espacio circundante.

La idea es encontrar un hiperplano que tenga la mayor distancia a cada clase porque se esfuerza por separar las categorías tanto como sea posible. La ecuación de hiperplanos que divide las categorías es la SVM para el caso específico.

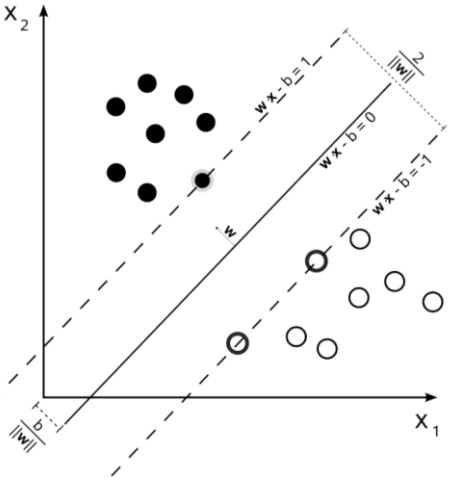


Figura 3.3 Hiperplano en un SVM

En la Figura 3.3 se observa un hiperplano de margen máximo y márgenes para un SVM entrenado con muestras de dos clases. Las muestras en el margen se denominan vector de soporte.

Para los casos en que los datos de entrenamiento son linealmente inseparables, es común proyectar los datos en una dimensión más alta para que se pueda hacer una separación más fácil de las clases no lineales. Una declaración clásica de T.M. La portada en 1965 llamada teorema de la portada establece que:

Un problema complejo de clasificación de patrones, fundido en el espacio de alta dimensión de forma no lineal, es más probable que sea linealmente separable que en un espacio de baja dimensión, siempre que el espacio no esté poblado densamente.

Esto es cierto porque ahora se usa una dimensión más alta del hiperplano para hacer la separación, por lo tanto, el nuevo hiperplano se ajustará mejor a los datos. Para evitar la proyección en un espacio más alto, se puede usar algo llamado "truco del kernel". Este "truco" suele ser computacionalmente más barato en comparación con el proyecto en una dimensión superior. Lo que hace el truco del kernel es que utiliza el resultado de una función del kernel de dos puntos de datos, lo que equivale a proyectarlos en un espacio dimensional superior y usar su producto interno en este nuevo espacio como se muestra en la Figura 3.4. Esto es posible ya que los puntos de datos que ingresan a la ecuación de SVM solo llegan como productos internos.

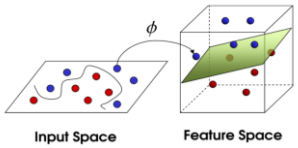


Figura 3.4 Datos inseparables linealmente proyectados en un espacio superior.

2.13 Naive Bayes

El clasificador *Naive Bayes* es un clasificador de aprendizaje automático popular, debido a su simplicidad y, a menudo, se utiliza como línea de base para la categorización de texto. El clasificador hace el supuesto "ingenuo" de que la independencia se produce entre todas las características. El clasificador se aplica a partir del teorema de Bayes. *Naive Bayes* se usa a menudo en el filtrado de spam y la clasificación de documentos. Un inconveniente de *Naive Bayes* es que es un mal estimador, por lo que los resultados de probabilidad no son muy confiables. Para los casos en que las características son discretas, como en la documentación de texto, las multinomiales y Bernoulli como en la ecuación (13) son distribuciones populares.

$$p(C_k|x) = \frac{p(C_k) \cdot p(x|C_k)}{p(x)} \tag{13}$$

2.14 Evaluación del modelo

Es de gran importancia estimar qué tan bien funciona el modelo. Para la estimación de errores, es común dividir todo el conjunto de datos en tres conjuntos, a saber: conjunto de trenes, conjunto de validación y conjunto de pruebas (Hastie, 2010). El conjunto de trenes se utiliza para entrenar el modelo, la capacitación se realiza utilizando los datos etiquetados para que el modelo aprenda cómo se conectan la entrada y la salida esperada. El conjunto de validación se utiliza para encontrar el mejor modelo ajustado al conjunto de tren. Los parámetros para el modelo también se ajustan utilizando el conjunto de validación. El modelo que obtiene el mejor resultado en los datos de validación se usa en el conjunto de pruebas para ver cómo se desempeña en datos imparciales. Si hay mucha información disponible, es preferible dividir los datos en tres conjuntos igualmente grandes, de lo contrario, una división normal es asignar el 50% para el entrenamiento del 25% para validación y prueba (Hastie, 2010).

Sin embargo, al dividir todo el conjunto en tres conjuntos separados, disminuimos drásticamente los ejemplos de números para la capacitación. Si el número de instancias de entrenamiento es bajo, podría afectar negativamente el resultado de todo el modelo. La validación cruzada (*CV*) es una solución a ese problema. Un conjunto aún debe reservarse para la prueba, pero el *CV* elimina la necesidad de un conjunto de validación. Un enfoque común es *K-fold CV*, donde los datos se dividen en *K* número de pliegues. Luego, el modelo se entrena en pliegues (*K* - 1) y se prueba en el resto [Hastie et al., 2010]. Esto se hace para *k* = 1, 2, ..., *K*. Finalmente, combinamos estos resultados y los usamos como la estimación de validación.

Validación cruzada 5 veces. (k - 1) los pliegues se usan para entrenamiento y el resto se usa para pruebas como se muestra en la figura 3.5

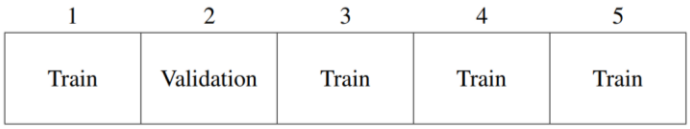


Figura 3.5 Validación cruzada 5 veces

Un problema común en el aprendizaje automático es el sobreajuste. En esta tesis, el sobreajuste se adapta cuando el modelo se adapta a las peculiaridades de los datos de entrenamiento. El problema con esto es que el ruido que se produce en los datos de entrenamiento también se adaptará al modelo, lo que hará que el modelo tenga un rendimiento deficiente en los datos nuevos que no se ven (Domingos, 2012). El modelo solo debe ajustarse a los patrones que discriminan a toda la población, no solo la parte sobre la que se está capacitando al modelo.

Estimación de error

Algunas estimaciones interesantes en problemas de clasificación son la precisión, la exactitud y el recuerdo. En términos estadísticos, recuerde la relación entre el número de verdaderos positivos y la suma de verdaderos positivos y negativos positivos tal cual muestra la ecuación (14).

$$\text{recall} = \frac{t_p}{t_p + f_n} \quad (14)$$

Donde t_p es el número de verdaderos positivos y f_n el número de falsos negativos. La precisión es la relación entre el verdadero positivo y la suma de los verdaderos positivos y los falsos positivos como se muestra en (15).

$$\text{precision} = \frac{t_p}{t_p + f_p} \quad (15)$$

Donde t_p es el número de verdaderos positivos y t_p el número de falsos positivos [Pedregosa et al., 2011]. Finalmente, el puntaje de F_1 es una medida de precisión que puede interpretarse como media ponderada de la precisión y el recuerdo como la muestra la ecuación (16).

$$\text{F1-score} = \frac{2t_p}{2t_p + f_p + f_n}. \quad (16)$$

2.15 Método

Describiremos cómo se procesan los datos a partir de texto en bruto. También presentaremos las características utilizadas para este modelo y motivaremos brevemente por qué se seleccionan. Además, describiremos las herramientas utilizadas para este estudio y el propósito de usarlas.

2.16 Preprocesamiento de datos

Los datos de texto rara vez se encuentran en un formulario susceptible de clasificación. Antes de extraer cualquier función, procesamos el texto para que se ajuste mejor a nuestro propósito, pero también para reducir la cantidad de ruido en nuestro texto, como los caracteres ilegales y las faltas de ortografía. Tecnologías - Python. Este proyecto necesita un lenguaje de programación que sea conveniente de usar para métodos particulares de aprendizaje automático. Python es un lenguaje de programación versátil que admite tanto la programación orientada a objetos como la programación funcional. Proporciona un amplio espectro de bibliotecas que admite muchas tareas de programación diferentes y también es de código abierto. Las bibliotecas son fáciles de instalar y están bien documentadas. Una ventaja más de Python es que es fácil de leer debido al uso de sangrías y nuevas líneas en lugar de más corchetes. Las bibliotecas que se describen a continuación son algunas de las más comunes que se han utilizado en el proyecto. Puede encontrar más información sobre Python y las bibliotecas en la página web de Pythons.

• Numpy da soporte para funciones matemáticas de alto nivel, matrices grandes y matrices. Cuando se usan capacidades de álgebra lineal y números aleatorios, Numpy es muy útil. Numpy también tiene utilidades para integrar código Fortran y C / C ++.

• Scikit-learn es una biblioteca de Machine Learning en la que Google comenzó a trabajar en 2007. El primer lanzamiento público fue en 2010 y se ha actualizado continuamente. Está construido sobre los paquetes matemáticos numpy y scipy. Entre muchas otras características, incluye una variedad de algoritmos de regresión, clasificación y

agrupamiento, como Bayes ingenuos, bosques aleatorios, máquinas de vectores de soporte, algoritmos de impulso, etc. (Pedregosa, 2011)

• NLTK es una abreviatura de Natural Language Toolkit, es una biblioteca de Python que es útil cuando se trabaja con datos en lenguaje humano. La biblioteca consta de más de 50 activos léxicos y corpora diferentes, como WordNet, análisis y etiquetado.

• SciPy es una biblioteca que contiene muchas rutinas para la integración y optimización numérica.

• Textblob es una biblioteca que se utiliza cuando se trabaja con datos textuales. Proporciona una API para procesamiento de texto, minería de texto y análisis de texto. Algunas herramientas específicas en estas áreas que hemos utilizado son el análisis de opiniones, la clasificación y la traducción.

2.17 COLAB

"Google Colab" es una herramienta para escribir y ejecutar código Python en la nube.de Google. También es posible incluir texto enriquecido, "links" e imágenes. En caso de necesitar altas prestaciones de cómputo, el entorno permite configurar algunas propiedades del equipo sobre el que se ejecuta el código. En definitiva, el uso de "Google Colab" permite disponer de un entorno para llevar a cabo tareas que serían difíciles de realizar en un equipo personal. Por otro lado, siguiendo la idea de "Drive", "Google Colab" brinda la opción de compartir los códigos realizados lo que es ideal para trabajos en equipo. (Baume, 2021).

Capítulo 3: Estado del arte

En 2003 Kushal Dave et al., publican “Mining the peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews”, artículo en el cual exponen la riqueza que hay en internet sobre opiniones de productos y proponen un sistema que selecciona y sintetiza las opiniones. (Dave, 2003).

En 2005, Jonathan Read en su publicación “Using Emoticons to reduce Dependency in Machine Learning Techniques for Sentiment Classification” muestra que los métodos propuestos hasta ese entonces para la clasificación de sentimientos habían demostrado resolver con éxito el problema, pero con la condición de que tenía que haber contexto entre los datos de entrenamiento y los de prueba, por ejemplo, un sistema entrenado con datos de reseñas de películas no obtendría tan buenos resultados si se prueba con reseñas de automóviles. También se propone que existe una dependencia en el dominio y en el tiempo, en otras palabras, un clasificador entrenado con opiniones de productos no es efectivo para evaluar el sentimiento de artículos de noticias, además este sería efectivo solo por un lapso de tiempo (Read, 2005).

Alec Go et al., en su artículo “Twitter Sentiment Classification using Distant Supervision” publicado en 2009, abordan la clasificación de sentimientos, positivos y negativos sobre textos cortos de Twitter, con el objetivo de hacer que su aproximación sea útil para consumidores que buscan opiniones antes de adquirir un producto, o para empresas que buscan monitorear las opiniones acerca de sus marcas. El equipo propone un sistema usando machine learning para clasificar el sentimiento en los mensajes de Twitter, esto mediante el uso de supervisión distante sobre los términos de consulta; el corpus de entrenamiento que usan consiste en mensajes de Twitter que contienen emoticonos, se basan en mensajes de este ese tipo para determinar la polaridad de las palabras y entrenar al corpus de entrenamiento. Establecen como base una categorización simple basada en el número existencias de palabras, positivas y negativas clasificadas por Twittratr (sistema clasificador de tweets), para determinar la polaridad. Hacen un comparativo de resultados con clasificadores Naive Bayes, entropía máxima y Maquinas de Soporte Vectorial; consideran unigramas, bigramas y etiquetado de partes de oración; los mejores resultados fueron obtenidos con los clasificadores

Naive Bayes y Entropía ‘Máxima usando unigramas y bigramas en conjunto como características para el entrenamiento (Go, 2009).

Alexander Pak y Patrick Paroubek, en su publicación “Twitter as a Corpus for Sentiment Analysis and Opinion Mining’ publicada en 2010, abordan de nuevo el tema de la clasificación de tweets, recolectando mensajes de Twitter para usarlos como corpus. Construyen una herramienta que clasifica los mensajes en positivos, negativos y neutros, usando los algoritmos Naive Bayes y SVM (Pak, 2010, may).

En 2013, Grigori Sidorov et al., publicaron “Empirical Study of Machine Learning for Opinion Mining in Tweets’, donde exponen los resultados obtenidos de una serie de experimentos en los que demuestran como diferentes aspectos modifican la precisión de los algoritmos de aprendizaje máquina. Para la experimentación usaron un corpus de 32,000 tweets en español de los cuales 8,000 fueron clasificados manualmente para el entrenamiento en las categorías: positive, negative, neutral o noticioso. Los algoritmos de aprendizaje que implementaron fueron: Naive Bayes, Decision Tree y ‘Support Vector Machines. Los factores que modificaron fueron: el tamaño de los n-gramas, el tamaño del corpus, el número de clases de sentimiento, el balance de los corpus y probaron varios dominios, con la finalidad de encontrar el mejor balance de configuraciones para la clasificación de tweets en español. (Sidorov, 2013).

3.1 Redes Transformer

Desde hace aproximadamente tres años se viene gestando otra gran revolución en la inteligencia artificial, en febrero de 2019 Open creó el modelo GPT2 capaz de sintetizar texto muy similar al que generaría un ser humano, también se lanzó GPT3 un modelo diez veces más grande que su antecesor y mucho más poderoso, alimentado con tan solo una corta frase de inicio esta inteligencia artificial es capaz de escribir una historia completa de manera bastante convincente y en abril 2020 Facebook liberó el código fuente de Blenderbot, un chat bot que se percibe más humano en una conversación que una persona real y a comienzos de abril de 2020 también presentó Transcoder, un modelo capaz de tomar código en python y convertirlo en código C++ y esto a pesar de que nunca fue entrenado para esta tarea. Detrás

de esta nueva revolución están las *transformer networks*, un nuevo tipo de red neuronal que es el tema de esta sección, esta arquitectura promete convertirse en un hito en el ya vertiginoso avance de la inteligencia artificial.

Las redes *Transformers* nacieron en el contexto del procesamiento del lenguaje natural, un área de la inteligencia artificial que busca que los computadores logren interpretar y manipular el lenguaje humano, hasta hace muy poco las redes recurrentes dominaban esta área pues el lenguaje humano es precisamente una secuencia de palabras y estas redes se especializan en procesar secuencias, pero las redes recurrentes tienen un gran inconveniente, tienen una memoria de corto plazo incluso si son redes del LSTM por ejemplo podemos observar la figura 3.1.1:

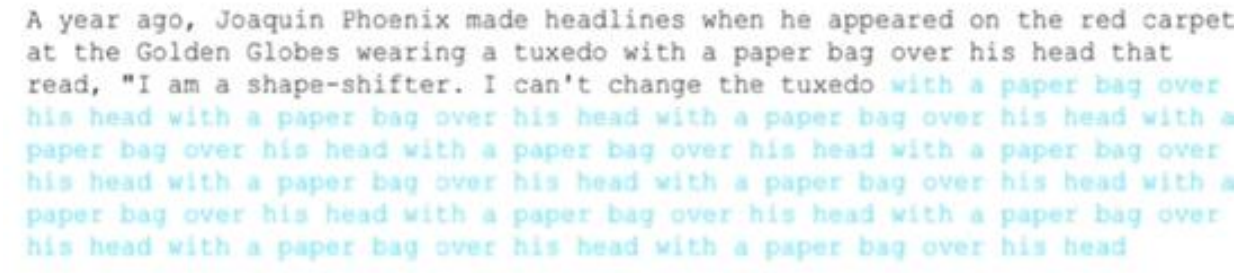


Figura 3.1.1 Historia creada de forma aleatoria

Para la generación de texto la red podría producir frases relativamente cortas y bastante coherentes, pero cuando la secuencia generada es extensa la red no está en capacidad de tener como referencia el texto que apareció al inicio y así el texto general no será del todo coherente además el texto es procesado de manera secuencial, es decir, palabra por palabra. Por el contrario, las redes *Transformers* tienen una memoria de mucho más largo plazo pues logran analizar secuencias muy extensas usando un mecanismo que se llama atención, adicionalmente logran procesar toda la secuencia en paralelo y no en serie como ocurre con las redes recurrentes. A continuación, se observa en la figura 3.1.2 una historia creada de forma aleatoria respetando el contexto de las primeras palabras.

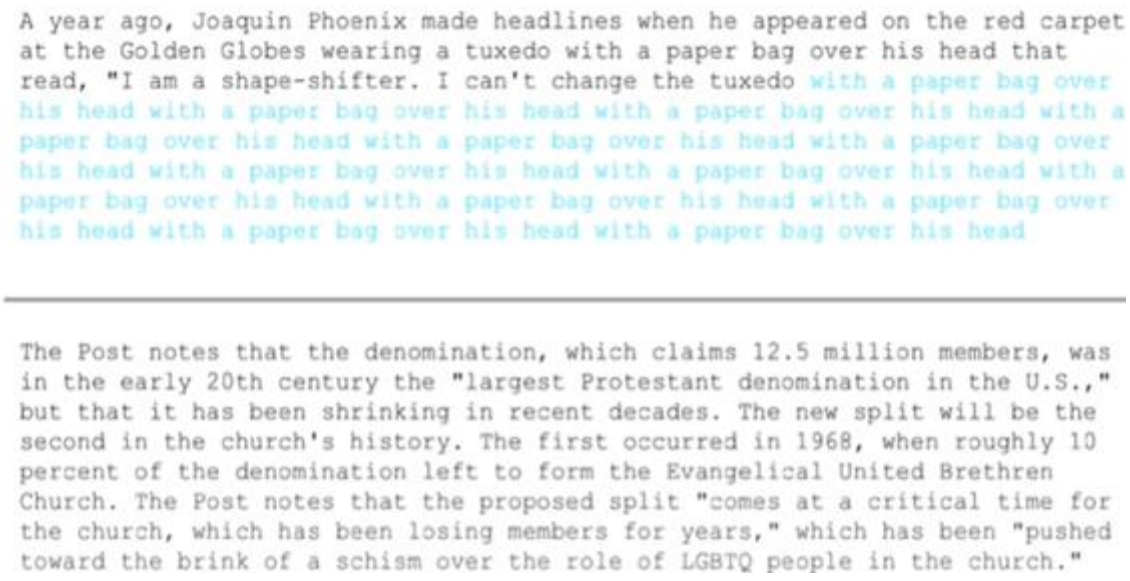


Figura 3.1.2 Historia larga creada de forma aleatoria

Estas redes *Transformers* son descritas en el artículo de 2017 "*attention it's all you need*" y nacieron inicialmente como una alternativa al problema de la traducción de texto de un idioma a otro.

En esta red de la totalidad de la secuencia de entrada es procesada en paralelo por la red esta secuencia es inicialmente convertida en una representación numérica usando un *MBEDDING*, después se añade una codificación de posición y los vectores resultantes ingresan a la etapa de codificación que se encarga de extraer la información más relevante de la secuencia en su idioma original la salida a esta etapa se conecta al decodificador que toma esta información para generar secuencialmente el texto traducido al segundo idioma aunque

tiene muchos elementos realmente sencillo entender cómo funciona, veamos entonces en detalle cada módulo de esta red.

