UNIVERSIDAD
DE LOS ANDES
VENEZUELA

Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes como requisito parcial para obtener el Título de Ingeniero de Sistemas

Identificación de expresiones verbales fijas de Venezuela usando aprendizaje automático

Por

Br. César David Barrios Vega

Tutor: Prof. Wladimir Rodríguez

Diciembre 2019

Identificación de expresiones verbales fijas de Venezuela usando aprendizaje automático

Br. César David Barrios Vega

Proyecto de Grado — Sistemas Computacionales, 108 páginas

Resumen: En la actualidad se desarrolla un creciente interés por la automatización, sobre todo por el reciente auge del Internet y el impacto que ha tenido en la comunicación entre las diferentes partes del mundo. Este trabajo propone el uso de estas tecnologías para la creación de una herramienta con la función de identificar expresiones verbales fijas venezolanas. Entre los problemas que presenta el procesamiento del lenguaje natural en los últimos años, se encuentra la necesidad de entender la semántica de un texto, papel que las expresiones verbales fijas toman de primera mano, pues su cualidad nada sintáctica las vuelve un objeto sumamente interesante de estudio. No obstante, la mayoría de los estudios realizados en procesamiento del lenguaje están enfocados al idioma inglés, debido a su establecimiento como lengua Franca, por esta razón, estudiar la morfología del español es un área que tiene muchos descubrimientos por hacer. El objetivo de esta investigación es el de lograr identificar expresiones verbales venezolanas conocidas, dentro de un contexto. Para esto se propone la creación de una base de conocimientos de expresiones construida mediante un diccionario de expresiones con la cuál se recopilará información. La recopilación se ha realizado utilizando la red social Twitter, pues, no solo ofrece una inmensa cantidad de datos, sino que también provee de geolocalización de dichos datos, información importante al momento de estudiar estas expresiones. Se ha adoptado un enfoque supervisado para la identificación de estas expresiones, y a partir del conjunto de datos obtenido desde Twitter se consiguieron mejores resultados al utilizar técnicas de extracción de características combinando métodos sintácticos y semánticos para los modelos entrenados.

Palabras clave: Minería de texto, Procesamiento del Lenguaje Natural, Identificación de expresiones verbales fijas, algoritmos de clasificación, extracción de características.

Este trabajo fue procesado en LaTeX.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

El ser humano a través de la historia se ha caracterizado como un ser social, valiéndose de diferentes estrategias para expresar sus pensamientos, emociones e inclusive vivencias con otros, originando los distintos tipos comunicación (verbal y no verbal); sin embargo, debido a que cada población se ha desarrollado de forma aislada, cada sociedad ha creado una forma diferente para comunicarse entre sí; un idioma propio que ha creado una barrera que debe superar cada individuo para poder entablar una conversación verbal con otro distinto a sus compañeros habituales. Debido a los procesos actuales de globalización, cada vez es más común el contacto entre distintas civilizaciones, siendo necesario entender al otro; debido a que en ello está la clave de una negociación efectiva, acuerdos beneficiosos, relaciones interpersonales exitosas, entre otros.

Esta problemática ha sido abordada mediante diferentes herramientas que sirven de traductores carentes de sentido idiomático, efectuando equivalencias de palabras entre ambos idiomas para una comprensión parcial... Si esto ha sido así para diferentes idiomas, ¿qué ha ocurrido con aquellos que utilizan el mismo idioma? ¿Pueden entenderse de manera efectiva? Es frecuente encontrar regiones que aun cuando hablan el mismo idioma, se pueden observar en ellos ciertas diferencias idiomáticas, denominadas como expresiones idiomáticas, que nacen en el seno de cada población debido a las distintas influencias históricas, sociales y culturales a las que están o han estado expuestas. Muchas de estas expresiones; tan comunes en el haber cotidiano,

suelen ser muy distintas dependiendo del lugar que provengan, por lo que presentan un obstáculo al momento de efectuar alguna interacción, en especial en la actualidad donde están presente más que nunca en las redes sociales, medios a los cuales cualquiera puede acceder y publicar con varios fines, tales como informativos, recreativos, laborales, entre otros.