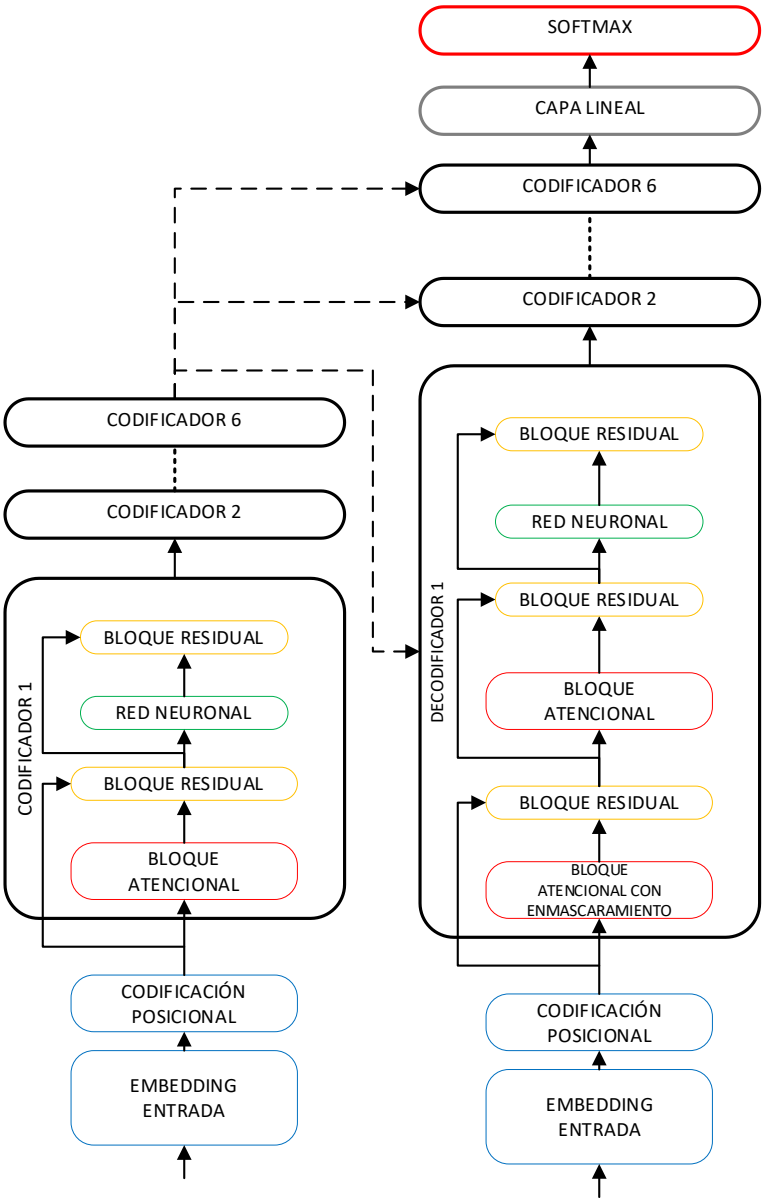


Figura 3.1.2 Estructura básica de una red Transformer

EMBEDDING ENTRADA

Como se observa en la figura 3.1.2 primero está el bloque en *EMBEDDING* que es simplemente un algoritmo que convierte el texto en una serie de vectores y tokens es decir en una representación numérica que puede ser comprendida.

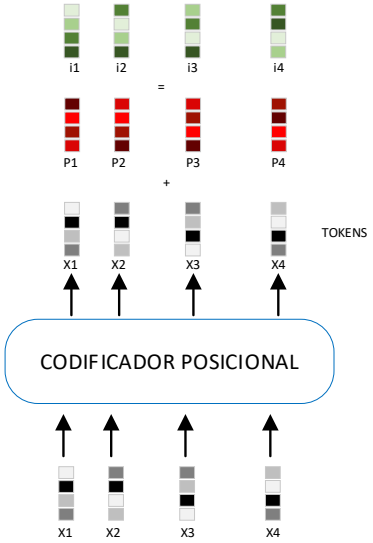


Figura 3.1.4 Tokenizacion de posición

Por paralelo es necesario indicarle a la red el orden en el que se encuentran las palabras dentro del texto, esto se logra con el codificador de posición como se ilustra en la figura 3.1.4, este codificador genera una serie de vectores que se sumarán a los tokens y que indican la posición relativa de cada token dentro de la secuencia, para esto se usan funciones senoidales para las posiciones pares y cosenoidales para las impares con lo que cada vector generado tendrá un patrón numérico único con la información de la posición.

Codificadores

Ahora viene el bloque de codificación que contiene 6 codificadores todos con una estructura idéntica a la de la figura 3.1.5, analicemos el detalle uno de estos codificadores.

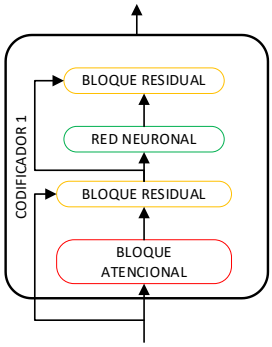


Figura 3.1.5 Estructura de un codificador

Cada codificador contiene cuatro elementos: un bloque atencional, un bloque de conexión residual, una red neuronal y otro bloque de conexión residual. El bloque atencional

es tal vez el más importante de toda la red pues se encarga de analizar la totalidad de la secuencia de entrada recordemos que la red la procesa de manera simultánea y debe encontrar relaciones entre varias palabras de esta secuencia.

Figura 3.1.6 Frase dividida en dos

Por ejemplo, si el texto de entrada es "*I love Italian food*" como se muestra en la figura 3.1.6 podemos ver que hay al menos dos posibles asociaciones entre palabras el verbo "Love" y el sujeto "I" y el sustantivo "*food*" asociado al adjetivo "*Italian*" .

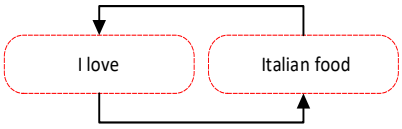


Figura 3.1.7 asociación de frases

Pero además entre estas dos "*I love*" e " *Italian food*" como se muestra en la figura 3.1.7 también hay una asociación así que lo que hace el bloque atencional es expresar numéricamente las relaciones que existen a diferentes niveles dentro de la secuencia y luego codifica cada una de ellas con esta información del contexto indicando así cuáles son los elementos del texto a los que se debe prestar más atención al momento de hacer la traducción . Esta es precisamente la manera como las redes Transformers comprenden este contexto para codificar adecuadamente cada palabra.

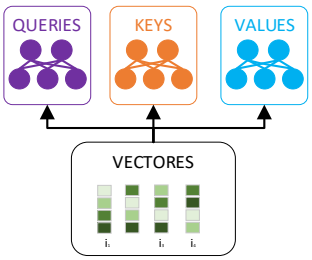


Figura 3.1.8 Redes neuronales de un codificador

Para lograr esto en primer lugar los tokens se llevan simultáneamente a tres pequeñas redes neuronales entrenadas para calcular los vectores QUERIES, KEYS y VALUE tal cual representa la figura 3.1.8, estos vectores son simplemente tres representaciones alternativas de los tokens originales.

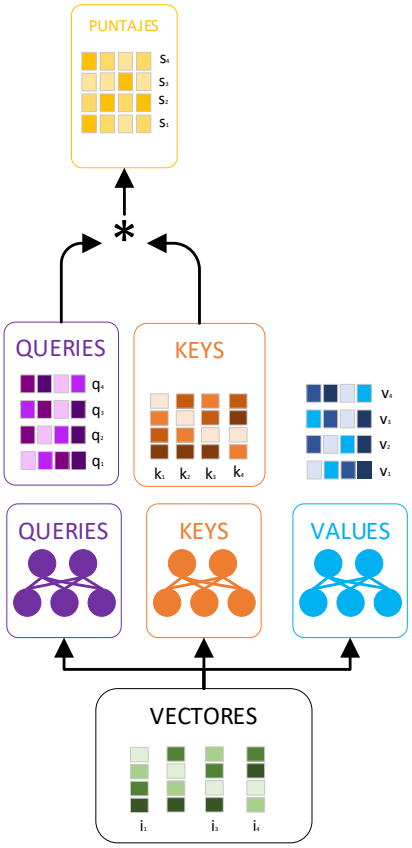


Figura 3.1.9 Comparaciones entre salidas

Después se toma el QUERIE de cada token y se compara con cada uno de los KEYS existentes como se muestra en la figura 3.1.9, esta comparación es simplemente una multiplicación de vectores y con esto se obtendrá un puntaje que mide el grado de asociación entre pares de palabras.

Así para el caso de la frase que queremos traducir si analizamos la palabra "*Italian*" los puntajes obtenidos indican que al codificar este token se le debería prestar más atención a la propia palabra "*Italian*" seguida por la palabra "*food*" y se debería enfocar menos en las palabras "*love*" y "*I*" que tienen los menores puntajes, la idea es ahora usar estos puntajes para ponderar cada uno de los vectores VALUE indicando así la importancia de cada palabra al momento de la codificación de los tokens para poder hacer esto se deben escalar los puntajes dividiendo los primero entre el tamaño de cada vector y luego llevándolos a una función SOFTMAX como se representa en la figura 3.1.12.

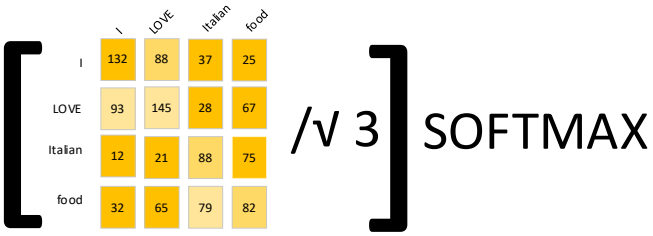


Figura 3.1.12 Aplicación de capa SOFTMAX a matriz de puntajes

Esta función permite simplemente representar cada puntaje como una probabilidad entre 0 y 1 como se muestra en la figura 3.1.13.

Figura 3.1.13 Matriz de puntajes normalizada

Un valor cercano a 1 indica que la red debe prestarle más atención a ese token en particular y un valor cercano 0 indica que la palabra no es muy relevante.

Finalmente se debe condensar toda esta información resultante de la comparación en un solo vector por cada token así que tomamos la matriz de puntajes que acabamos de obtener y la multiplicamos por la matriz de VALUES el resultado serán cuatro nuevos tokens que contendrán la codificación de la información de contexto más relevante para cada palabra de

la secuencia. Así que en resumen el bloque atencional toma los tokens iniciales y codifica los tokens resultantes, los elementos de la secuencia a los que se debe dar más relevancia, sin embargo, recordemos que nuestra frase original encontramos además de asociaciones entre palabras asociaciones entre frases para traducir la porción “Italian food” se necesita prestar atención a “I love” así que un solo bloque atencional no es suficiente.

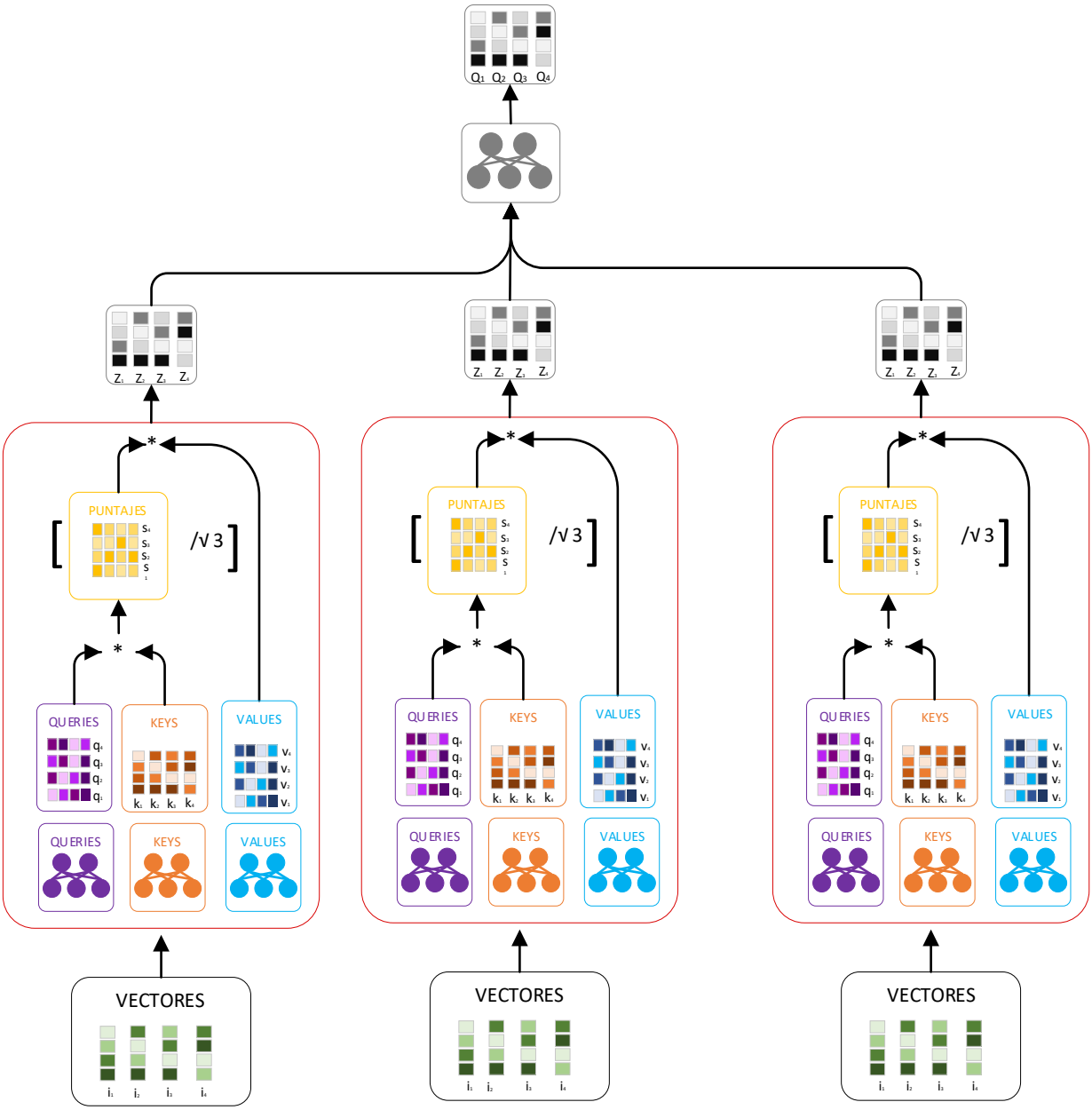


Figura 3.1.1 múltiples bloques atencionales

Como se muestra en la figura 3.1.1 al usar múltiples bloques atencionales es posible detectar y codificar asociaciones entre palabras y grupos de palabras a diferentes niveles. Las salidas de estos bloques se combinan en una última red neuronal que condensa toda la información resultante con un único vector por cada token de entrada y listo, este es el bloque atencional. Ya tenemos el 90% de la red *Transformer* lista, los demás bloques son menos complicados así que los revisaremos rápidamente.

Bloque residual

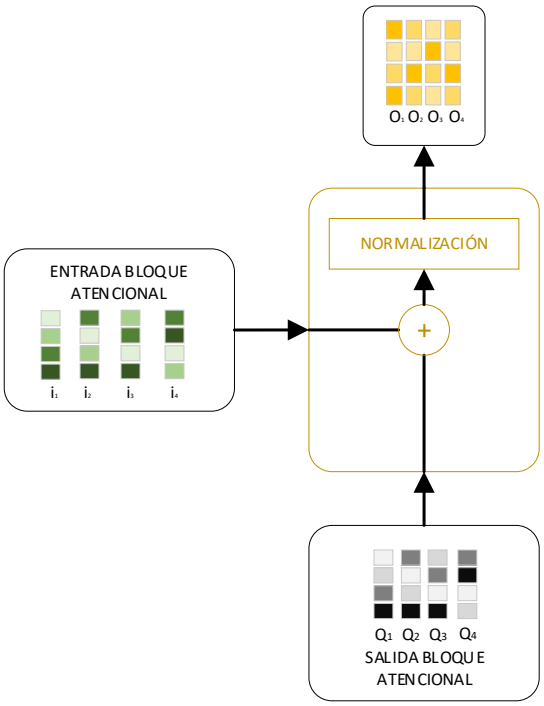


Figura 3.1.16 Estructura del bloque residual

Como se observa en la figura 3.1.16 a este bloque se llevan tanto la entrada como la salida del bloque atencional y esto se hace porque la red es muy profunda y tiene muchas capas, si tan solo se enviara solo el dato de salida la información progresivamente se degradará y esto dificultará el entrenamiento y desempeño de la red. Esta etapa toma los dos datos, los suma y luego los normaliza para que tengan la escala adecuada requerida por el siguiente bloque.

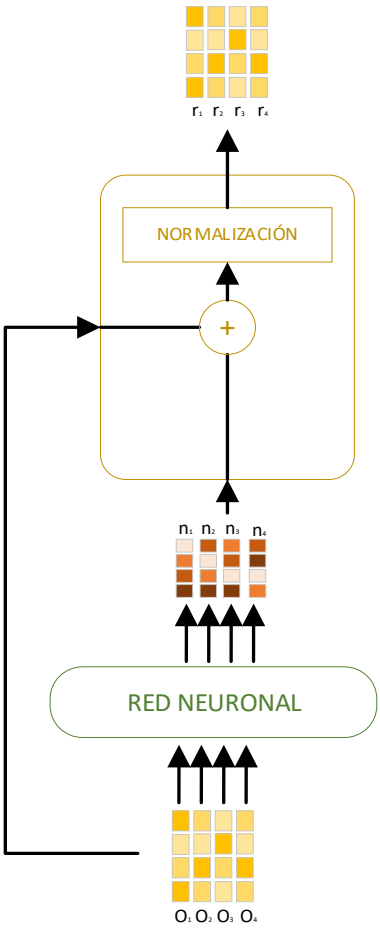


Figura 3.1.17 Estructura de normalización

Después como se muestra en la figura 3.1.17 tenemos una red neuronal seguida también por un bloque residual, la red neuronal procesa en paralelo todos los vectores de la secuencia tomando la información atencional de las capas anteriores y consolidándola en una única representación, la entrada y la salida de esta red neuronal son luego llevadas a un bloque residual que tiene exactamente las mismas características del bloque anterior, una suma seguida por una normalización de los datos y con esto está listo el primer codificador.

Así que en resumen este bloque toma los tokens de entrada, los procesa en paralelo y entrega a la salida una representación que contiene información atencional sobre las diferentes relaciones entre palabras o grupos de palabras de la secuencia y este proceso se repite para los codificadores restantes que son idénticos en estructura al codificador que acabamos de analizar.

Ahora nos enfocamos en el segundo bloque importante de la red *Transformer* que se encarga de hacer la traducción , como se puede observar en la figura 3.1.1 en primer lugar tenemos los bloques de EMBEDDING de salida y un codificador posicional que cumplen exactamente la misma función de los bloques que vimos en la etapa de codificación, luego viene el decodificador que es muy similar al bloque de codificación, en total cuenta con 6 decodificadores cada uno de ellos conectado al codificador, lo que permite conocer la información atencional de la entrada en el idioma original para poder realizar la traducción, cada decodificador es similar a los bloques de codificación que vimos anteriormente, cuenta con bloques atencionales residuales y redes neuronales que tienen la misma estructura de los codificadores, sin embargo tienen un bloque atencional adicional, luego viene una capa lineal que junto con la capa SOFTMAX permite generar una a una las palabras de la secuencia de salida.

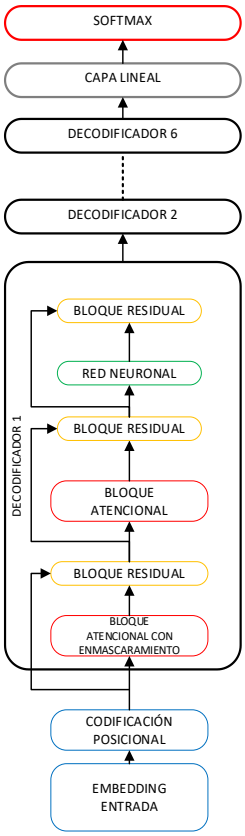


Figura 3.1.1 Bloque traductor

La traducción comienza con la palabra clave “START” la cual es codificada con el EMBEDDING y posicionalmente al ingresar al primer decodificador es procesada por el bloque atencional de enmascaramiento como se muestra en la figura 3.1.19.

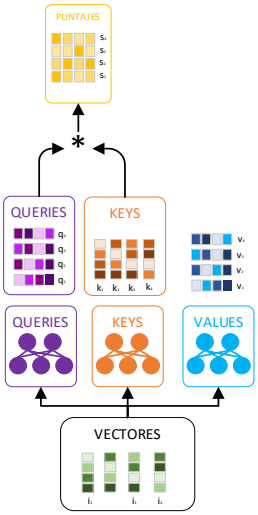


Figura 3.1.19 comparadores de salidas

Este bloque de la figura 3.1.1 es prácticamente idéntico al bloque atencional visto anteriormente, codifica la relación entre diferentes elementos de la secuencia de salida usando los QUERIES, KEYS y VALUES vistos anteriormente, pero con una diferencia importante.

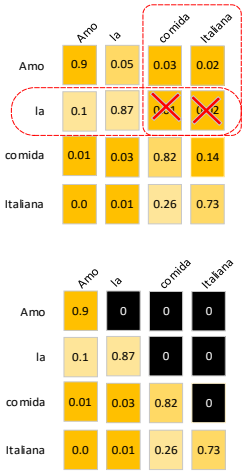


Figura 3.1.20 enmascaramiento

Como se está generando cada palabra de manera secuencial, una a una, el decodificador debe prestar atención únicamente a la palabra generada actualmente y a las anteriores, no a las palabras futuras, por ejemplo en la figura 3.1.20, si en la secuencia traducida nos ubicamos en la palabra “la” él decodificador debería tener acceso a esta palabra y a la palabra “amo” pero no a las palabras que aparecerán posteriormente en la secuencia:

“comida” e italiana , así que para evitar esto se agrega un bloque que enmascara, es decir que simplemente hace 0 las palabras a las que durante la decodificación no se debe prestar atención, al igual que con el codificador en este caso también se emplean múltiples bloques atencionales para detectar relaciones en diferentes niveles, todos los bloques residuales así como la red neuronal de este decodificador funcionan de forma idéntica a como ocurriría en los codificadores .