Poco o nada se ha hecho al respecto para superar esta problemática, y debido a que sería accidentado (y costoso) asignar a un experto (o varios) la tarea de traducir en tiempo real, resulta más efectivo asignar un personal no humano para tal actividad, es por ello que una computadora es la más indicada para asumir esta tarea, pero debido a la poca fijación de las expresiones idiomáticas y a su significado variante dependiente del contexto, esta debe ser enseñada previamente, para que así pueda efectuar de forma eficiente su labor. Aquí es donde comienza el rol del aprendizaje automático como solución del problema.

El Aprendizaje Automático o Aprendizaje de Máquina (del inglés Machine Learning) es un tipo de inteligencia artificial (IA) que permite que un computador tenga la capacidad de aprender sin ser explícitamente programado. Se define como el proceso de programar una máquina o computador para que sea capaz de entregar resultados lo suficientemente útiles en base al uso de datos de ejemplo o experiencias pasadas [1]. En este proyecto se utilizará aprendizaje de máquina para la clasificación e identificación de expresiones verbales fijas venezolanas dentro de textos en español.

Según [2], la clasificación es el proceso de dividir el mundo en grupos de entidades cuyos miembros son similares entre sí. Análogamente, la clasificación de textos consiste en la asignación de un grupo de palabras del lenguaje natural a ciertas categorías dependiendo de su contenido. En esta investigación no se hace distinción entre las palabras categorización y clasificación en el contexto textual, de hecho, según el Diccionario de la Real Academia Española [3] se define el concepto de clasificar como “Ordenar o disponer por clases algo”, mientras que categorizar se describe como “Organizar o clasificar por categorías”; esto sugiere que son sinónimos, en contraste con eso, el autor Coseriu hace distinción entre los conceptos categoría y clase, definiéndolos como el “qué” y el “cómo” respectivamente, además, describe que las categorías pueden justificar la constitución de las clases, pero no pueden definirse como clases [4]. En [2]

también se detalla de manera más técnica la diferencia de los conceptos realizando una comparación exhaustiva de los términos tomando en cuenta características como proceso y estructura. Para este trabajo, cuando se trate de clasificación se referirá a la identificación de las expresiones verbales fijas venezolanas de entre las demás expresiones fijas.

Es importante destacar que los conceptos de "unidad fraseológica" y "expresión idiomática" son utilizados sin distinción, y que esta investigación, además, se centra en un tipo de expresión idiomática: las locuciones verbales, pues a pesar de ser fijas, son las que presentan mayor variación en su núcleo: el verbo; este tipo de unidad fraseológica "...es la más sistematizada y la que ha despertado un mayor interés entre los investigadores" [5].

Para clasificar textos con aprendizaje de máquina o para el manejo de datos en general, es necesario conocer algunos de algoritmos usados en programación, estos suelen clasificarse en tres grandes categorías: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Los algoritmos supervisados pueden aplicar lo que se ha aprendido en el pasado con los nuevos datos; los algoritmos no supervisados pueden extraer inferencia de los conjuntos de datos y los algoritmos por refuerzo descubren vía prueba y error las acciones que conllevan a mejores resultados [1]. Para el diseño del módulo se utilizarán aprendizaje supervisado.

El proyecto está dividido en cinco partes: El primer capítulo introduce los conceptos preliminares y trabajos relacionados, la razón por la cual se realiza el proyecto y la metodología a utilizar; el segundo capítulo, el marco teórico, desglosará con detalle los algoritmos a utilizar y los conceptos de lingüística y programación necesarios para desarrollar el módulo. El tercer capítulo contiene la metodología utilizada para la preparación de los datos a utilizar, así como también las técnicas y recursos de los que se hará uso para cumplir los objetivos. El cuarto capítulo corresponde al desarrollo del módulo del identificador, la etapa experimental; las simulaciones y resultados están disponibles en el quinto capítulo, y por último en el capítulo seis se contemplan las conclusiones, trabajos futuros y recomendaciones.