Bloque atencional

Ahora nos ubicamos en el bloque atencional, que en este caso tiene la misma estructura, pero en funcionamiento es ligeramente diferente al del codificador.

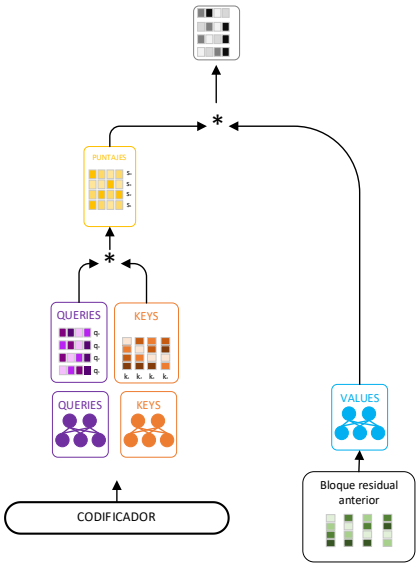


Figura 3.1.21 Estructura del bloque atencional

Este bloque que se representa en la figura 3.1.21 enfoca su atención tanto en la secuencia original como en la de salida y para ello toma la salida del codificador y la lleva a las redes QUERIES y KEYS mientras que la red VALUES tiene como entrada el dato proveniente del bloque residual anterior, es de esta manera como el codificador le indica al decodificador a qué elementos debe prestar más atención al momento de generar la secuencia de salida de nuevo se usan múltiples bloques atencionales de manera simultánea para codificar asociaciones a diferentes niveles, con esto ya tenemos el primer decodificador. Este bloque

se replica un total de 6 veces y al final genera un vector con cantidades numéricas y lo único que faltaría seria convertirlo en una palabra, para eso se usa en primer lugar la capa lineal y posteriormente una capa SOFTMAX.

Capa lineal y Softmax

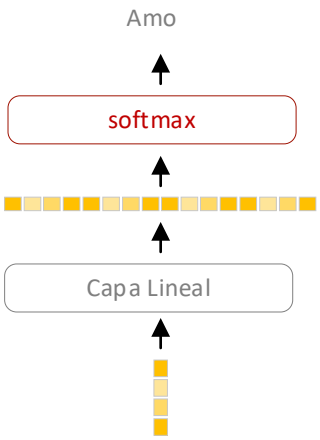


Figura 3.1.22 Capa lineal y SOFTMAX

Como se muestra en la figura 3.1.22 esta sección consta simplemente una red neuronal que toma el vector producido por el decodificador y lo transforma en un vector mucho más grande, por ejemplo si el traductor aprende 10.000 palabras es decir el tamaño del vocabulario entonces el vector de salida de la capa lineal tendrá precisamente 10.000 elementos la capa SOFTMAX toma cada elemento de este vector y lo convierte en una probabilidad todos con valores positivos entre 0 y 1 la posición con la probabilidad más alta será seleccionada y la palabra asociada a dicha posición será precisamente la salida del modelo en ese instante de tiempo y el proceso se repite hasta generar la totalidad de la secuencia de salidas.

Esta es la red Transformer, aunque para analizarla miramos el caso particular de la traducción realmente esta red funciona con cualquier tipo de aplicación de NLP, por ejemplo la generación de texto o la sumarización y esto funciona mucho mejor que las convencionales redes recurrentes.

3.2 BERT

El procesamiento del lenguaje natural es una de las joyas de la corona hasta el momento inalcanzable para la inteligencia artificial en el NLP se busca lograr que los humanos se puedan comunicar con los computadores usando el lenguaje natural es decir la conversación o el texto en lugar de un lenguaje de computador y aunque durante los últimos años ha habido desarrollos impresionantes sólo hasta 2018 apareció una nueva arquitectura desarrollada por Google que marcó el inicio de una nueva era en el NLP se trata de BERT, un modelo que ha sido pre entrenado con un corpus gigantesco, veremos en detalle este modelo y entenderemos por qué ha roto una barrera que permitirá tal vez más pronto de lo que pensamos que humanos y computadores logren comunicarse de manera natural así que comencemos. Para comprender qué es y la importancia de BERT en la inteligencia artificial necesitamos entender qué es el procesamiento del lenguaje natural el lenguaje natural se refiere simplemente a la forma como nosotros los humanos nos comunicamos unos con otros bien sea por escrito usando el texto o de manera oral con la voz, así que el procesamiento del lenguaje natural consiste en desarrollar algoritmos de computador que sean capaces de entender el lenguaje natural y que permitan una interacción hombre computador en ambas direcciones, pero esta no es una tarea trivial.

Si queremos que un computador entienda lo que decimos o lo que escribimos tendrá que ser capaz de ejecutar varias tareas propias del procesamiento del lenguaje natural, por ejemplo:

Inferencia: Dadas dos frases deberían ser capaz de predecir si la segunda frase guarda relación con la primera o es una contradicción o la relación es neutral, a continuación, un ejemplo en la figura 3.2.1.

Si acabas de encontrar este libro, probablemente ya es tarde. El examen fue ayer. Realmente nunca es tarde para aprender.

Figura 3.2.1 Ejemplo de inferencia

Equivalencia semántica

Determinar si dos preguntas son semánticamente equivalentes es decir si significan prácticamente lo mismo, Podemos observar un ejemplo en la figura 3.2.2

¿Cuáles son las pizzas más populares en el menú de dominos pizza? ¿Cuántas calorías tiene una pizza de Dominós pizza?

Figura 3.2.2 Equivalencia semántica

Pregunta-Respuesta

Dado pasaje y una pregunta encontrar en el texto la respuesta correspondiente como en la figura 3.2.3

“Twilight Princess” es un juego de acción y Aventura desarrollado por Nintendo para sus consolas GameCube y Wii, y que fue lanzado en Norteamerica, Europa y Japón a finales del 2006… ¿En cuales consolas se puede jugar “Twilight Princess?”

Figura 3.2.3 Pregunta respuesta

Análisis de sentimientos

Dado un texto como en la figura 3.2.4 Identificar si el sentimiento que se está expresando es positivo o negativo o neutral

El servicio fue excelente
El servicio pudo ser mejor
El servicio no fue bueno

Figura 3.2.4 Análisis de sentimientos

Aceptabilidad lingüística

Dada una frase identificar si esta es gramaticalmente correcta. Un ejemplo de una mala gramática sería el texto de la figura 3.2.5.

Me llamó la atención de que no viniera

Figura 3.2.5 Aceptabilidad lingüística

Equivalencia Semántica

Determinar el grado de similitud entre dos frases, por ejemplo, en la figura 3.2.6

-Varios chicos juegan futbol en la playa
-Un grupo de hombres está jugando futbol en la playa
Equivalencia 5.0

-El chef está picando cebollas
-El chef está picando tofu
Equivalencia 1.8

Figura 3.2.6 Equivalencia semántica

Respuesta correcta

Dada una frase determinar su continuación entre cuatro posibles opciones como en la figura 3.2.7

La mujer subió al escenario, se sentó al frente del piano y: Vio a su hermana jugar a las muñecas Comenzó a sonreírle al público -Se sentó con el público a disfrutar la obra - Comenzó a tocar una pieza musical

Figura 3.2.7 Respuesta correcta

El gran problema es que para ninguna de estas tareas se cuentan con un set de entrenamiento lo suficientemente grande y esto dificulta el desarrollo de modelos robustos que resuelvan precisamente estas tareas, pues bien, acá es donde entra en escena BERT (Bidirectional Encoder Representations from Transformers), el modelo

Propuesto por Google en 2018 y que resuelve el problema de una manera muy ingeniosa. BERT,es un codificador que obtiene representaciones bidireccionales a partir de redes Transformer pero esto no nos dice mucho así que te desglosemos su significado.

Básicamente los autores de proyecto proponen lo siguiente: ¿por qué no en lugar de crear un modelo que resuelva cada tarea individualmente se entrena un modelo base que aprende a interpretar el lenguaje en general y luego a este modelo ya entrenado le agregamos unas capas adicionales que permitan que el modelo se especialice en una tarea en particular?

Este modelo base puede ser pre entrenado con sets de datos inmensos y luego de agregar las capas adicionales se puede re entrenar el modelo con sets más pequeños para que realice tareas más específicas , pues bien, resulta que esta idea funciona, así que BERT se ha convertido en el primer modelo de inteligencia artificial multipropósito capaz de interpretar el lenguaje humano en diferentes aplicaciones y por eso se ha convertido en un hito en el NLP, habiendo entendido esto veamos en detalle en qué consiste este modelo.

BERT nace precisamente de las redes Transformer, un nuevo tipo de arquitectura creada en 2017 y que ha sido el precursor de toda esta revolución del NLP, la red transformer

procesa el texto en dos fases una para la codificación encargada de procesar el texto de entrada y decodificar lo numéricamente extrayendo su información más relevante y una fase de decodificación que se encarga de generar una nueva secuencia de texto que es útil, por ejemplo en la figura 3.2.7 cuando se hace la traducción de un idioma a otro, en ambos casos cada bloque está conformado por múltiples codificadores y decodificadores.

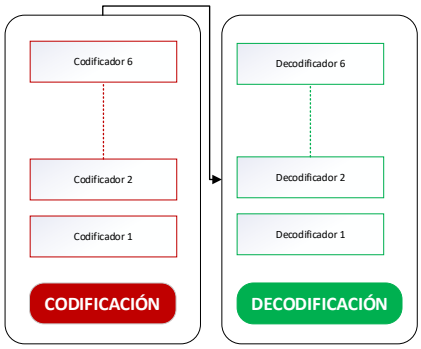


Figura 3.2.7 Codificación y decodificación

Con BERT lo que nos interesa es tener un modelo capaz de codificar el texto obteniendo así una representación numérica que permita su correcta interpretación, así que BERT es prácticamente el resultado de tomar la red transformer y quedarnos únicamente con el bloque de codificación.

En ambos casos es necesario adecuar el texto de entrada para que la red pueda procesarlo, en las tareas que vemos anteriormente el dato de entrada siempre está conformado por dos elementos como en la figura 3.2.10:

Figura 3.2.10 Dato de entrada

Bien sea dos frases un párrafo y una pregunta o dos preguntas así que el texto de entrada se codifica como dos frases, al inicio se incluye siempre un token de clasificación y para separar una frase de otra se incluye el token de separación tal cual se observa en el ejemplo de la figura 3.2.11.

Figura 3.2.11 Datos de entrada con tokens de separación e inicio

Después por cada token se obtienen tres representaciones; un embeding que representa el token como un vector y una codificación posicional como las usadas en la red transformer original recordemos que la codificación posicional es necesaria para indicarle al bloque de codificación la posición relativa de cada palabra dentro de la frase pues todas las palabras son procesadas de manera simultánea por la red. A estas dos modificaciones se añade un embeding que indica a que segmento pertenece la frase (CLS1 y SEP), la primera frase (CLS1) antes del separador se codifica con un embeding diferente al de la segunda frase (SEP) finalmente se suma estas tres representaciones (EMBEDING, POSICIÓN y SEGMENTO) y los vectores resultantes son procesados por BERT como se ve en la figura 3.2.12 que ilustra de manera grafica la representación de todos los enconder.

Figura 3.2.12 Representación de endoder.

El entrenamiento de BERT se hace en dos fases, en la primera fase es pre entrenado usando dos sets gigantescos wikipedia y google books que en conjunto contienen más de tres mil millones de palabras, con este entrenamiento BERT aprende a analizar el texto de manera bidireccional es decir que aprende a codificar cada palabra teniendo en cuenta todo su contexto tanto lo que está a su derecha como a su izquierda como en el ejemplo que se en la figura 3.2.13.

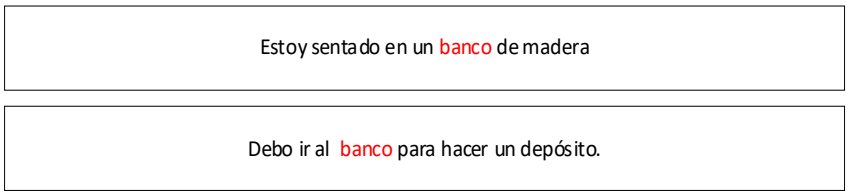


Figura 3.2.13 Ejemplo de bidireccionalidad de palabras

Una misma palabra puede tener dos significados distintos dependiendo de su contexto, con esta direccionalidad es posible codificar de manera precisa su significado, para lograr esto primero se enseña al modelo a completar una palabra faltante en una frase, esta palabra faltante puede estar en cualquier ubicación y con esto se está forzando el modelo a que aprenda a analizar el texto de manera bidireccional o aprendiendo así a comprender el lenguaje tal como lo hacemos los seres humanos, esto se puede ver en la figura 3.2.14.

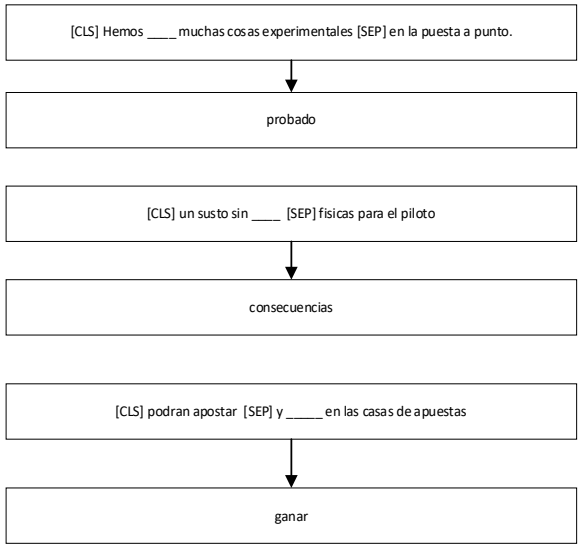


Figura 3.2.14 Ejercicios de bidireccionalidad

Se continúa con el pre entrenamiento enseñando el modelo a predecir la continuación de una frase esta tarea es sencilla dadas dos frases A y B es B realmente la frase que viene después de A o es simplemente una frase aleatoria para que BERT aprenda a identificar estas diferencias durante el pre entrenamiento el 50% de las veces la frase B es efectivamente la que sigue la frase A y el 50 por ciento de las veces restantes no lo es y listo con estas dos tareas ya se tendrá un modelo bastante robusto que es capaz de representar de manera muy completa relaciones bidireccionales entre palabras y frases como en la figura 3.2.15

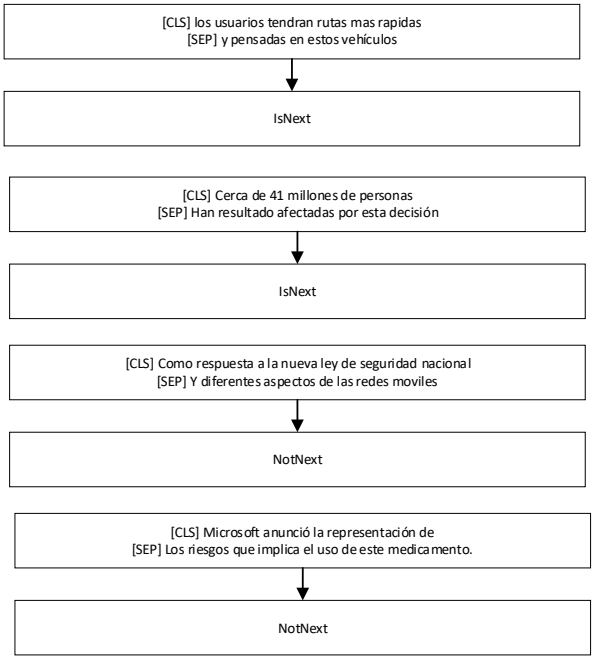


Figura 3.2.15 Bidireccionalidad de frases

Culminado el preentrenamiento resulta muy sencillo afinar el modelo para tareas específicas. Al modelo pre entrenado se agregan dos pequeñas capas, una red neuronal y una capa SOFTMAX, luego se presentan las frases de la tarea específica y se reentrena el modelo de extremo a extremo con la única diferencia de que esta fase requiere menos tiempo pues el modelo ya ha sido pre entrenado en la fase anterior cuando se tienen tareas como en la figura 3.2.16.

Inferencia: Dadas dos frases, se debería ser capaz de predecir si la segunda frase guarda relación con la primera o es una contradicción o la relación es neutral.

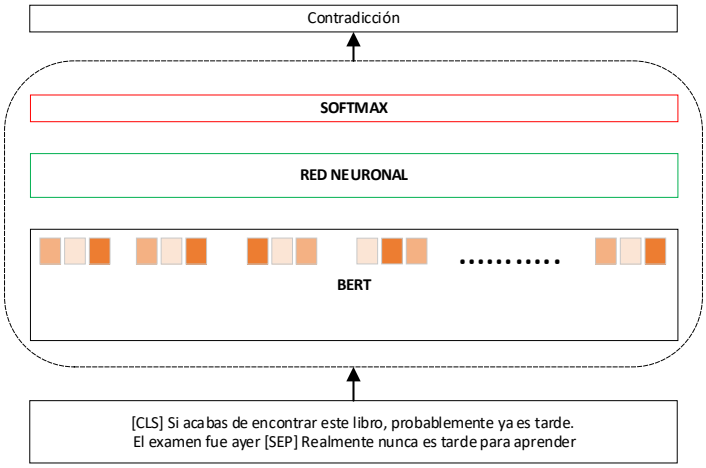


Figura 3.2.16 Ejemplo de inferencia

Equivalencia semántica: Determinar si dos preguntas son semánticamente equivalentes, es decir, si significan prácticamente lo mismo, este entrenamiento se puede observar en la figura 3.2.17.

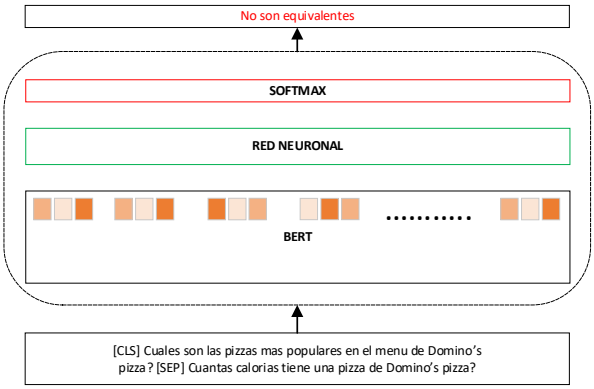


Figura 3.2.17 Equivalencia semántica

Análisis de sentimientos: Dado un texto, identificar si el sentimiento que se está expresando es positivo o negativo o neutral, se puede observar un ejemplo en la figura 3.2.18 donde identifica que el comentario en la entrada es positivo.

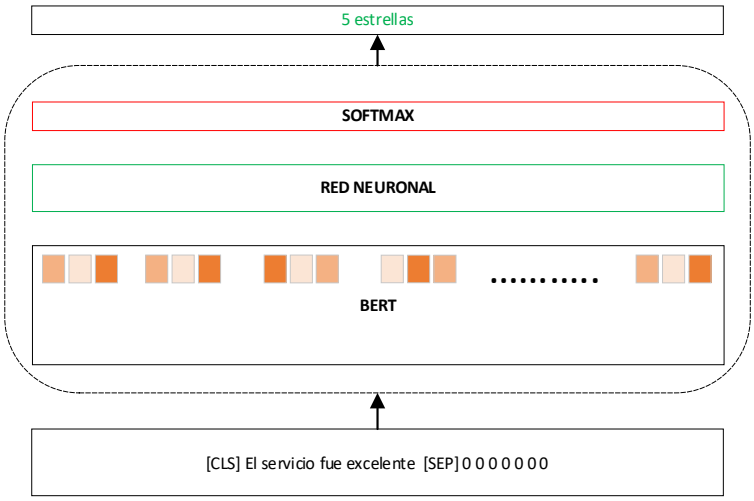


Figura 3.2.18 Análisis de sentimientos

Aceptabilidad lingüística: Dada una frase, identificar si es gramaticalmente correcta, en el ejemplo de la figura 3.2.19 se puede ver que BERT identifica que la frase de entrada no es correcta gramaticalmente hablando.