1.1. Antecedentes

1.1.1. Clasificación de textos y documentos

En 1961, Maron hace mención del "indexado automático", un indicador que decide de qué manera se hace que un documento pertenezca a una categoría. Señala que la dificultad para categorizar documentos de forma automática reside en su contenido, porque existe un conjunto de documentos y una serie de categorías en las que dichos documentos podrían encajar, pero estas categorías no son exclusivas ni completamente independientes. La forma de hacerlo, consiste en leer documentos y en base a las palabras claves determinar a qué categoría pertenece [6].

La clasificación automática de textos ha sido estudiada en gran medida desde hace más de dos décadas [7] y hoy en día sigue siendo usada. En [8] definen la categorización o clasificación de textos matemáticamente como la tarea de asignar un valor binario a cada par $< d_j, c_i >$ perteneciente a DxC, donde D es un dominio de documentos y $C = c1, \ldots, c|c|$ es un conjunto predefinido de categorías. Un valor de T asignado a $< d_j, c_i >$ indica la opción de poner a d_j bajo c_i, mientras que un valor de F indica una decisión de no poner a d_j bajo c_i. Formalmente, es la tarea de aproximar una función objetivo $\phi^* : DxC- > T, F$ (la cual describe como los documentos deberían ser categorizados) por medio de la función $\phi : DxC- > T, F$ llamada clasificador, de tal manera que $\phi^* y \phi$ "coincidan lo mejor posible".

Los sistemas de categorización de textos pretenden imitar el juicio que tienen los seres humanos para cualificar la información. Diversos métodos han sido practicados para la categorización de textos usando aprendizaje supervisado entre los que se encuentran modelos de regresión (RM, por sus siglas en inglés), árboles de decisión (DT), redes neuronales (NN) y máquinas de vectores de soporte (SVM), entre otros [9]. Estos métodos siguen siendo populares hoy día [10, 11, 12, 13].

1.1.2. Identificación de expresiones

Distintos trabajos han sido propuestos sobre identificación de expresiones, y diferentes términos han sido utilizados para representar las mismas, siendo en inglés,

el término “idiom” es el mas frecuente para referirse a las expresiones. En [14] se propone la identificación de combinaciones idiomáticas verbo-sustantivo en inglés con el método de etiquetado de palabras, para ello utiliza tanto aprendizaje supervisado como aprendizaje no supervisado. Para la estrategia supervisada propuesta, primero extrae características basándose en vectores de palabras usando la biblioteca word2vec y después usa estas representaciones de los tokens de las combinaciones idiomáticas para entrenar un clasificador de textos; por otra parte, utiliza aprendizaje no supervisado combinando el etiquetado de palabras con agrupamiendo de k-medias; el enfoque supervisado obtuvo mejores resultados al obtener un máximo de 88.3 % de exactitud para el conjunto de prueba versus 87.9 % para el conjunto de prueba del enfoque no supervisado.

En [15] presentan un método para la extracción de “expresiones multipalabras” (MWEs, Multiword Expressions) para el idioma inglés, caracterizando expresiones basándose en sus propiedades dentro del contexto en el que están. Para cada muestra (candidato a expresión), se define un grupo de patrones contextuales (como prefijos y sufijos) y luego, se usa un modelo de aprendizaje supervisado para mejorar estas características con la ayuda de otros aspectos como la frecuencia de la expresión o el tipo. Las expresiones fueron extraídas con el uso de Máquinas de Vectores de Soporte y el sistema alcanzó un alto nivel de precisión en la predicción.

En [16] se presenta una metodología para la identificación de secuencias verbales fijas, basada en la construcción de un corpus etiquetado y una base de conocimientos de secuencias verbales utilizando técnicas de aprendizaje automático que tienen en cuenta el orden de las palabras, denominadas Campos Condicionales Aleatorios.

En [17] se utilizan técnicas de “tokenización”, y “lematización” para el reconocimiento de incisos con el soporte de un módulo llamado Smorph. Estos métodos son utilizados cuando se identifican en una frase u oración sus signos de puntuación, caracteres especiales, entre otros. Todo este proceso permite la separación de textos con significados autónomos para un posterior proceso de análisis [18].

1.1.3. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (NLP, del inglés Natural Language Processing) es un área de desarrollo conceptual con más de 50 años de historia y se encarga de estudiar la manera de simular el lenguaje diario a través de un programa [19]. Vilares en su tesis desarrolla una tecnología en base al NLP y al estudio de la viabilidad de su aplicación en sistemas de recuperación de información sobre documentos en español, también señala la escasez de estudios en español frente a los existentes en inglés [20]. En [21] se realiza un trabajo sobre un analizador de errores gramaticales, describen el NLP como una tecnología que permite programar el computador con suficiente información lingüística en forma de reglas y patrones que hacen posible el realizar numerosas actividades relacionadas con el aprendizaje de lenguas.

1.1.4. Minería de texto en redes sociales

Las redes sociales constituyen uno de los principales medios de comunicación de la última década, la cantidad de datos que se pueden encontrar es abismal y resultan una fuente rica en datos en bruto que pueden ser fácilmente recopilables; por ende, muchos trabajos relacionados a la inteligencia artificial y en particular, al aprendizaje automático, utilizan estos medios para la construcción de los conjunto de datos usados para predecir, analizar, identificar y clasificar. La minería de texto es la técnica detrás de la recolección desde estos medios.

En [22] realizan una comparación entre las distintas herramientas utilizadas para la minería de datos de la red social Twitter centrándose principalmente en el algoritmo de Máquinas de Vectores de Soporte evaluando algunos software de minería de textos como Rapidminer, Leximancer, entre otros, de acuerdo a su calidad y funcionalidad.

En [23] evalúan las técnicas usadas para la minería de datos en redes sociales para segmentación de mercado entre las que se encuentran las bibliotecas de Python, R y software con licencia como Oracle Data Mining. En esta investigación, se escogió Python por su dinamismo y la gran variedad de librerías disponibles en el área del aprendizaje automático que existen como código abierto.

En [24] contextualizan el proceso de limpiado de datos al realizar análisis de

sentimientos en la red social Twitter al momento de clasificar textos. Esta investigación describe los elementos que puede contener un mensaje que pudieran no ser relevantes al momento de analizar: como los símbolos # para los hashtags y para las menciones; la inclusión de la secuencia RT o los hipervínculos.

1.2. Planteamiento del problema

La clasificación de textos ha sido un tópico de gran auge en los últimos años, en especial con la expansión del internet y la creciente disponibilidad de textos científicos, informales, comerciales, entre otros, en formato digital. La demanda por la gestión y análisis de los mismos ha ido aumentando, y debido a que el manejo "a mano" de tales cantidades de información resulta poco factible, se han estado desarrollando herramientas que permitan automatizar la tarea de clasificar [25]. La clasificación de textos es el agrupamiento de palabras que se relacionan de alguna forma, o que comparten alguna característica; su automatización podría permitir conseguir la clasificación de las palabras de forma más eficiente, siendo uno de los métodos para esta automatización más efectivos las técnicas de aprendizaje automático o "Machine Learning", se basa en el entrenamiento de datos previamente seleccionados con el fin obtener resultados favorables al analizar nuevos datos [26]. Sin embargo, una de las más grandes barreras que debe romper una máquina para poder entender un escrito, es la parte semántica del mismo; lo que no está definido literalmente, incluso para los seres humanos, supone un reto el identificar las excepciones al momento aprender un nuevo idioma.

Este trabajo se centra en una de las excepciones del idioma: las expresiones idiomáticas, las cuales constituyen un papel importante en la comprensión de un texto y se definen como unidades léxicas marcadas culturalmente, Alousque las denota como "... una fuente indiscutible de inequivalencias traductológicas [27]" que plantean problemas a la hora de ser transvasadas a otra lengua. Sevilla por su parte, se refieren a estas expresiones de distintas maneras: modismos, idiotismos, refranes, proverbios [28]. Mientras otros autores encapsulan los conceptos en una misma definición [29]. Un ejemplo de estas expresiones sería la expresión venezolana "Dar la cola", la cual tiene

un significado como frase completa, pero cuyo significado cambia al separarse. Durante el desarrollo de la presente investigación se describirá con más detalle sobre ellas, y sobre otras unidades léxicas.