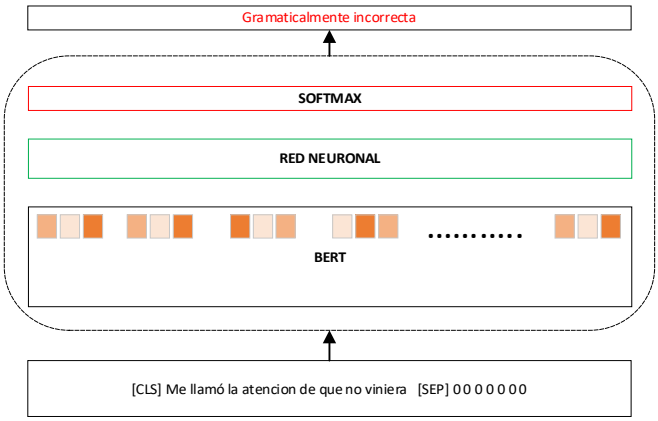


Figura 3.2.19 Aceptabilidad lingüística

Con esto se tiene simplemente un problema de clasificación, así que a la salida de BERT se analizará únicamente la representación correspondiente al token de clasificación pues al ser bidireccional ver condensa toda la información necesaria para la clasificación en este único token para las tareas tipo pregunta respuesta se usa un esquema similar, la única diferencia es que al final la red neuronal y la capa soft max no entregarán una categoría como en el caso anterior sino que realizarán la predicción de los tokens de inicio y finalización de la respuesta encontrada dentro del texto.

Esto es BERT, un poderoso modelo que ha generado una revolución en el NLP, además de existir modelos pre entrenados en más de 100 idiomas se pueden encontrar muchas librerías de código abierto que implementan BERT para diferentes tipos de aplicaciones del NLP.

¿Qué hace que BERT sea diferente?

BERT se basa en trabajos recientes en representaciones contextuales previas al entrenamiento, incluido el aprendizaje de secuencia semi supervisado, el entrenamiento previo generativo, ELMo y ULMFit. Sin embargo, a diferencia de estos modelos anteriores, BERT

es la primera representación de lenguaje no supervisada y profundamente bidireccional, entrenada previamente utilizando solo un corpus de texto sin formato (en este caso, Wikipedia).

¿Por qué importa esto? Las representaciones pre entrenadas pueden ser contextuales o independientes del contexto, y las representaciones contextuales pueden ser unidireccionales o bidireccionales. Los modelos independientes del contexto, como word2vec o GloVe, generan una representación incrustada de una sola palabra para cada palabra del vocabulario. Por ejemplo, la palabra "banco" tendría la misma representación sin contexto en "cuenta bancaria" y "banco del río". En cambio, los modelos contextuales generan una representación de cada palabra que se basa en las otras palabras de la oración. Por ejemplo, en la oración "Accedí a la cuenta bancaria", un modelo contextual unidireccional representaría "banco" basado en "Accedí a" pero no a "cuenta". Sin embargo, BERT representa "banco" utilizando tanto su contexto anterior como el siguiente: "Accedí a la... cuenta", comenzando desde el fondo de una red neuronal profunda, haciéndola profundamente bidireccional.

La fuerza de la bidireccionalidad: Si la bidireccionalidad es tan poderosa, ¿por qué no se ha hecho antes? Para entender por qué, considere que los modelos unidireccionales se entrenan eficientemente al predecir cada palabra condicionada a las palabras anteriores en la oración. Sin embargo, no es posible entrenar modelos bidireccionales simplemente condicionando cada palabra en sus palabras anteriores y siguientes, ya que esto permitiría que la palabra que se predice se "vea a sí misma" indirectamente en un modelo de múltiples capas.

Para resolver este problema, usamos la técnica directa de enmascarar algunas de las palabras en la entrada y luego condicionar cada palabra bidireccionalmente para predecir las palabras enmascaradas. Si bien esta idea existe desde hace mucho tiempo, BERT es la primera vez que se usa con éxito para pre entrenar una red neuronal profunda.

Entrenamiento con Cloud TPU

Todo lo que hemos descrito hasta ahora puede parecer bastante sencillo, entonces, ¿cuál es la pieza que falta para que funcione tan bien? TPU en la nube. Cloud TPU nos dio la libertad de experimentar, depurar y ajustar rápidamente nuestros modelos, lo que fue fundamental para permitirnos ir más allá de las técnicas de capacitación previa existentes. La arquitectura del modelo Transformer, desarrollada por investigadores de Google en 2017, también nos brindó la base que necesitábamos para que BERT tuviera éxito. El transformador está implementado en nuestra versión de código abierto, así como en la biblioteca tensor2tensor.

Capítulo 4: El sarcasmo y sus implicaciones

4.1 Representando formas de sarcasmo

Los esfuerzos de los usuarios para generar sarcasmo se manifiestan de muchas maneras en *YouTube*. En esta sección, se describe cómo se realizan diferentes formas de sarcasmo y cómo se pueden construir características relevantes para capturar estas formas en el contexto de *YouTube*.

El sarcasmo como contraste de sentimientos: Una percepción popular del sarcasmo entre los investigadores es que el sarcasmo es un contraste de sentimientos. Una visión clásica del sarcasmo, basada en el modelo pragmático tradicional (Grice, 1975), sostiene que las expresiones sarcásticas se procesan primero en el sentido literal y si el sentido literal se encuentra incompatible con el contexto actual, solo entonces se procesa la oración en su sentido literal. Forma opuesta (irónica). Este contraste percibido se puede expresar a través de múltiples facetas, como el estado de ánimo, el afecto o el sentimiento.

Un medio común de expresar el sarcasmo es emplear palabras con connotaciones contrastantes dentro del mismo comentario. Por ejemplo, en ¡Me encanta recibir correos electrónicos no deseados !, el spam tiene una connotación negativa obvia mientras que el amor es abrumadoramente positivo. Para modelar tales ocurrencias, se propone construir características basadas en

(1) puntuaciones de afecto

(2) de sentimiento.

Obtenemos la puntuación de afectación de las palabras a partir de un conjunto de datos compilado por Warriner et al. (Warriner, 2013). Este conjunto de datos contiene puntuaciones de afecto (valencia) para 13,915 lemas que están en una escala de 9 puntos, siendo 1 el menos agradable. La puntuación del sentimiento se propone calcular utilizando SentiStrength (Thelwall, 2010). SentiStrength es una herramienta basada en léxico optimizada para la detección de sentimientos de textos basada en sentimientos de palabras individuales en el comentario. Además de proporcionar un resultado de sentimiento ternario {positivo, negativo, neutral} para todo el comentario. Una puntuación de sentimiento negativo de -1 a -5 (no negativa a extremadamente negativa) y una puntuación de sentimiento positiva de 1 a 5 (no positiva a extremadamente positiva).

4.2 El sarcasmo como forma compleja de expresión

Rockwell (Rockwell P., 2000) mostró que existe una pequeña pero significativa correlación entre la complejidad cognitiva y la capacidad de producir sarcasmo. Una alta complejidad cognitiva implica comprender y tener en cuenta múltiples perspectivas para tomar decisiones convincentes. Además, expresar el sarcasmo requiere determinar si el ambiente es adecuado para el sarcasmo, crear una frase sarcástica adecuada y evaluar si el receptor sería capaz de reconocer el sarcasmo. Por lo tanto, el sarcasmo es una forma compleja de expresión que requiere más esfuerzo del usuario (McDonald, 1999).

4.3 El sarcasmo como medio de transmitir emociones

El sarcasmo es principalmente una forma de transmitir las emociones de uno. Si bien el sarcasmo se interpreta en ocasiones como humor agresivo (Basavanna, 2000) y como forma de agresión verbal (Toplak, 2000), también funciona como una herramienta de autoexpresión. Estudios anteriores (Grice, 1975) reconocen que el sarcasmo generalmente se expresa en situaciones con emociones y actitudes negativas.

4.4 El sarcasmo en función de la familiaridad.

Se encuentra que los amigos y familiares son mejores para reconocer el sarcasmo que los extraños (Rockwell P., 2003). Además, se ha demostrado que la familiaridad con el lenguaje (Cheang, 2011) y los factores culturales (Rockwell P. a., 2001) también juegan un papel importante en el reconocimiento y uso del sarcasmo.

4.5 El sarcasmo como forma de expresión escrita.

En psicología, el sarcasmo se ha estudiado principalmente como una forma de expresión hablada. Sin embargo, el sarcasmo también es bastante frecuente en el contexto escrito, especialmente con el advenimiento de los sitios de redes sociales en línea. A través del tiempo, los usuarios se han vuelto más adeptos a transmitir el sarcasmo por escrito al

incluir marcadores sutiles que indican al lector modesto que la frase es sarcástica. Por ejemplo, mientras que "eres tan inteligente" no insinúa un sarcasmo, "Woowwww eres TAAN genial" suscita algunas dudas sobre la sinceridad de la declaración.

Se tiene la hipótesis que, al expresar el sarcasmo, el usuario invariablemente exhibirá una o más de las formas de sarcasmo mencionadas anteriormente. Por lo tanto, se ha construido un marco de modelado de comportamiento para la detección de sarcasmo que utiliza características que modelan estas diferentes formas. Estas características extraídas se utilizan para entrenar un modelo de clasificación supervisado para determinar si el comentario es sarcástico o no. Como la novedad del enfoque reside en el modelo de comportamiento y no en el clasificador en sí, explicamos más detalladamente cómo se modela el sarcasmo y cómo se incorpora en el marco. Si el lector no está familiarizado con *Youtube*, se incluye una breve introducción de *Youtube* en la sección Apéndice. Se recomienda a los lectores que conocen bien *Youtube* que continúen con el siguiente capítulo que describe en detalle la construcción de la función.

4.6 Contrastando el presente con el pasado.

Mientras que los usuarios a menudo usan palabras de contraste en el mismo comentario para expresar sarcasmo, muchas veces, un usuario puede configurar un contexto de contraste en su comentario anterior y luego, optar por utilizar un comentario sarcástico en su comentario actual. Para modelar dicho comportamiento, obtenemos el sentimiento expresado por el usuario (es decir, positivo, negativo, neutral) en el comentario anterior y el comentario actual usando SentiStrength. Luego, se incluye el tipo de transición de sentimiento que se lleva a cabo desde el último comentario al comentario actual (por ejemplo, positivo → negativo, negativo → positivo) como una característica (1 característica).

4.7 Afecto y sentimiento.

Como el sarcasmo es una combinación de afecto y expresión de sentimiento, se explora la posibilidad de observar diferencias con respecto a cómo se expresa el afecto y el sentimiento en un comentario sarcástico. Para este fin, se propone construir una distribución de puntuación de sentimiento *SS* en la que cada conteo es el número de palabras en el

comentario con puntuación de sentimiento *i* donde $i \in [-5, 5]$. También construimos una distribución de puntuación de afecto *AS* en la que cada conteo es el número de palabras en el comentario de la puntuación de afecto *j* donde $j \in [1, 9]$. Normalizamos los conteos en *SS* y *AS*. Incluimos como características estas dos distribuciones (20 características). Similar a la sección 4.6 contrastando el presente con el pasado representamos estas distribuciones como 6-tuplas y las incluimos como características (12 características). También incluimos el número de palabras afectivas, el número de palabras de sentimiento y el sentimiento expresado (positivo, negativo y neutral) que se obtienen de SentiStrength como características (3 características).

Para capturar la diferencia en la expresión de sentimiento, comparamos la distribución de puntuación de sentimiento de los comentarios anteriores del usuario con la de su comentario actual. Calculamos la divergencia de *JS* entre las distribuciones de puntuación de sentimiento actuales y pasadas y las incluimos como una característica (1 característica). Con el fin de obtener información sobre el rango de sentimientos expresados por el usuario para evaluar cómo usa *YouTube* como una herramienta para expresar emociones, construimos una distribución normalizada sobre la puntuación del sentimiento [−5, 5] de cada palabra de sus comentarios anteriores e incluir la distribución como una característica (11 características). Esta distribución da una percepción de cuán expresivo es el usuario, en YouTube. Esto es crucial ya que diferentes usuarios usan *YouTube* por diferentes razones. Algunos usuarios de *YouTube* comentan datos objetivos o noticias y generalmente son información, mientras que otros usuarios son bastante informales y comentan problemas personales, emociones, opiniones, etc.

4.8 Frustración.

Cuando las personas observan o experimentan una situación injusta, a veces recurren a las redes sociales que actúan como salidas efectivas para sus quejas y frustraciones (Bi, 2012). Esta frustración se expresa a menudo en forma de sarcasmo (Gibbs, 2000). Por lo general, el sarcasmo no es premeditado; es una reacción espontánea a ciertos eventos o

escenarios desagradables / perturbadores. Por lo tanto, cuantificar la espontaneidad de un comentario puede proporcionar información sobre si el comentario es sarcástico o no.

Para modelar la espontaneidad, utilizando los comentarios anteriores del usuario, se propone construir una distribución de probabilidad de publicación de comentarios esperada que describe las normas regulares de comentar del usuario. De cada uno de los comentarios anteriores del usuario, se propone extraer el tiempo de creación del comentario, utilizando el cual construimos una distribución $T\,T$ normalizada de 24 contenedores (una para cada hora). $T\,T$ se aproxima a la probabilidad de que el usuario comente a cada hora. Para cada comentario examinado, utilizando la $T\,T$ del usuario respectivo, encontramos la posibilidad de que un usuario publique el comentario a esa hora. Cuanto menor sea la probabilidad, más divergente será el comentario de los patrones de comentar habituales del usuario. Los puntajes de baja probabilidad indican que no se espera que el usuario envíe un comentario en ese momento en particular y que el usuario haya hecho todo lo posible para comentar en ese momento, por lo tanto, en cierto sentido, el comentario es de naturaleza espontánea. Incluimos como característica, la puntuación de probabilidad real del usuario que comenta en esa hora en particular (1 característica).

Otra forma común de expresar la frustración es a través del uso de malas palabras. Usando la compilación de (Wang, 2006) de las palabras más comunes, se propone comprobar la presencia de tales palabras en el comentario e incluir su presencia como una característica booleana (1 característica).

4.9 Familiaridad del lenguaje.

Intuitivamente, uno esperaría que un usuario que usa una forma de lenguaje tan compleja como el sarcasmo tenga un buen dominio del idioma. Por lo tanto, obtenemos un perfil de las habilidades lingüísticas del usuario mediante la medición de características inspiradas en las pruebas de cloze estandarizadas de competencia lingüística. Como parte de la prueba de cloze (Oller, 1972), la competencia se evalúa en base a los niveles de vocabulario, gramática, dictado y lectura. Como los niveles de dictado y lectura corresponden a las habilidades de oratoria y lectura del usuario que no pueden medirse a partir de un texto escrito,

concentramos nuestros esfuerzos en construir las características que mejor representan las habilidades de vocabulario y gramática. Usando los comentarios anteriores del usuario, determinamos el tamaño de su vocabulario. Incluimos como características, el total de palabras, el total de palabras distintas utilizadas y la proporción de distintas palabras utilizadas para medir la redundancia del usuario en el uso de palabras (3 funciones).

A menudo, la ubicación es un factor importante en cómo se habla el idioma. Además, se ha demostrado que las personas en diferentes regiones perciben y usan el sarcasmo de manera diferente. Por ejemplo, comparando norteños y sureños en los EE. UU., (Dress, 2008) mostró que los norteños formulan más oraciones sarcásticas en comparación con los sureños. Por lo tanto, tratamos de inferir la ubicación aproximada del usuario. Sin embargo, como el campo de ubicación en *YouTube* es un campo de texto libre en el que se puede ingresar cualquier texto, a menudo es ruidoso. Por lo tanto, se propone aproximar la ubicación del usuario con su zona horaria e incluirla como una característica (1 característica). Esta característica indica si el usuario está familiarizado con el sarcasmo como forma de expresión.

4.10 Familiaridad con el medio ambiente

En general, los usuarios expresan mejor el sarcasmo cuando conocen bien el medio ambiente. Así como es menos probable que las personas utilicen el sarcasmo en un entorno nuevo y desconocido, creemos que los usuarios tardarán un tiempo en aclimatarse con *YouTube* antes de publicar comentarios sarcásticos. Por lo tanto, se propone medir la familiaridad en términos de la cantidad de comentarios publicados, la cantidad de días desde que el usuario creó su perfil de *YouTube* (edad de *YouTube*), la cantidad de comentarios divididos por la edad de *YouTube* del usuario y usarlos como funciones (3 funciones). Estas características dan una indicación de la duración durante la cual el usuario ha estado usando *YouTube*.

4.11 El sarcasmo como forma de expresión escrita.

Mientras que el tono bajo, la intensidad alta y el tempo lento (Rockwell P. , 2000) son indicadores vocales de sarcasmo, los usuarios que intentan expresar el sarcasmo por

escrito carecen de tales dispositivos. Por lo tanto, los usuarios pueden verse obligados a innovar y usar ciertos estilos de escritura para compensar la falta de señales visuales y verbales. Clasificamos las variaciones que se derivan de dicho comportamiento como (i) prosódico o (ii) estructural.

Prosódico:*Variations Prosody* ha sido estudiada e identificada como una de las principales señales de sarcasmo (Wang, 2006). Las variaciones prosódicas se refieren a los cambios realizados en los estilos de escritura para expresar la entonación y el estrés. El lenguaje en las redes sociales está evolucionando continuamente a medida que los usuarios encuentran formas simples pero efectivas de expresarse mejor dentro de las limitaciones impuestas por el sitio de redes sociales. Los usuarios a menudo repiten letras en palabras para enfatizar y sobre enfatizar ciertas partes del comentario (por ejemplo, taaaaaaaaaaaaaaaaaaaan) para indicar que significan lo contrario a lo que está escrito. Se propone capturar dicho uso incluyendo como características booleanas, la presencia de caracteres repetidos (3 o más) y la presencia de caracteres repetidos (3 o más) en palabras cargadas de sentimiento (como “amooor”) (2 características). También se observa que los usuarios a menudo escriben mayúsculas en ciertas palabras para enfatizar los cambios en el tono (si se leyera el texto en voz alta). Se propone identificar tales cambios al incluir como características el número de palabras en mayúsculas en el comentario (1 característica).

4.12 Propósito de la aplicación

El propósito de la aplicación es clasificar textos cortos o comentarios de carácter específico, es decir el tema de cada texto no puede ser de política, entretenimiento, economía o deportes y las clases o categorías que se consideran son sarcástico y no sarcástico.

Capítulo 5. Metodología

El objetivo deseado del proyecto es determinar la clase a la que pertenecen los textos a procesar, para desarrollar esto se han creado dos aplicaciones que están basadas en el aprendizaje automático supervisado. Se hacen necesarias las tareas de preprocesamiento, selección de características que aporten información para la determinación de clases y un sistema de clasificación automática. La segunda aplicación se basa en el Deep Learning utilizando BERT como base para clasificar el corpus con mejor desempeño con el sistema anterior. A continuación, en la figura 5.1 se muestran los pasos que se proponen para realizar las pruebas.

Los corpus que se tienen contemplados son los siguientes:

1. Corpus multilenguaje: Este corpus es una recopilación de comentarios en twitter en diferentes idiomas, tanto alemán, inglés, español, italiano y portugués, con más de 10000 comentarios etiquetados como sarcástico y no sarcástico, un volumen considerable, pero sin contexto y ni siquiera en el idioma en el que se desea hacer la prueba.
2. Corpus SP: comentarios su contenido se basa en comentarios etiquetados de un capítulo de la famosa serie animada South Park como sarcásticos o no sarcásticos, este corpus es muy pequeño alrededor de 100 comentarios.

3. Corpus Original 100 muestras: Este corpus fue creado desde 0 con muchos comentarios en español de usuarios en YouTube exclusivamente. La polaridad de los comentarios se distribuye en 2 etiquetas: sarcastic y non-sarcastic.
4. Corpus Original 255 muestras: Este corpus fue creado desde 0 con muchos comentarios en español de usuarios en YouTube exclusivamente. La polaridad de los comentarios se distribuye en 2 etiquetas: sarcastic y non-sarcastic .

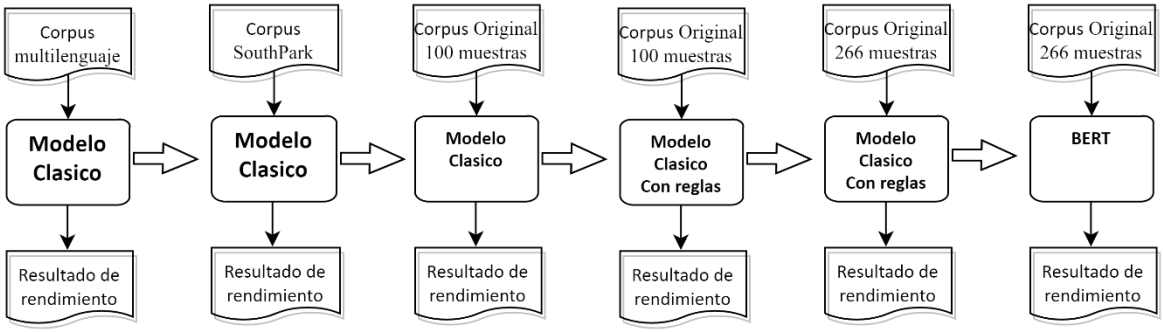


Figura 5.1 Proceso de experimentación

El proceso para realizar la experimentación consta de diferentes corpus que sirven de información para entrenar los modelos y también para realizar las pruebas de rendimiento, El primer modelo será el mismo en las primeras pruebas, posteriormente se agregaran reglas de sustitución de términos para buscar conseguir más información que ayude a la correcta clasificación de los comentarios, posteriormente el corpus con mejor rendimiento será el que sea usado en las pruebas con BERT.

5.1 Modelo 1- Clasificador Naive Bayes

El primer modelo de clasificación está basado en trabajos anteriores, se basa en la clasificación por medio de la estadística concretamente en el método de clasificación Naive Bayes, el modelo toma el corpus seleccionado y lo preprocesa para mejorar los procesos siguientes, posteriormente es separado en dos conjuntos de datos, uno de entrenamiento y otro de prueba, posteriormente es entrenado y evaluado con el conjunto de prueba como se muestra en la figura 5.1.1.

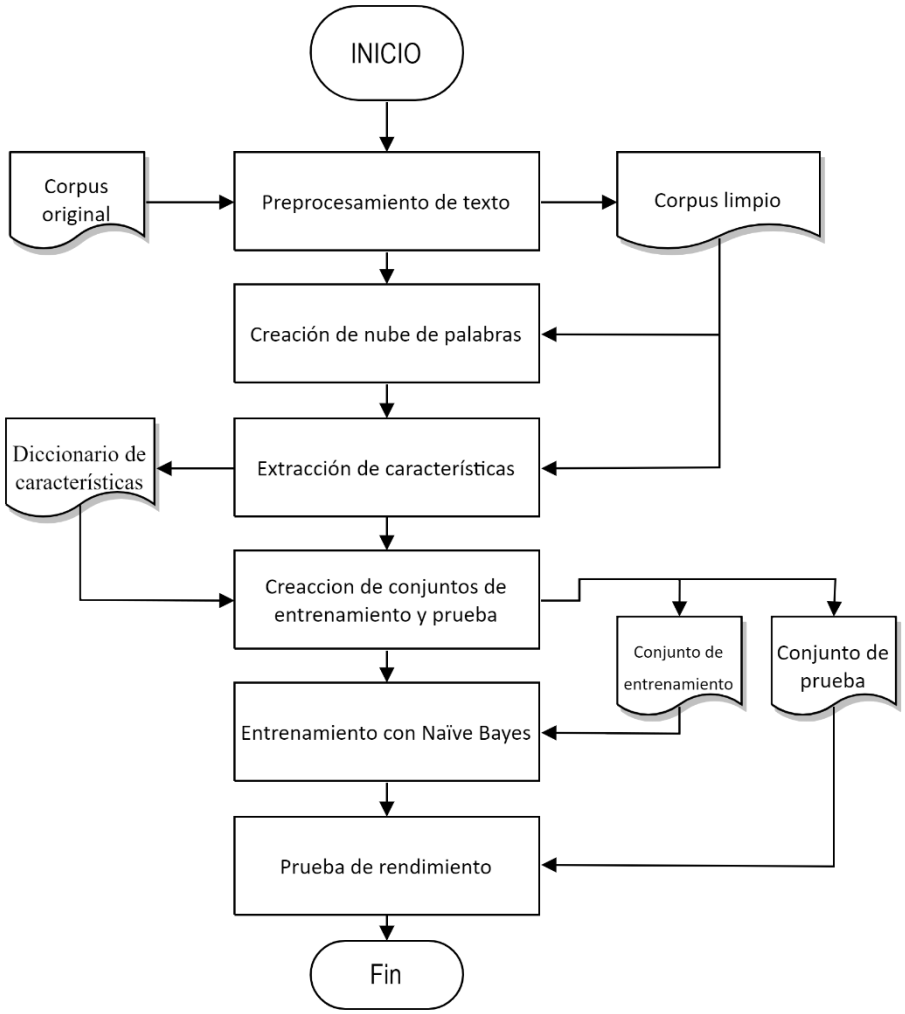


Figura 5.1.1 Modelo de clasificación 1

5.1.1 Entorno de programación

Como se puede ver en la representación de la figura 5.1.2, el entorno de programación es LINUX Ubuntu, en este sistema operativo se crea un entorno virtual de programación en el que se instala el entorno R para realizar los pasos de tratamiento y prueba del corpus.

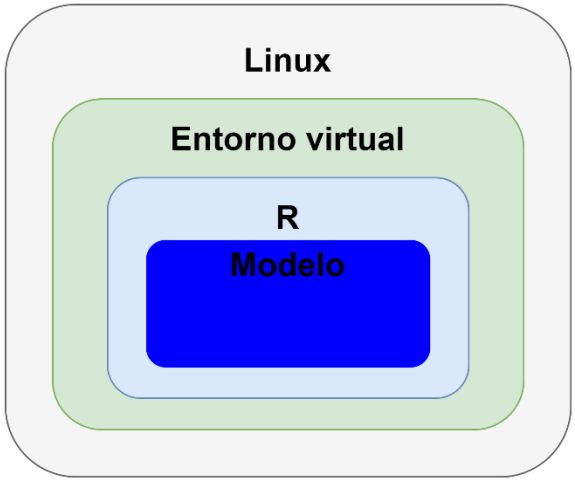


Figura 5.1.2 Representación gráfica del entorno de programación.

5.1.2 Estructuración de los elementos

La solución a la problemática de clasificación de textos usando aprendizaje automático tiene varias aproximaciones. La primera y más comúnmente usada es realizar un preprocesamiento del texto debido al contexto y la jerga que se usa en internet y si a esto se le agrega el hecho de que los mismos usuarios comentan en diferentes plataformas en línea, en diferentes convenciones e idiomas se genera un lenguaje mutado y deformado que se necesita corregir para facilitar la aproximación del problema. Debido a esto es necesario contar con un paso que normalice el texto, en este caso en concreto el preprocesamiento del texto es necesario en la parte del corpus de entrenamiento y en la parte de pruebas. A continuación, se muestran los casos de uso a realizar por el módulo de preprocesamiento.

5.1.3 Eliminación de texto innecesario

Es de suma importancia la eliminación de texto que no aporta ninguna información útil o relevante al modelo de clasificación, por lo que direcciones web, números, caracteres especiales y todas las palabras de enlace son excluidas del texto esto debido a que no aportan información útil, si los números no fueran eliminados de la información útil estos podrían ser considerados como características de comentarios sarcásticos o no sarcásticos afectando negativamente el resultado. Los caracteres especiales también suelen ser negativos para la clasificación con excepciones que se podrían considerar como en el caso de las comillas.

Es necesario reestructurar todos y cada uno de los elementos en el corpus para los próximos pasos por lo tanto se considera necesaria la eliminación de texto que no aporta información útil o relevante para el modelo de clasificación, por lo que emoticones, palabras de enlace y caracteres especiales son excluidos

El primer paso consiste en pasar de un corpus no estructurado a uno estructurado utilizando un código el cual limpia el corpus de palabras que no son necesarias para pasos futuros además de token izar cada comentario en varias partes.

Primero se limpia el ambiente de programación en R y se crea una función para tomar el vector del conjunto de datos y realizar todos los pasos de preprocesamiento.

Pasos de preprocesamiento: Eliminación de números, palabras y puntuación, realización de lematización y eliminación de espacios en blanco adicionales. Se utiliza la función vec2clean.corp toma dos argumentos: x = vector a limpiar, y = número de documento para mostrar el proceso mediante la impresión.

5.1.4 Normalización y sustitución de términos

La normalización de términos se usa en aquellos casos en los que han sido modificados por la jerga de internet por ejemplo en la tabla 5.1.3 se muestran las palabras "largooo…" y "Dolooor", de las cuales es fácil deducir que equivalen al adjetivo "largo" y al nombre "dolor", por lo que la normalización hace necesaria una función que reduzca el número de caracteres repetidos y elimine los signos de puntuación. Se creó un método para reemplazar 'y "a los espacios antes de eliminar la puntuación para evitar que se unan palabras diferentes, además conforme avanzó el proyecto esta misma función sirvió para reemplazar algunas palabras u onomatopeyas que sirvieron para agregar contexto al corpus, por ejemplo, las risas, o algunos signos de puntuación como "¡¡¡!!!" o puntos suspensivos "…" estos fueron reemplazados por palabras como "risa", "admiración" o "Suspensivo".

La normalización comienza eliminando el término disperso, es decir palabras que son muy poco usadas, elimina todos los números, signos de puntuación y convierte todo a minúsculas, después convierte la matriz de términos (todas las palabras que no fueron eliminadas por ser dispersas) del documento en un marco de datos. Al resultado final se le llama corpus limpio y queda guardado para los siguientes pasos del modelo.

Texto original	Texto normalizado	Texto sin aplicar eliminación y normalización de terminos
Salgo de #VeoTV, que día más largoooo!...	Salgo de que día más largo	Salgo voto que día más largoooo..
@Mauperez preciosa de verdad- perdóname by Pablo Alboran, from	Preciosa de verdad perdóname by Pablo Alboran, from	Mauperezmk Preciosa de verdad - perdóname by Pablo Alboran, from

Tabla 5.1.3 Normalización

En pasos posteriores de la experimentación además de eliminar términos dispersos obtiene algunas onomatopeyas y/o palabras de la jerga común que no son precisamente oficiales y las convierte en palabras como se puede apreciar en la tabla 5.1.4 que pueden ser clasificadas como características más adelante.

Herramienta	Texto
Original	"Jajajajaja 21 km/L siempre y cuando no rebases los 50km/hr. ¡Cascaron bonito... Misma receta vieja!"
Normalización a minúsculas	"jajajajaja 21 km/l siempre y cuando no rebases los 50km/hr. cascaron bonito... misma receta vieja!"
Eliminación de números	"jajajajaja km/l siempre y cuando no rebases los km/hr. cascaron bonito... misma receta vieja!"
Eliminación de palabras de enlace	"jajajajaja km/l siempre rebases km/hr. cascaron bonito... misma receta vieja!"
Eliminación de comillas y sustitución de palabras clave	" riendo ajaja kilometros por hora siempre rebases kmhr cascaron bonito suspenso misma receta vieja!"
Eliminación de signos de puntuación	" riendo ajaja kilometros por hora siempre rebases kmhr cascaron bonito suspenso misma receta vieja"
Eliminación de palabras no reconocidas	"riendo kilometro hora siempre rebases cascaron bonito suspenso misma receta vieja"

Tabla 5.1.4 Preprocesamiento de datos en corpus

```r
## No estructurado a estructurado

# En este código, creamos un corpus,
#limpiamos el corpus e implementamos la tokenización creando DTM

# limpiar el medio ambiente
rm(list= ls())
# verificar y configurar el directorio de trabajo
getwd()
#setwd("C:/Data Scientist/Edwisor/Sarcasm_R")
setwd("/home/ubuntu/kivy_venv")
# cargando bibliotecas requeridas

require(tm)           # para minería de texto
require(data.table)   # para operaciones de datos más rápidas

# leyendo el conjunto de datos del directorio de trabajo actual
tweet <- fread('TweetsDataSet.csv', encoding="Latin-1")

# comprobación de estructura y resumen
dim(tweet)
str(tweet)
summary(tweet)

head(tweet)
tail(tweet)

## Preprocesamiento de los comentarios y limpieza del corpus

# variables y funciones definidas por el usuario

# Función para tomar el vector del conjunto de datos de comentarios y realizar todos los pasos de preprocesamiento a continuación:
# pasos de preprocesamiento:  Eliminacion de números, palabras y puntuación, lematización y eliminación de espacios en blanco adicionales

# vec2clean.corp La función toma dos argumentos: x = vector a limpiar, y = número de documento para mostrar el proceso imprimiendo
vec2clean.corp <- function(x,y=NULL){

  # Como se utilizan muchos idiomas en los datos en un corpus, consideramos palabras vacías de todos los idiomas, cuando no se usan otros idiomas se comentan las stopwords
  a = c(#stopwords("danish"),stopwords("dutch"),stopwords("english"),
        #stopwords("finnish"),stopwords("french"), stopwords('SMART'),
        #stopwords("german"),stopwords("hungarian"),stopwords("italian"),
        #stopwords("norwegian"),stopwords("portuguese"),stopwords("russian"),
        stopwords("spanish")#,stopwords("swedish"))
)

  # Función para reemplazar risas, puntos suspensivos y otros caracteres por palabras  antes de eliminar la puntuación para evitar perdida de informacion.
  AposToSpace = function(x){
```

Figura 5.1.3 Fragmento de código

En la figura 5.1.3 se puede apreciar parte del código del primer proceso, pasando los datos de texto no estructurado a un texto estructurado, cargando los componentes necesarios y creando una función la cual se encarga de limpiar cada vector de palabras del corpus, en este caso se comentaron las stopwords ya que en los últimos corpus solo se usaron vectores de palabras en idioma español.

5.1.5 Ingeniería de características

Este paso del código trata sobre la ingeniería de funciones y la selección de funciones. Se comprueba correlaciones y asociaciones entre variables, es decir que si en un comentario se encuentran dos palabras y la etiqueta de ese comentario fue sarcástico, entonces esa correlación será buscada en los demás comentarios que también fueron catalogados como sarcásticos.

```
## Este código trata sobre la ingeniería de funciones y la selección de funciones
# Comprobamos correlaciones y asociaciones entre variables.

# limpiar el medio ambiente
rm(list= ls())
require(data.table)
require(tm)

#

# cargando los conjuntos de datos anteriores
load('step1_DS.dat')
load('step2_DS.dat')
setwd("/home/ubuntu/kivy_venv")
# cargando  el conjunto de datos del corpus limpio
tweet <- fread('TweetsDataSet.csv', encoding="Latin-1")
dim(tweet)
tweet$label <- as.factor(tweet$label)

#eliminando la escasez y preparar el marco de datos
sparse <- removeSparseTerms(dtm, 0.9992)
sparse

df <- data.table(as.matrix(sparse))
df
dim(df)
head(df,2)
# Encuentra asociaciones
unlist(findAssocs(sparse, findFreqTerms(sparse,100),corlimit = 0.5))
rm(dtm,sparse)

# optamos por la matriz de correlación de Pearson para obtener información más detallada
corr <- data.table(cor(df, use = "complete.obs", method= "pearson"))
dim(corr)
corr.terms <- NULL
for(i in 1:(nrow(corr)-1)){
  for(j in (i+1):ncol(corr)){
    if((abs(corr[[i,j]])>0.49) ==T){
      corr.terms = c(corr.terms, names(corr)[i])
      print(paste(colnames(corr)[i],',',colnames(corr)[j])) # imprimir filas y números de columna que están correlacionados
    }
  }
}
# corr.terms consiste en términos correlacionados que son más del 50% con cualquier otra variable
# solo se agrega un par correlacionado de término mientras se ejecuta el ciclo 'for'
rm(corr,i,j)
corr.terms
```

Figura 5.1.4 Fragmento de código

Como se muestra en la figura 5.1.4 primero se limpia el ambiente de programación. Se carga el conjunto de datos anteriores, se elimina el sparcity (escasez) eliminando las correlaciones de palabras que menos reaparecieron a lo largo del corpus y se prepara el marco de datos. Se encuentran asociaciones, se crea un conjunto de datos maestros y se guarda para el muestreo como un diccionario de características.

5.1.6 Muestreo de entrenamiento y prueba

Este paso trata sobre la preparación de muestras del conjunto de datos maestros y la división de conjuntos de entrenamiento y prueba. Parte del código se puede ver en la figura 5.1.6 donde el inicio de este paso es limpiar el ambiente de programación en R, luego se cargan las bibliotecas necesarias. Posteriormente se usa una función creada con el fin de producir los conjuntos de entrenamiento y prueba, La función contiene un parámetro el cual divide el conjunto maestro para usar solo una parte del total de muestras para crear los dos conjuntos de entrenamiento y prueba.

```
# Este código trata sobre la preparación de muestras del conjunto de datos maestros
# dividiendo conjuntos de entrenamiento y prueba

# limpiar el  ambiente
rm(list = ls())

# cargando las bibliotecas necesarias
require(caTools)

# función para crear una muestra y empujar el tren y los conjuntos de datos de prueba
# estamos considerando el conteo de muestras como el 98% del total del corpus con un 80% para el entrenamiento
# proporción de 0.8 como proporción de entrenamiento y prueba

sample2train.test <- function(master.x, seed.x, samp.ratio= 0.98, train.ratio= 0.8){
  set.seed(seed = seed.x)
  samp.split = sample.split(master.x$label, samp.ratio)
  sample = subset(master.x, samp.split == T)

  # training y testing
  spl = sample.split(sample$label, train.ratio)
  train.x = subset(sample, spl == T )
  test.x  = subset(sample, spl == F )
  return(list(train.x, test.x))
}

#  carga el conjunto de datos

# to load master.numeric
load('master.numeric.dat')

# to load master.factor
load('master.factor.dat')

# pasando el conjunto de datos maestros deseado, la semilla (para producir una muestra aleatoria), la relación de muestra y la relación de entrenamiento


# Para producir conjuntos de entrenamiento y prueba de master.numeric
Train.Test.list <- sample2train.test(master,123)

train.num <- Train.Test.list[[1]]
test.num  <- Train.Test.list[[2]]

# Guardando sets numericos de train y test
save(train.num, test.num, file = 'TrainTest_num.dat')

# Para producir sets de Training y Testing  para  master.factor
Train.Test.list <- sample2train.test(master.factor,123)
```

Figura 5.1.5 Fragmento de código

La proporción de ratio que se pretende usar para fines de probar el procedimiento es variable, pero al final se pretende optar por la proporción 0.8 para entrenamiento y 0.2 para prueba. Ambos conjuntos se guardan en dos archivos diferentes para ser usados en los siguientes pasos.

5.1.7 Entrenamiento con Nayve Bayes

Se construye el modelo Naive Bayes en master.factor y se guarda el modelo, posteriormente se limpia el ambiente de programación, se cargan los archivos creados en pasos anteriores y se usa un modelo de Naive Bayes con estimador de Laplace 1 para asegurar una probabilidad distinta de cero para cada característica. Luego se guarda el modelo para la evaluación.

5.1.8 Código de evaluación Naive Bayes

En este paso como se aprecia en la figura 5.1.8 se evalúa el modelo creado en los pasos anteriores. De igual manera como en los pasos anteriores primero se limpia el ambiente de programación en R, luego se cargan los paquetes requeridos. Se cargan los conjuntos de entrenamiento y prueba, luego se carga el modelo Naive Bayes y posteriormente se prueba el modelo con estos conjuntos.

```
# Código de evaluación Naive Bayes

# limpiar el medio ambiente
rm(list = ls())


# cargando paquetes requeridos
require(e1071)

# cargando conjuntos de entrenamiento y prueba
load('TrainTest.dat')

# cargando los modelos Naive Bayes
load('NB_models.dat')

# Predicción de la clase objetivo de prueba
# para el modelo clásico Naive Bayes
n.pred <- predict(n.model, test[,-1], type = 'class')

xtab.n <- table('Actual class' = test[,1], 'Predicted class' = n.pred )
caret::confusionMatrix(xtab.n)

# para un modelo Naive Bayes robusto con estimador de Laplace
n.pred.lap <- predict(n.model.lap, test[,-1], type = 'class')

xtab.lap <- table('Actual class' = test[,1], 'Predicted class' = n.pred.lap )
caret::confusionMatrix(xtab.lap)

result <-caret::confusionMatrix(xtab.lap)

result
result$byClass["Precision"]
result$byClass["Recall"]
result$byClass["F1"]
```

Figura 5.1.8 Fragmento de código

5.2 Modelo 2 - Clasificador Deep Learing BERT

El siguiente modelo de clasificación propuesto como se observa en la figura 5.2.1 se basa en técnicas de Deep Learning cargando la plataforma BERT en COLAB y utilizando la estructura de redes transformer que tiene BERT para ayudar a clasificar los comentarios del mejor corpus utilizado en el modelo.

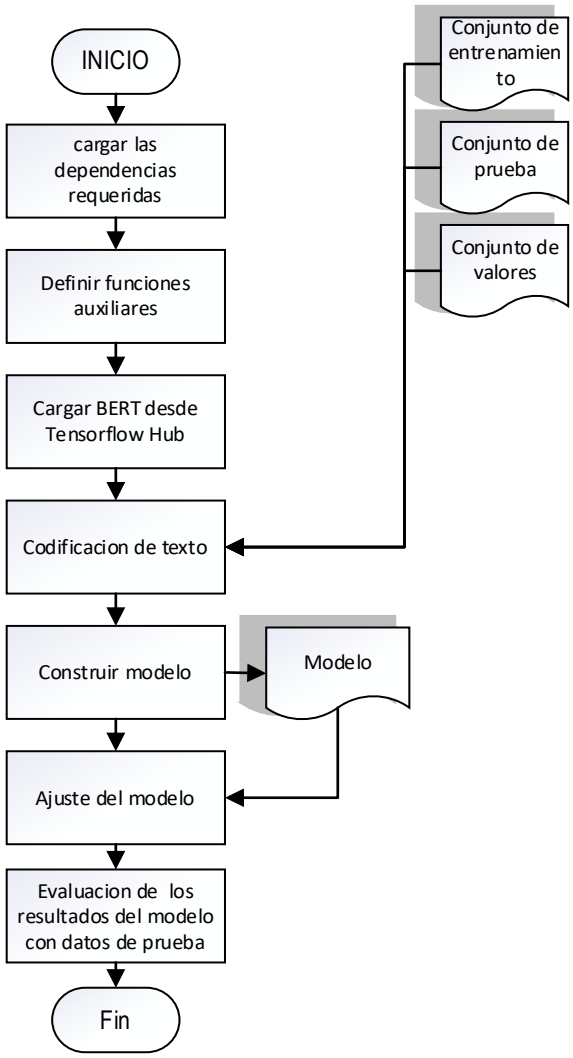


Figura 5.2.1 Modelo de clasificación 2 propuesto

5.2.1 Entorno de programación

El entorno de programación es COLAB de Google, en este sistema se usa Python para cargar las dependencias necesarias para cargar BERT y utilizarlo con su estructura de redes transformer en su encoder como se muestra en la figura 5.2.1.

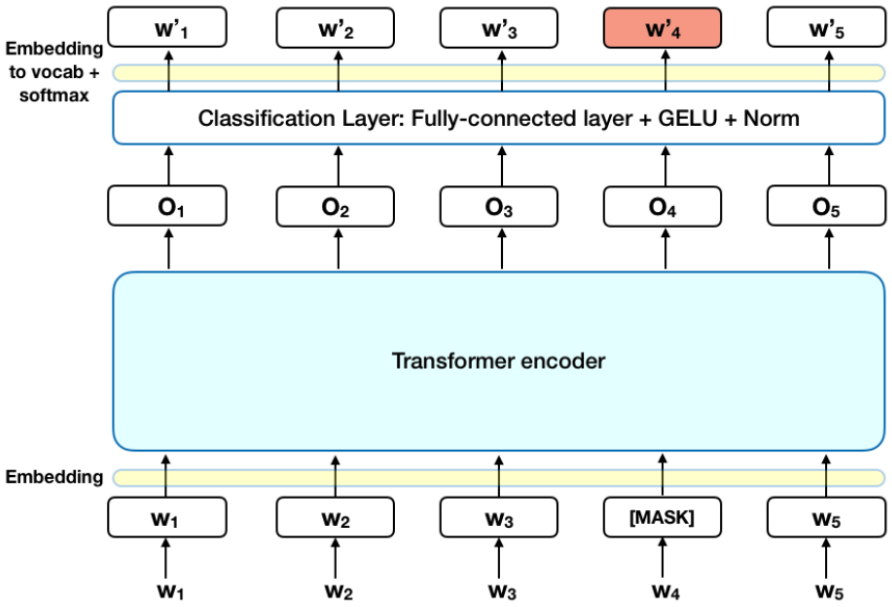


Figura 5.2.1 Modo de operación de BERT

5.2.2 Cargar las dependencias requeridas

En esta sección se instalan y se cargan todas las dependencias que se usaran, tanto para crear los tokens como para poder cargar BERT y poder usar sus redes transformer, podemos observar en la figura 5.2.2, se necesitan ciertas dependencias para poder correr el resto de código, dentro de las librerías más destacadas se encuentran tensorflow de la cual se podrá cargar BERT, además de pandas la cual proporciona herramientas de manipulación y análisis de datos.

```
[ ]  !pip install sentencepiece
     !pip install tokenization
     !pip install bert-tensorflow
     !pip install bert-for-tf2
     !pip install sentencepiece
     !pip install shap
     !pip install transformers
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow_hub as hub
from sklearn import model_selection
from sklearn import metrics
import keras

from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint

from bert import BertModelLayer

import tokenization
import shap
import transformers
```

Figura 5.2.2 Cargar las dependencias requeridas.

5.2.3 Definir funciones auxiliares

En esta sección se definen funciones para utilizar los encoders de BERT y para construir modelos, estas funciones se usan en los pasos posteriores, en la figura

```python
def bert_encode(texts, tokenizer, max_len=160):
    all_tokens = []
    all_masks = []
    all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)

        text = text[:max_len-2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)

        tokens = tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] * pad_len
        segment_ids = [0] * max_len

        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)

    return np.array(all_tokens), np.array(all_masks), np.array(all_segments)
```

```python
def build_model(bert_layer, max_len=160):
    input_word_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_word_ids")
    input_mask = Input(shape=(max_len,), dtype=tf.int32, name="input_mask")
    segment_ids = Input(shape=(max_len,), dtype=tf.int32, name="segment_ids")

    _, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])
    clf_output = sequence_output[:, 0, :]
    out = Dense(1, activation='sigmoid')(clf_output)

    model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=out)
    model.compile(Adam(lr=2e-6), loss='binary_crossentropy', metrics=['accuracy',keras.metrics.Precision(), keras.metrics.Recall(), keras.metrics.TruePositives()])
```

Figura 5.2.3 Cargar las dependencias requeridas.

5.2.4 Cargar BERT desde Tensorflow Hub

TensorFlow ofrece una colección de flujos de trabajo para desarrollar y entrenar modelos mediante Python o JavaScript, y poder implementarlos con facilidad en la nube, de forma local, en el navegador o en el dispositivo, más allá del lenguaje que se use. Es por eso que BERT se carga desde Tensorflow.

```python
%%time
module_url = "https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/1"
#module_url = "https://tfhub.dev/tensorflow/bert_en_uncased_L-24_H-1024_A-16/1"
bert_layer = hub.KerasLayer(module_url, trainable=True)
```

Figura 5.2.4 Cargar las dependencias requeridas.

5.2.5 Cargar datos y Codificación de texto

En esta sección se cargan los conjuntos de entrenamiento, prueba y valor, luego se carga el tokenizador y se codifican los conjuntos para usarse en posteriores pasos, en la figura 5.2.5 se puede observar que los archivos se cargan desde un contenedor en Google Drive, además se especifica el tipo de encoding que se tiene que usar por la librería pandas, en este caso es latín1 ya que los datos contienen caracteres exclusivos del abecedario latino.

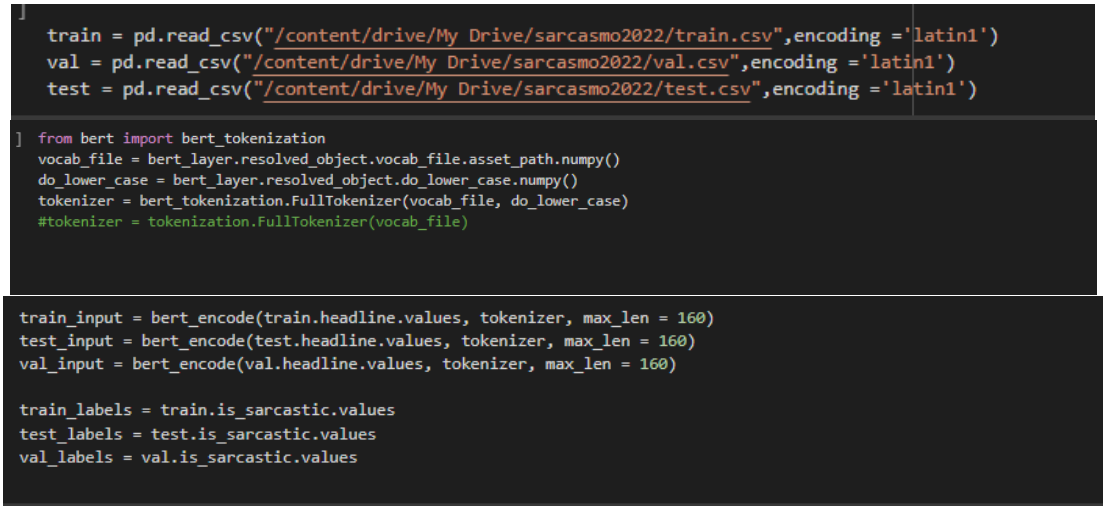
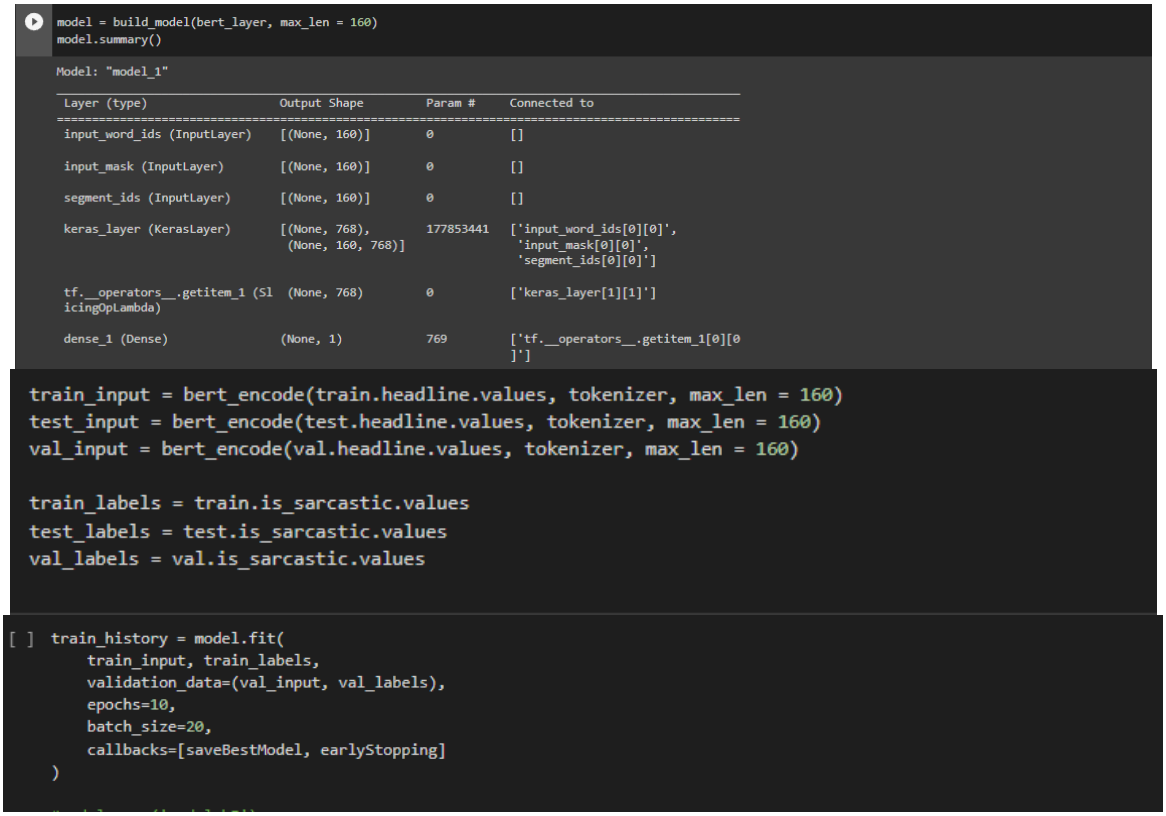


Figura 5.2.5 Cargar datos y Codificación de texto

5.2.6 Construir modelo

En esta sección del código como se muestra en la figura 5.2.6 se usan las funciones creadas en pasos posteriores para crear modelos y mejorarlos hasta conseguir un modelo idóneo para las pruebas posteriores, para esto se usa un método llamado Early stopping, un método que nos permite especificar una gran cantidad arbitraria de épocas de entrenamiento y detener el entrenamiento una vez que el rendimiento del modelo deja de mejorar en un conjunto de datos de validación/retención.



```
model = build_model(bert_layer, max_len = 160)
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_word_ids (InputLayer)    [(None, 160)]        0           []

 input_mask (InputLayer)        [(None, 160)]        0           []

 segment_ids (InputLayer)       [(None, 160)]        0           []

 keras_layer (KerasLayer)       [(None, 768),        177853441   ['input_word_ids[0][0]',
                                 (None, 160, 768)]                'input_mask[0][0]',
                                                                  'segment_ids[0][0]']

 tf.__operators__.getitem_1 (Sl  (None, 768)         0           ['keras_layer[1][1]']
 icingOpLambda)

 dense_1 (Dense)                (None, 1)            769         ['tf.__operators__.getitem_1[0][0
                                                                 ]']
```

```
train_input = bert_encode(train.headline.values, tokenizer, max_len = 160)
test_input = bert_encode(test.headline.values, tokenizer, max_len = 160)
val_input = bert_encode(val.headline.values, tokenizer, max_len = 160)

train_labels = train.is_sarcastic.values
test_labels = test.is_sarcastic.values
val_labels = val.is_sarcastic.values
```

```
[ ]  train_history = model.fit(
        train_input, train_labels,
        validation_data=(val_input, val_labels),
        epochs=10,
        batch_size=20,
        callbacks=[saveBestModel, earlyStopping]
    )
```

Figura 5.2.6 Construir modelo

5.2.7 Entrenamiento con BERT

Utilizando la plataforma COLAB se construye el modelo BERT y se guarda el modelo, posteriormente se limpia el ambiente de programación, se cargan los archivos de corpus originales y se usa BERT. Luego se guarda el modelo para la evaluación.

5.2.8 Código de evaluación con BERT

Como se puede apreciar en la figura 5.2.8, la parte final del modelo de predicción consta de instrucciones para predecir el comportamiento del modelo, posteriormente se obtiene el rendimiento del modelo obteniendo parámetros como la precisión, la medida F1 y la exactitud.

```
test_pred = model.predict(test_input)

test_pred = test_pred.round().astype(int)
print (test_pred,test_labels)

#print (test_input)
```

```
[ ]
```

```
[ ] recall = metrics.recall_score(test_labels,test_pred)
    precision = metrics.precision_score(test_labels,test_pred)
    f1_score = metrics.f1_score(test_labels,test_pred)
    accuracy = metrics.accuracy_score(test_labels,test_pred)
    loss = metrics.log_loss(test_labels,test_pred)
```

```
[ ] print('Loss:',loss)
    print('Accuracy:',accuracy)
    print('Precision:',precision)
    print('Recall:',recall)
    print('f1 score:',f1_score)
```

Figura 5.2.8 Construir modelo

Capítulo 6. Experimentación y resultados

Basado en el capítulo anterior el sistema se encuentra dividido en pasos o módulos. También se implementan recursos lingüísticos auxiliares en diferentes tareas. En este capítulo se describe la estructura del sistema y los recursos usados.

6.1 Secuencia lógica general del sistema

De acuerdo con el análisis y diseño de la aplicación, se determinó que la secuencia ideal fuera la siguiente:

- Iniciar programa
- Importación de recursos (diccionarios y listas)
- Se importa corpus general
- Preprocesamiento del texto
  - Normalización a minúsculas
  - Eliminación de números
  - Eliminación de palabras de enlace
  - Eliminación de comillas y sustitución de palabras clave
  - Eliminación de signos de puntuación
  - Eliminación de palabras no reconocidas
- Extracción de características
  - Exploración de relaciones
  - Creación de diccionario basado en frecuencias de relaciones
- Creación de corpus de entrenamiento y prueba
- Clasificar textos del corpus de prueba
  - Preprocesar texto
  - Analizar y aplicar reglas a texto
  - Determinar polaridad

En la figura 6.1 se ilustra la secuencia que siguió el sistema, Este sistema se fue usando con diferentes corpus a lo largo de la experimentación.

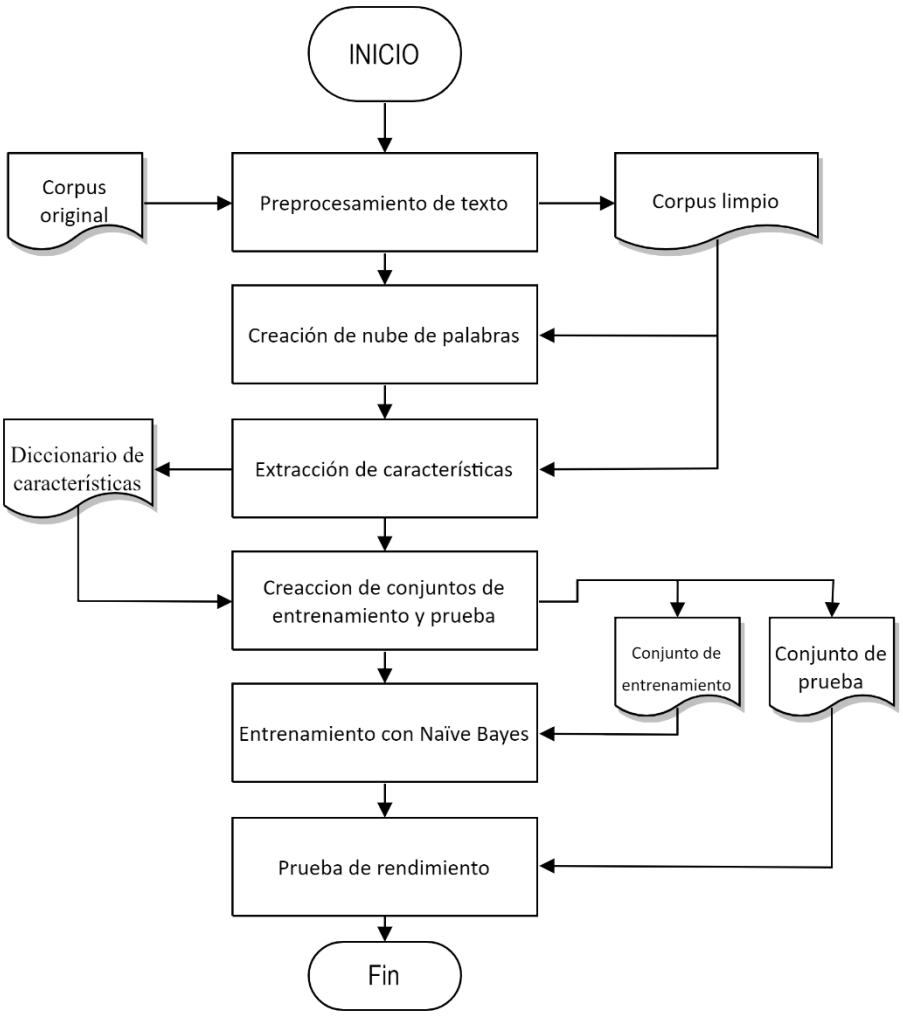


Figura 6.1 Secuencia lógica general del sistema

Se puede observar cómo módulos principales al preprocesamiento, la extracción o sección de características y la clasificación de los textos, mientras que la eliminación de texto no útil, normalización de las palabras y corrección de palabras son submódulos del preprocesamiento.

6.2 Diccionario de características

Para llegar a clasificar textos cortos se prefirió el uso de palabras como características, para ello se utilizó el corpus creado desde 0 y mediante el código se detectaron las relaciones de palabras con mayor frecuencia a lo largo del corpus, esta fue la fuente para la corrección y normalización de las palabras.

Debido al tamaño del corpus se optó por implementar como requisito para tomar en cuenta una relación que esta se repitiera como mínimo solo 10 veces a lo largo de todo el corpus, esto debido a que en los primeros experimentos, este número era mayor lo que provocaba que muy pocas relaciones se tomarán en cuenta, lo que a su vez afectaba en gran manera el juicio final, dando como resultado una exactitud de alrededor de 55%, en el siguiente capítulo se muestran con mayor detalle las pruebas y resultados obtenidos.

6.3 Corpus de entrenamiento y pruebas

Como se mencionó anteriormente, se utilizaron varios corpus para correr las pruebas, a continuación, se da una descripción general del corpus y se amplía esta descripción. A lo largo del proyecto se consiguieron varios corpus y se crearon otros para probar el modelo de clasificación, también se creó un nuevo corpus desde 0 con comentarios solo en español sobre reseñas de autos, el propósito principal de este corpus fue evaluar el modelo creado con el corpus más extenso, pero a lo largo del proceso de experimentación se aumentó el número de muestras en este para realizar los conjuntos de entrenamiento y prueba solo de este nuevo corpus.

El primer corpus que se consiguió fue uno bastante extenso con un número considerable de comentarios, el problema con este corpus extenso fue el idioma, esto debido a que se incluían diversos comentarios en inglés, alemán, italiano y otros idiomas aparte del español, además el contexto de los comentarios no era exclusivamente dirigido a videos de reseña de autos en YouTube por lo que al momento de querer evaluar el modelo creado con el corpus nuevo y exclusivo en español no daba buenos resultados.

Después se consiguió otro corpus, en este caso si era en español, su contenido se basaba en comentarios etiquetados de un capítulo de la famosa serie animada South Park como sarcásticos o no sarcásticos. La estrategia con este corpus fue combinarlo con el corpus nuevo y de ahí sacar una parte para entrenamiento y otra parte para evaluación. La última estrategia fue crecer el corpus creado desde 0 con muchos comentarios en español de usuarios en YouTube exclusivamente, eliminar todos los comentarios en inglés, y combinar los mejores comentarios del corpus de South Park y de ahí sacar una parte para entrenamiento y otra parte para evaluación.

6.4 Estándar de oro

Partiendo de que se utiliza un corpus etiquetado tanto para entrenamiento como para pruebas el estándar de oro o *gold standard* es la clasificación dada a cada documento por la SEPLN. En el caso de la sección de entrenamiento del corpus, las etiquetas se encuentran dentro de archivos CSV junto al comentario en forma de texto. A continuación, un ejemplo de etiquetado.

ID	Comentario	label
T1	No todos tienen casi medio millón para comprarse uno. México ya debería aplicarse en los autos eléctricos	non-sarcastic
T2	¡CHULADA DE CARRO!	non-sarcastic
T3	No deja de ser una marca con poco valor de reventa. ¿Está fabricado en Brasil?	non-sarcastic
T4	César ¿Qué garantía tiene esta SUV? No está nada mal, aunque uno mirando por ahí, no se sabe realmente por ejemplo cuales son los reales competidores, uno creería que son Tracker, Kicks etc, pero parecieran ser Qashqai, Compass etc.	non-sarcastic
T5	Está súper fregón, es un sueño que pocos logran, pero bueno soñar no cuesta nada y la vida da muchas sorpresas	non-sarcastic
T6	Audi es la gorda mamona que no rola la tarea, Ferrari es la de intercambio buena onda	non-sarcastic
T7	Así que un @ferrari? Sería bueno que tome nota la gente de @AudideMexico	non-sarcastic
T8	460 mil pesos por un crossover con el motor 1.6? en serio es verle la cara al consumidor.	non-sarcastic
T9	Cara, bonita, con mecánica nada asombrosa, el que la compre será por gusto y diseño. Yo me quedo con mi Vitara turbo manual, mayor torque, mejor respuesta, menor precio.	non-sarcastic
T10	Cara por tan poco, como todo lo ultimo que ha vendido.	non-sarcastic
T11	Manual de 5? 1.6? Que acaso estamos en 1998?	non-sarcastic
T12	Si tuviera el dinero para comprarme un ferrari, lo último que me compraría sería un ferrari…	sarcastic
T13	Y no le pierden?...	sarcastic
T14	Que clase de suzuki es esta??. Ja ja el diseño de s cross y vitara en parrilla.	sarcastic

Tabla 6.4.1 Ejemplo de codificación del corpus

Por otra parte, el estándar del oro para la sección de pruebas del corpus cuenta con otro archivo sin las etiquetas de las tareas a evaluar

6.5 Estructuración del sistema

Basado en la secuencia lógica del sistema y utilizando R como entorno de programación para desarrollar el sistema, la aplicación consiste en un conjunto de pasos o módulos que a su vez tienen submódulos que se apoyan en los recursos lingüísticos mencionados anteriormente.

6.6 Preprocesamiento

El preprocesamiento, como se mencionó anteriormente tiene el propósito de realizar una limpieza, corrección y normalización del texto. Este paso requiere de un submódulo de normalización a minúsculas, eliminación de números, eliminación de palabras de enlace, eliminación de comillas y sustitución de palabras clave, eliminación de signos de puntuación y eliminación de palabras no reconocidas como se muestra en la figura 6.6.

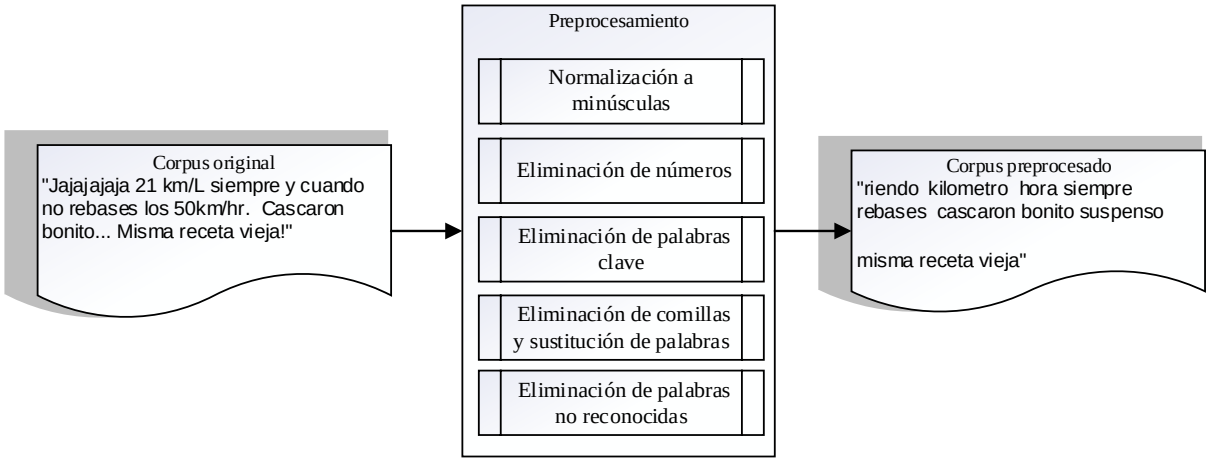


Figura 6.6 Ilustración del preprocesamiento de corpus.

6.7 Selección de características

La selección de características se realiza usando un modelo de bolsa de palabras, cada palabra es considerada como independiente y únicamente conserva su PTF o frecuencia de término por polaridad. El modelo se basa en obtener las palabras con más frecuencia dentro del corpus, sólo las palabras con determinada frecuencia serán útiles para el sistema debido a su peso de cada polaridad en el término, esto se puede observar de manera representativa en la figura 6.7 con una frase.

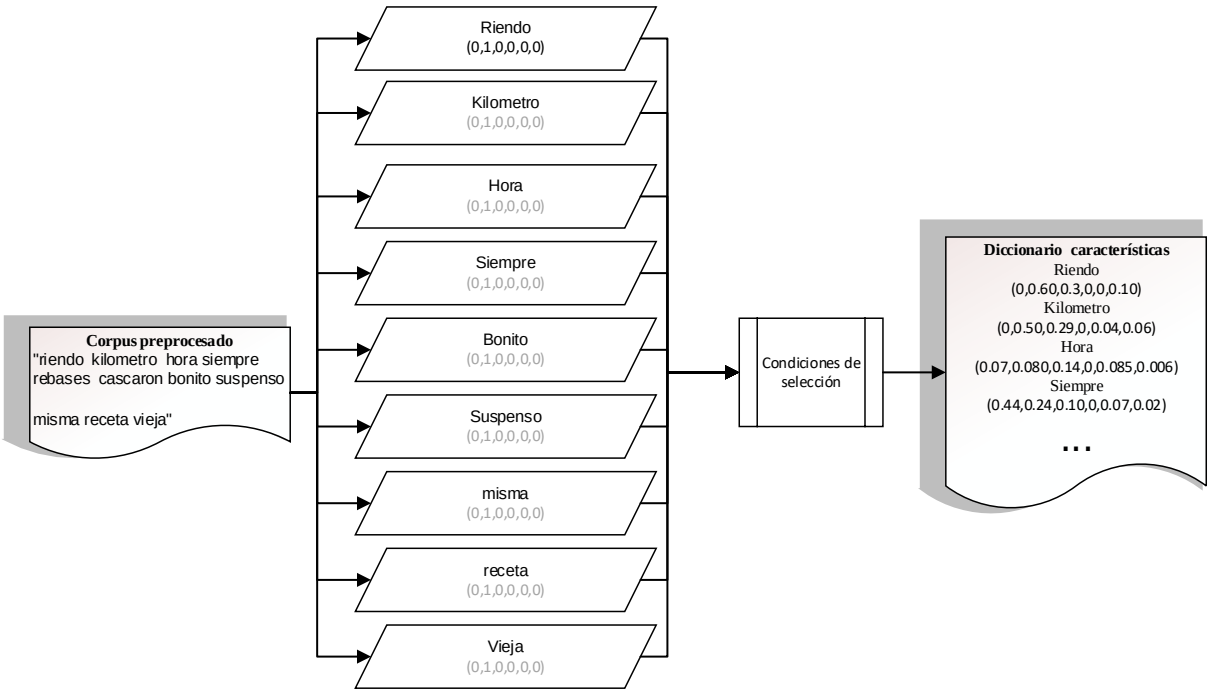


Figura 6.7 Ejemplo de paso de selección de características

6.8 Conjuntos de entrenamiento y prueba

Este paso trata sobre la preparación de muestras del conjunto de datos maestros y la división de conjuntos de entrenamiento y prueba. El inicio de este paso es limpiar el ambiente de programación en R, luego se cargan las bibliotecas necesarias. Posteriormente se usa una función creada con el fin de producir los conjuntos de entrenamiento y prueba, La función contiene un parámetro el cual divide el conjunto maestro para usar solo una parte del total de muestras para crear los dos conjuntos de entrenamiento y prueba.

6.9 Salidas

En el sistema se generan salidas a lo largo de su ejecución, las cuales son utilizadas en procesos posteriores. Estas salidas son archivos planos .dat que a continuación se listan:

1. Corpus preprocesado
2. Diccionario de características
3. Conjunto de entrenamiento
4. Conjunto de prueba

El corpus preprocesado tiene la función de poder usarse en otros modelos sin necesidad de ser limpiado de nuevo. El diccionario de características tiene como objetivo, servir como recurso para futuras implementaciones. El conjunto de entrenamiento y prueba tienen como propósito servir como una referencia en la realización de experimentos.

6.10 Configuraciones alternas

Como muchos sistemas, existen parámetros que pueden ser alterados para obtener diferentes resultados. Estos parámetros pueden afectar al diccionario de características, o al proceso de creación de los conjuntos de entrenamiento y prueba. A estos parámetros se les ha denominado parámetros de configuraciones alternas.

1. Frecuencia mínima de término
2. Umbral de diferencia de pesos de polaridades de término.
3. Ratio de proporción entrenamiento-prueba.

El parámetro “frecuencia mínima de término” tiene por objetivo servir como filtro para eliminar las palabras que son poco frecuentes y por lo tanto no aportan mucha información, por ejemplo, supongamos que un comentario contiene la palabra “cassete” y el texto fue clasificado como positivo, pero el término no ocurre otra vez, entonces la palabra “cassete” no es de utilidad. La frecuencia mínima que dio mejores resultados fue 10, es decir que un término debe tener al menos 10 menciones en el corpus de entrenamiento para tener la posibilidad de ser considerada como característica.

El umbral de diferencia de pesos de polaridades de termino se usa una vez que se obtuvieron las polaridades contrastantes, y consiste en verificar que estas no se neutralicen, es decir, supongamos que la distribución de pesos de un término fuera la siguiente [0.41 0.04 0.00 0.00 0.44 0.11], podemos observar que la carga de polaridad está presente en polaridades opuestas, por o que ambas se neutralizan, haciendo del término no muy útil para determinar la polaridad. De esta forma, si establecemos un umbral de diferencia entre las polaridades opuestas, obtenemos los términos que están mejor polarizados.

Ratio de proporción entrenamiento prueba es un parámetro de define cuanto porcentaje del corpus seleccionado en ese momento será dedicado a el conjunto de datos de prueba, y cual será destinado a entrenamiento, esto puede hacer variar los resultados ya que si el modelo es entrenado con mayor número de muestras puede llegar a conseguir puntajes más altos si el corpus es consistente en sus etiquetas de comentarios sarcásticos y no sarcásticos y al contrario si el corpus no es consistente puede empeorar el rendimiento del modelo entre más muestras sean destinadas al conjunto de datos de entrenamiento.

Capítulo 7. Análisis

En el presente capítulo está destinado a describir y analizar los experimentos hechos con los resultados obtenidos. Para ello fue necesario contar con no uno sino varios corpus sobre los cuales se realizaron los experimentos, teniendo acceso al corpus en el entorno de programación en R y en la última prueba utilizando Bert en COLAB.

Para la medición de resultados obtenidos se usó la exactitud (*Accuracy*) en la clasificación y adicionalmente se incluyen las medidas de precisión, exhaustividad y medida F (*precision, recall, F1-measure*).

7.1 Corpus usados en la evaluación

Es considerado que la selección del corpus es una tarea muy importante ya que una buena elección permite ahorrarse la laboriosa tarea de crear uno nuevo, además elegir un corpus que sea ampliamente conocido o utilizado por otros investigadores del área permite comparar resultados y aproximaciones abordadas, en este caso se decidió abordar todas las posibilidades, ya que se necesitaba saber si un corpus totalmente encausado con el contexto objetivo tendría mejores resultados que otros sin ese contexto se decidió utilizar más de un corpus en todas las pruebas.

A lo largo del proyecto se experimentó con diferentes corpus ajustando los parámetros configurables en los módulos del sistema y se creó un corpus pequeño de alrededor de 260 comentarios desde 0 con el fin de ser usado como conjunto de prueba, a continuación, se describen cronológicamente todos y cada uno de los experimentos con sus resultados.

Corpus multilenguaje: Este corpus es una recopilación de comentarios en twitter en diferentes idiomas, tanto alemán, inglés, español, italiano y portugués, con más de 10000 comentarios etiquetados como sarcástico y no sarcástico, un volumen considerable, pero sin contexto y ni siquiera en el idioma en el que se desea hacer la prueba.

Corpus SP: comentarios su contenido se basa en comentarios etiquetados de un capítulo de la famosa serie animada South Park como sarcásticos o no sarcásticos, este corpus es muy pequeño alrededor de 100 comentarios.

Corpus Original 100 muestras: Este corpus fue creado desde 0 con muchos comentarios en español de usuarios en YouTube exclusivamente. La polaridad de los comentarios se distribuye en 2 etiquetas: sarcastic y non-sarcastic las cuales se detallan en la tabla 7.1 y figura 7.1.

Corpus Original 255 muestras: Este corpus fue creado desde 0 con muchos comentarios en español de usuarios en YouTube exclusivamente. La polaridad de los comentarios se distribuye en 2 etiquetas: sarcastic y non-sarcastic las cuales se detallan en la tabla 7.1 y figura 7.1.

Polaridad	Número de comentarios	Porcentaje
sarcastic	50	50%
non-sarcastic	50	50%

Tabla 7.1 Distribución de comentarios del corpus por polaridad en la sección de entrenamiento.

Figura 7.1 gráfica de distribución del conjunto de entrenamiento

Como se puede observar en la tabla 7.2 en esta distribución de polaridades se buscó tener un 50% en cada categoría para eliminar cualquier disturbio en la clasificación del modelo en la fase de prueba.

Polaridad	Número de comentarios	Porcentaje
sarcastic	10	50%
non-sarcastic	10	50%

Tabla 7.2 Distribución de comentarios del conjunto por polaridad en la sección de prueba.

7.2 Exactitud (*Accuracy*)

La principal medida de evaluación que se utilizó para poder clasificar los sistemas de participantes fue la exactitud, es decir que porcentaje de los textos clasificados estuvieron correctos, esta medida se obtiene mediante la siguiente ecuación (17).

$$Exactitud = \frac{verdaderos\ positivos + verdaderos\ negativos}{verdaderos\ positivos + falsos\ positivos + falsos\ negativos + verdaderos\ positivos} \quad (17)$$

La cual puede ser vista como en la ecuación (18) aplicada al proyecto:

$$Exactitud = \frac{verdaderos\ positivos}{total\ de\ muestras\ del\ corpus} \quad (18)$$

Donde falsos positivos es el número de textos clasificados correctamente por el sistema en la categoría correcta y el total de muestras del corpus es el número de textos del corpus que se usaron para realizar la prueba.

7.3 Precisión

La precisión es usada para medir la proporción de los textos clasificados correctamente por el sistema, dicho de otra forma, mide el porcentaje de documentos clasificados correctamente del conjunto total de documentos que el sistema evalúa. Para obtener la medición por clase o etiqueta la evaluación se realiza exclusivamente sobre el número total que el sistema asignó esa etiqueta de forma correcta o incorrecta.

La ecuación (19) para obtener la precisión de los textos evaluados por un sistema que se utilizó en el TASS es la siguiente:

$$Precision = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + falsos\ positivos} \quad (19)$$

Donde Verdaderos positivos es el número de textos clasificados correctamente en su categoría y falsos positivos son los comentarios clasificados por el sistema de manera incorrecta.

7.4 Exhaustividad (*Recall*)

La exhaustividad es usada para determinar qué porcentaje de los textos seleccionados forman parte del conjunto objetivo. Es decir, el porcentaje de documentos que el sistema clasifica correctamente sobre el conjunto total de documentos que se deben clasificar. En el caso de la exhaustividad por polaridad, se comprueban los documentos clasificados correctamente sobre el conjunto de documentos total que pertenecen a polaridad evaluada.

La ecuación (20) para obtener la medida de exhaustividad por los sistemas establecida en el TASS es la siguiente:

$$Exhaustividad = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ negativos} \quad (20)$$

Donde Verdaderos positivos es el número de textos clasificados correctamente en la categoría que les corresponde y falsos negativos son los textos clasificados con una etiqueta diferente a las establecidas.

7.5 Medida F (*F1-measure*)

Finalmente, la medida F combina la precisión y la exhaustividad, es decir, se obtiene una medida de desempeño general, en la que no se sacrifica la precisión o la exhaustividad, es decir, se obtiene una medida de desempeño general, en la que no se sacrifica la precisión o la exhaustividad. En el TASS se usa la ecuación (21) para obtener la medida F en los sistemas es la siguiente:

$$F1 = 2 * \frac{Precision * exhaustividad}{precision + exhaustividad} \quad (21)$$

Donde exhaustividad y precisión fueron obtenidos previamente y 2 es una constante.

Resultados obtenidos con los diferentes corpus y estrategias.

7.6.1 Prueba 1 Corpus Multilenguaje como entrenamiento.

En la tabla 7.6.1 podemos observar un mal desempeño de clasificación de comentarios sarcásticos en la matriz de confusión, además de una exactitud muy baja, alrededor de 63%.

Clase actual	No sarcástico	Sarcástico
No sarcástico	78	30
Sarcástico	49	61

Tabla 7.6.1 Matriz de confusión

Exactitud	0.63
Precisión	0.72
Exhaustividad	0.55
F measure	0.62

Tabla 7.6.1 Parámetros

En este resultado se puede observar el pobre nivel de desempeño en este corpus, debido a que a pesar de tener un volumen de muestras considerable de más de 10000 muestras la mayoría de los datos están en otro idioma, ya sea inglés, alemán o portugués y esto a su vez provoca que el paso de entrenamiento del modelo sea deficiente, esto a pesar de tener casi todo el corpus (98%) dedicado al entrenamiento.

7.6.2 Prueba 2 Corpus SP como entrenamiento.

Los resultados de esta prueba mejoraron respecto a la pasada, pero sigue teniendo problemas para clasificar los comentarios No sarcásticos, esto debido a que muchos comentarios en este corpus constan solo de una palabra, lo cual para el modelo hace que sean descartadas, lo que

a su vez provoca que existan pocas características significativas, al final esto provoca que el juicio se incline mucho hacia un solo lado como se muestra en la tabla 7.6.3 y 7.6.4.

Clase actual	No sarcástico	Sarcástico
No sarcástico	22	4
Sarcástico	10	16

Tabla 7.6.3 Matriz de confusión

Exactitud	0.75
Precisión	0.61
Exhaustividad	0.84
F measure	0.71

Tabla 7.6.4 Parámetros

Estos resultados tuvieron una mejoría respecto al corpus multilenguaje debido a que el corpus en su totalidad está integrado por comentarios en español, pero sigue siendo insuficiente su información debido a que en muchas muestras el contenido de los comentarios se reduce a frases monosilábicas o comentarios sin ningún tipo de relación con el mundo automotriz.

7.6.3 Prueba 3 Corpus original sin Reglas de preprocesamiento como entrenamiento y prueba.

Clase actual	No sarcástico	Sarcástico
No sarcástico	22	4
Sarcástico	10	16

Tabla 7.6.5 Matriz de confusión

Exactitud	0.730769
Precisión	0.615385
Exhaustividad	1.142857
F measure	0.8

Tabla 7.6.6 Parámetros

Estos resultados son muy parecidos a los anteriores como se puede observar en la tabla 7.6.5, y 7.6.6 pero aquí se observó que al ser un corpus muy pequeño y teniendo unas frecuencias mínimas muy altas, el modelo solo considero muy pocas relaciones para identificar el sarcasmo o no sarcasmo en los comentarios lo cual provocaba que la cantidad de clasificaciones de no sarcástico se elevara drásticamente debido a que si el modelo llegaba a detectar que no había en un comentario las pocas relaciones tomadas por las frecuencias mínimas este comentario era clasificado como no sarcástico, provocando que no se clasificaran correctamente muchos comentarios sarcásticos.

7.6.4 Prueba 4 Corpus original con 100 muestras agregando reglas de sustitución en preprocesamiento.

Clase actual	No sarcástico	Sarcástico
No sarcástico	23	3
Sarcástico	10	16

Tabla 7.6.7 Matriz de confusión

Exactitud	0.75
Precisión	0.615385
Exhaustividad	1.230769
F measure	0.820513

Tabla 7.6.8 Parámetros

En la tabla 7.6.7 y 7.6.8 Se puede observar una mejoría respecto a la primera experimentación, al ser un corpus completamente en español y enfocado hacia automóviles, inclusive siendo un número muy reducido de muestras se obtuvieron mejores resultados.

7.6.5 Prueba 5 Corpus original con 100 muestras, agregando reglas de sustitución en preprocesamiento.

Clase actual	No sarcástico	Sarcástico
No sarcástico	24	2
Sarcástico	10	16

Tabla 7.6.9 Matriz de confusión

Exactitud	0.769231
Precisión	0.615385
Exhaustividad	1.333333
F measure	0.842105

Tabla 7.6.10 Parámetros

Al agregar reglas de signos de admiración, reglas de comillas, regla de risas y la palabra suspenso en lugar de "…". Como entrenamiento y prueba el modelo pudo obtener más información útil y crear características relevantes para la clasificación, sigue existiendo una preponderante preferencia por clasificar comentarios sarcásticos como no sarcásticos, pero en menor medida, esto lo podemos ver en las tablas 7.6.9 y 7.6.10.

7.6.6 Prueba 6 Corpus original con 266 muestras, agregando reglas de sustitución en preprocesamiento.

Clase actual	No sarcástico	Sarcástico
No sarcástico	32	2
Sarcástico	10	24

Tabla 7.6.11 Matriz de confusión

Exactitud	0.82
Precisión	0.94
Exhaustividad	0.7
F measure	0.80

Tabla 7.6.12 Parámetros

Como se observa en las tablas 7.6.11 y 7.6.12 al tener más muestras para distribuir entre los conjuntos de entrenamiento y prueba se pudieron extraer más características las cuales sirvieron para ayudar a identificar de mejor manera los comentarios sarcásticos, los problemas que se tuvieron al añadir las reglas de sustitución de texto en la parte de pre procesamiento fueron las ocasiones donde no se lograban sustituir correctamente los elementos originales, lo cual a su vez provocaban perdida de información en la creación de características.

En los resultados de las tablas 7.6.11 y 7.6.12 se puede observar la diferencia entre los pesos al tener pocas y muchas muestras, además las reglas para detectar risas ayudan en gran medida para identificar la intención del comentario, pero cuando el modelo no es capaz de identificar las risas este no puede clasificar correctamente la intención del comentario por lo que el juicio se ve afectado negativamente. El contexto de las palabras es difícil de identificar debido a que el modelo no toma todas las palabras anteriores

7.6.7 Prueba 7 Corpus original con 266 muestras, utilizándolo como entrenamiento y prueba con BERT.

Clase actual	No sarcástico	Sarcástico
No sarcástico	14	0
Sarcástico	1	15

Tabla 7.6.13 Matriz de confusión

Exactitud	0.966667
Precisión	1.0
Exhaustividad	0. 9375
F measure	0.9677

Tabla 7.6.14 Parámetros

Al utilizar el ultimo corpus en el modelo 2 basado en Deep learning, se tuvieron que hacer algunas adecuaciones para lograr utilizar el modelo, pero se puede apreciar una clara mejoría en el desempeño de clasificación, claramente se ve en la tabla 7.6.13 y 7.6.14 como el problema de clasificación errónea ya no está presente.

Estrategia	BERT	Naive Bayes
Exactitud	0.966667	0.82
Precisión	1	0.94
Exhaustividad	0. 9375	0.7
F measure	0.9677	0.8

Tabla 7.6.15 Concentrado de Naive Bayes y BERT

Como se aprecia en la tabla concentrada 7.6.15 el desempeño en la clasificación que tiene BERT con sus redes Transformer es superior al mejor caso de Naive Bayes con el corpus limpio, y reglas añadidas para ayudar a la clasificación.

Análisis de desempeño en diferentes corpus

En esta sección se detalla la forma en la que se llevaron a cabo las pruebas con las diferentes estrategias seleccionadas, así como sus resultados ajustando algunos parámetros como el número de muestras totales en el corpus o ratio de muestras elegidas para los conjuntos de entrenamiento y prueba. Además, en las estrategias donde el corpus no era tan grande se ajustaron los valores que determinan las frecuencias mínimas de término necesarias para hacer valida una relación de palabras, sumado a esto, en estos corpus se hicieron pruebas con 100, 200 y 266 muestras con el paso del tiempo mientras se conseguían más muestras. Al realizar las pruebas con todas las estrategias propuestas se llegó a varias conclusiones.

Corpus multilenguaje

Al realizar las pruebas con este corpus no se pudieron obtener buenos resultados debido a la variedad de idiomas que existían y la poca relación en los contextos.

2.0 Corpus SP

Al analizar su contenido se observaron muchas muestras que tenían muy poca información útil, esto debido a que el corpus fue creado como una transcripción de audio a texto de un capítulo de la serie Southpark, por lo que había muchos comentarios que era monosilábicos o con muy pocas palabras que al final eran eliminadas por el sistema por ser *Stop-words* y si alcanzaban a pasar al módulo de selección de características al ser poco frecuentes el sistema nos los tomaba en cuenta, esto a su vez provocó un fuerte desbalance de peso en las características, dándole mucha importancia a pocas características, lo que a su vez inclinaba de gran manera al juicio de todas los comentarios en el conjunto de pruebas juzgando todo como sarcástico o todo como no sarcástico debido a las pocas características obtenidas para el entrenamiento.

3.0 Corpus original

Al realizar las pruebas con este corpus se fueron recopilando más muestras a lo largo del proyecto de manera escalonada se realizaron pruebas con el corpus compuesto de 100, 200 y 250 comentarios. Debido al número tan bajo de muestras se cambió la frecuencia mínima a 10 muestras para obtener características relevantes en un mayor número.

4.0 Corpus original + BERT

Al realizar las pruebas con el corpus creado desde 0 con su número máximo de muestras, se decidió migrar el corpus a BERT con la finalidad de experimentar con él. Se tuvieron que hacer algunas adecuaciones como separar manualmente los conjuntos de entrenamiento y prueba además de crear un nuevo archivo de valores sustituyendo las etiquetas con un 1 y 0 en lugar de sarcastic y non-sarcastic. El proceso se llevó a cabo a través de la plataforma COLAB.

Análisis de pruebas en general

La frecuencia mínima de término es un parámetro que afecta de manera significativa al juicio del modelo creado, ya que si este valor es muy exigente solo muy pocas palabras serán vistas como características y por lo tanto solo unas pocas definirán el juicio al momento de clasificar cada comentario agregando un peso significativo a estas palabras. Esto se vio reflejado en las pruebas donde en el proceso de ingeniería de características no se modificó el rango mínimo de frecuencia con el corpus de 200 muestras, las características inclinaban de manera casi uniforme todos los juicios a sarcástico o no sarcástico dependiendo cuál de las dos tenía el mayor número de muestras etiquetadas en los corpus.

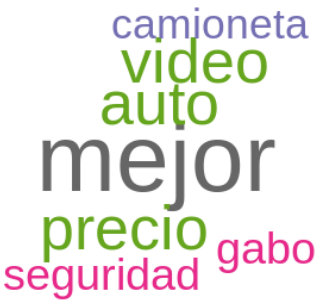


Figura 7.7.1 Imagen de palabras que forman partes de las características con 200 muestras sin modificar la frecuencia mínima necesaria

En la figura 7.7.1 se observa la poca cantidad de palabras seleccionadas por el modelo al tener una frecuencia mínima necesaria muy alta, la cual discriminaba las palabras con menor frecuencia y al contrario en la figura 7.7.2 se muestra una frecuencia mínima muy baja la cual deja pasar muchas palabras como características.

Figura 7.7.2 Baja frecuencia mínima. Imagen de palabras que forman parte de las características con 200 muestras modificando la frecuencia mínima necesaria.

Este problema de balance de pesos también se presenta cuando el número de muestras es muy bajo, cuando se realizaron las pruebas con un corpus de únicamente 100 muestras el juicio de nuevo se inclinaba totalmente hacia un lado u otro dependiendo del número de muestras mayoritario en el corpus, esto se puede apreciar en la figura 7.7.3.

Figura 7.7.3 Nube de palabras con pocas muestras

Imagen de prueba con 100 muestras

Por lo tanto, el número de muestras en el corpus tiene que estar fuertemente relacionado con la frecuencia mínima para considerar las características relevantes extraídas.

Correcto encoding

Al principio, todos los corpus estaban guardados con un encoding no apto para comentarios en español, ya que el tipo de encoding original no soportaba caracteres propios del idioma como tildes, la letra “ñ” o los acentos, al realizar pruebas con el encoding original en los archivos CSV y con un encoding “latin1” se obtuvieron mejores resultados con el encoding “latin1”.

Trabajos por realizar

La aproximación que se ha propuesto consiguió buenos resultados, sin embargo, aún se puede tener mejoras por lo que algunos trabajos a futuro pueden ser los siguientes:

Experimentar con diferentes tamaños de n-gramas.

Considerar emoticones, onomatopeyas e intensificadores.

Realizar pruebas con BERT utilizando corpus preprocesado.

Lo anterior con el propósito de crear un sistema más robusto que pueda resolver con un mejor rendimiento las tareas, sobre todo en clases de bajos resultados como la identificación de neutralidad.

Conclusiones

Con los primeros corpus fue muy evidente que la falta de contexto en el conjunto de entrenamiento afectaba las pruebas debido a que el conjunto de prueba tenía características que no existían en el conjunto de entrenamiento, al probar el sistema con el corpus creado desde 0 el porcentaje de exactitud mejoró pero se puede analizar que el sistema probado muestra una clara falta de contexto, debido al propio diseño del sistema, la predicción le da independencia a cada palabra o conjunto de palabras y deja fuera del juicio la combinación de las características extraídas. El juicio es tomado únicamente por las características que se encuentran en cada frase, pero no toma en cuenta la combinación de estas ni el probable contexto que esto puede tener.

Al realizar las pruebas de clasificación con BERT se comprobó una relación importante entre la exactitud y el contexto del comentario, BERT al ser un sistema que toma en cuenta el contexto de las palabras de manera bidireccional, puede identificar de mejor manera el tono de la frase en su totalidad gracias a sus redes Transformer que cuentan con bloques atencionales que toman en cuenta la posición de las palabras dentro de la frase y la relación entre frases dentro de un mismo comentario. Bert por sí solo no puede identificar el sarcasmo en videos de reseñas de autos, pero gracias al corpus que se recopilo y que fue etiquetado ahora la combinación de ambas partes pueden lograr identificar casi en su totalidad aquellos comentarios que se realizan de manera sarcástica con el fin de mostrar atributos negativos sobre los vehículos, compañías o incluso sobre el presentador del video.

Capítulo 8: Referencias

Ameeta Agrawal, A. A. (s.f.). *Unsupervised Emotion Detection from Text using Semantic and Syntactic Relations.* York University, Toronto, Canada: Department of Computer Science and Engineering.

Ankit Ramteke, P. B. (2013). Detecting turnarounds in sentiment analysis: Thwarting. *Proceedings of ACL*.

Banko, M. a. (2001). Scaling to very very large corpora for natural language disambiguation.

Basavanna, M. (2000). *Dictionary of psychology.* Allied Publishers.

Bi, F. a. (2012). Customer service 2.0: Where social computing meets customer relations. *IEEE Computer*, 45, 11, 93–95.

Binali, H. (2010). *Computational Approaches for Emotion Detection in Text.* Perth , Australia : Curtin University of Technology .

Brill, E. (1992). A simple rule-based part of speech tagger.

Butler, Q. (1953). The instituto oratoria of quintilian. with an english.

CC Liebrecht, F. K. (s.f.). The perfect solution for detecting sarcasm in tweets #not.

Channon, S. P. (2004). Brain and Language. *Social cognition after head injury: sarcasm and theory of mind*.

Cheang, H. S. (2011). Recognizing sarcasm without language: A crosslinguistic study of english and cantonese. *Pragmatics & Cognition*, 19, 2.

Chin. (2011). The science of sarcasm? yeah, right. *www.smithsonianmag.com/science-nature/*.

Dmitry Davidov, O. T. (2010). Semi-supervised recognition of sarcastic sentences. *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, 107–116.

Domingos, P. (2012). A few useful things to know about machine learning.

Dress, M. L. (2008). Regional variation in the use of sarcasm. *Journal of Language and Social Psychology*, 27, 1, 71–85.

Ekman, P. &. (2005). *What the face reveals. 2nd.* New York: Oxford.

Ellen Riloff, A. Q. (2013). Sarcasm as contrast between a positive sentiment and negative situation. *Proceedings of the 2013 Conference on Empirical Methods in Natural*, 704–714.

Filatova, E. (2012). Irony and sarcasm corpus generation and analysis using crowdsourcing.

Gibbs, R. W. (2000). Irony in talk among friends. *Metaphor and symbol*, 15, 1-2, 5–27.

Giora, R. (1995). On irony and negation. *discourse processes*.

Grice, H. P. (1975). *Logic ad conversation.*

Hastie, T. T. (2010). *The Element of Statistical Learning.* Springer.

Jodi Eisterhold, S. A. (2006). Reactions to irony in discourse: Evidence for the least disruption principle. *Journal of Pragmatics*, 1239–1256.

Joseph Tepperman, D. R. (2006). sarcasm recognition for spoken dialogue systems. *INTERSPEECH*.

Katz, J. D. (2012). Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, 459–480.

Katz, J. F. (1991). *Bimodal expression of emotion by face.* 6th ACM international conference on Multimedia:.

M. Chunling, H. P. (2005). *Emotion Estimation and Reasoning Based on Affective Textual Interaction.* Springer Berlin : Affective Computing and Intelligent.

McDonald, S. (1999). Exploring the process of inference generation in sarcasm: A review of normal and clinical studies. *Brain and Language*, 68, 3, 486 – 506.

Michie, D. S. (1994). Machine Learning, Neural and Statistical Classification. *Broke Books*.

Mitchell, T. M. (1977). Machine Learning. *McGraw-Hill*.

Nicolosi, N. (2008). Feature selection methods for text classification.

Oller, J. J. (1972). Scoring methods and difficulty levels for cloze tests of proficiency in english as a second language. *The Modern Language Journal*, 56, 3, 151–158.

Paula Carvalho, L. S. (2009). Clues for detecting irony in user-generated contents: oh...!! it's too easy ;-). *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, 53–56.

Pedregosa, F. V. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2825–2830.

Pexman, P. M. (2013). Children's processing of emotion in ironic language.

Pexman, S. L. (2003). Context incongruity and irony processing. *Discourse Processes*, 241–279.

Porter, M. F. (1980). An algorithm fo suffix stripping. *American Society for Information Science.*

R. Cowie, E. D.-C. (2001). *Emotion recognition in human-computer.* IEEE Signal Processing Magazine.

Roberto Gonzalez-Ibanez, S. M. (2011). Identifying sarcasm in twitter: a closer look. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short*, 581–586.

Rockwell, P. (2000). Lower, slower, louder: Vocal cues of sarcasm. *Journal of Psycholinguistic Research*, 29, 5, 483–495.

Rockwell, P. (2003). Empathy and the expression and recognition of sarcasm by close relations or strangers. *Perceptual and motor skills*, 97, 1, 251–256.

Rockwell, P. a. (2001). Culture, gender, and gender mix in encoders of sarcasm: A self-assessment analysis. *Communication Research Reports*, 18, 1, 44 – 52.

Silver, J. (2014). Us secret service wants software to ”detect sarcasm” on social media. *http://arstechnica.com/tech-policy/2014/06/*.

Sperber, D. a. (1995). Postface to the second edition of relevance: Communication and cognition.

Sporer, S. L. (2016). Deception and Cognitive Load: Expanding Our Horizon with a Working Memory Model. *Frontiers in Psychology*, 7.

Thelwall, M. K. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61, 12, 2544–2558.

Toplak, M. a. (2000). *On the uses of sarcastic irony.* Journal of Pragmatics.

Wang, A. T. (2006). Neural basis of irony comprehension in children with autism: the role of prosody and context. *Brain*, 129, 4,932–943.

Warriner, A. B. (2013). Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 1–17 .

Wilson, D. (2006). The pragmatics of verbal irony. *Lingua*, 1722–1743.

Witten, I. H. (2005). *Data mining: Practical machine learning tools and technique.*

Zhu, X. (2007). Semi-supervised learning tutorial. *http://pages.cs.wisc.edu/˜jerryzhu/pub/sslicml07.pdf.*

Zloteanu, M. (2016). Emotions and Deception Detection. *Division of Psychology and Language Sciences*.

Glosario

Accuracy. O exactitud, es una medida que obtiene el porcentaje de documentos clasificados correctamente

Aldea Global. Termino introducido por sociólogo canadiense Marshall McLuhan, que describe a la sociedad como una gran sociedad mundial que supera las limitaciones de distancia para estar interconectados.

Análisis de sentimientos. Identificación y extracción de información subjetiva de un texto por medio del procesamiento del lenguaje natural.

Árbol de decisión. Es un modelo de predicción usado en la inteligencia artificial, que se construye en base a un diagrama lógico. Son similares a los sistemas de predicción basada en reglas.

Características (features). Es el nombre que se le da a los términos que se usan para describir algún objeto. En este caso son las palabras con diferentes pesos.

Corpus Dentro del campo de la lingüística, es el conjunto de datos, textos u otros materiales sobre determinada materia que pueden servir de base para una investigación o trabajo.

Corpus de entrenamiento (train). Es el nombre que se da al conjunto de textos que generalmente están etiquetados y sirven como fuente para entrenar a sistemas de cómputo por medio de aprendizaje automático o machine learning.

Corpus de prueba (test). Es el nombre que se le da al conjunto de textos que generalmente no están etiquetados y se usa para evaluar los resultados de algún sistema.

Diccionario. Libro que contiene y define una o más palabras de algún idioma. Dentro de la programación en Python, los diccionarios son una estructura de datos en la cual la indexación clave-valor se realiza por medio de claves alfanuméricas únicas.

Document frecuency thresholding . O umbral de frecuencia en el documento, se pueden ver como lo opuesto a las stopwords, debido a su baja frecuencia no aportan información.

Emoticón. Neologismo formado por las palabras emoticón e icono que usando caracteres ASCII se presentan de forma simbólica estados de ánimo o formas de mímicas de expresión.

F-score. O medida F1, es una medida que sirve para medir el desempeño de un sistema que tiene que ver con la recuperación de documentos. Esta medida requiere que se calcule previamente precisión y recall.

IDF Del inglés inverse document frecuency, frecuencia inversa de documento, es una medida que sirve para identificar que si un término es frecuente o no en una colección de documentos.

Inteligencia artificial. Área multidisciplinaria que estudia la creación y diseños de sistemas capaces de resolver problemas cotidianos por sí mismos.

Lematización Tarea del procesamiento del lenguaje natural en la que se identifica la forma canónica o lema de una palabra flexionada.

Lingüística computacional. Área donde converge la lingüística y la computación con especial interés por el estudio y la modelación del lenguaje humano mediante métodos computacionales.

Lista. Matriz unidimensional o vector, que puede contener una secuencia de datos. Las listas generalmente usan una indexación numérica que inicia en 0.

Naive Bayes. Clasificador bayesiano ingenuo, es una metodología que se apoya en la probabilidad fundamentada en el teorema de bayes. Las variables predictoras generalmente adquieren cierta independencia entre sí.

Preprocesamiento. Termino usado para indicar la necesidad de realizar tareas previas antes de la tarea principal, en este aso depuración, corrección y normalización de texto.

Preprocesamiento del lenguaje natural. Tiene especial interés por estudiar y modelar el lenguaje mediante métodos computaciones, se le puede considerar sinónimo de la lingüística computacional.

Recall. O exhaustividad, es una medida que obtiene el porcentaje de efectividad en la recuperación de documentos sobre un conjunto objetivo.

Recursos lingüísticos. Conjunto de archivos como diccionarios, tesauros o lexicón que apoyan a la solución de algún problema, ya que contienen información especializada. El spanish Emoticon Lexivcon es un ejemplo de ello.

Red social. Termino usado para nombrar a los círculos de amistades de una persona. En la actualidad este término se utiliza principalmente referirse a paginas como Facebook, Twitter, Pinterest entre otras sirven para mantener contacto visual con otras personas.

Sentimiento. Termino usado en la lingüística computacional para nombrar la expresividad subjetiva de un texto, por ejemplo: positivo, negativo, neutral, etcétera.

SEPLN sociedad española para el procesamiento del lenguaje natural, es una asociación científica sin ánimo de lucro con el objetivo de promover todo tipo de actividades relacionadas con el estudio del procesamiento de lenguaje natural

Stopwords O palabras auxiliares o vacías, son aquellas palabras que, debido a su alta frecuencia en documentos, independientemente de su temático o propósito, no aportan información

TASS. taller de análisis de sentimientos de la SEPLN, es un taller de evaluación experimental en el contexto de la sociedad española para el procesamiento del lenguaje natural.

TF. Del inglés term frecuency- inverse document frequency, combina ambas medidas por medio de las cuales se puede medir que tan común es un término en una colección de documentos.

Tokenizacion. tarea del procesamiento del lenguaje natural que consiste en dividir una oración en unidades. Cabe señalar que existen tokens compuestos como es el caso de Buenos Aires.

Web 2.0 termino usado para describir la etapa en la que se popularizo la producción de contenidos por parte de los usuarios comunes.

Xml. Del inglés extensible Markup Lenguaje, es un lenguaje de marcas utilizado para guardar datos de forma legible, permite utilizar una definición gramática similar a la de HTML