Se ha desarrollado una herramienta que permita la identificación de expresiones verbales fijas en textos en español, partiendo de la información textual extraída de Twitter, una de las redes sociales más populares. Esta red social proporciona textos informales lo suficientemente recientes como para poder analizar expresiones que aún son usadas. La herramienta desarrollada en Python, se pone a prueba con algunos métodos de aprendizaje de máquina supervisado para categorización textual que incluyen algoritmos de Naive Bayes, árboles de decisión, impulso adaptativo y máquinas de vectores de soporte; Estos son algunos métodos que han sido examinados profundamente en los primeros trabajos sobre categorización de textos [9] pero que siguen vigentes hoy día [30].

1.3. Alcance

El siguiente proyecto de investigación describe la creación de una herramienta para la identificación de expresiones verbales fijas mediante algoritmos de aprendizaje de máquina. Para su desarrollo se han utilizado algunas bibliotecas open-source que ofrece Python, entre ellas la biblioteca SpaCy, la cual está enfocada en NLP y posee soporte en español para el tratamiento de textos. Para la implementación de los algoritmos de clasificación y la selección de características se utilizará la biblioteca scikit-learn.

Python es ampliamente utilizado en ciencias e ingeniería y dispone de una gran comunidad que brinda soporte a desarrolladores en el campo de la Minería de Datos. La librería Tweepy, permite el acceso a la Api de Twitter y pone a disposición las diferentes herramientas que la red social ofrece a los programadores. Los textos que se usarán para el entrenamiento del conjunto de datos del módulo constituyen en textos extraídos de mensajes (Tweets) de Twitter a partir de una base de conocimientos de expresiones. Por otra parte, el Diccionario de Venezolanismos [31], se utilizará para la extracción de las expresiones venezolanas para la creación de la base de conocimientos de expresiones

verbales, aunque, debido a la antiguedad de este manuscrito se realizarán encuestas a personas venezolanas con el fin de probar que estas expresiones siguen siendo conocidas, además de búsquedas de cada expresión dentro de la base de datos de Twitter.

1.4. Objetivos

1.4.1. Objetivo general

- Implementar algoritmos de clasificación para la identificación de expresiones verbales fijas venezolanas en textos en español.

1.4.2. Objetivos específicos

- Estudiar algoritmos de aprendizaje de máquina para la clasificación de textos.
- Investigar sobre la mineria de texto en la red social Twitter.
- Definir una base de conocimientos de expresiones verbales fijas venezolanas utilizando técnicas de procesamiento del lenguaje natural y extracción de información.
- Construir un corpus de textos con expresiones verbales fijas a partir de información recopilada de Twitter.
- Construir un corpus de candidatos a expresiones verbales fijas a partir de información recolectada de Twitter.
- Comparar la eficacia de los algoritmos clasificación para la caracterización de expresiones verbales fijas y la identificación expresiones verbales fijas venezolanas.
- Interpretar los resultados obtenidos de las pruebas realizadas a los clasificadores.

1.5. Metodología

El desarrollo de este trabajo constará en cuatro etapas:

1.5.1. Etapa de Investigación

- Revisión de artículos científicos sobre categorización de texto e investigación referente a los métodos de categorización de textos.
- Investigación acerca de información pertinente sobre expresiones idiomáticas.
- Consultas a un tutor experto en lingüística.
- Extracción de expresiones del diccionario de venezolanismos.
- Lectura de la documentación necesaria para desarrollar en Python
- Lectura de la documentación necesaria para utilizar el API de Twitter
- Investigación sobre el funcionamiento de las bibliotecas para procesamiento de lenguaje natural.
- Estudio de los métodos para la extracción de características.

1.5.2. Etapa de Desarrollo

- Recolectar los datos necesarios para la construcción de la base de conocimientos de expresiones verbales fijas.
- Creación del corpus etiquetado a partir de mensajes de la red social Twitter.
- Verificar la vigencia de las expresiones verbales del Diccionario de Venezolanismos, mediante la realización de encuestas a un grupo de la población
- Propuesta de metodología para el etiquetado del conjunto de datos
- Entrenar los algoritmos de clasificación para los distintos métodos de extracción de características.

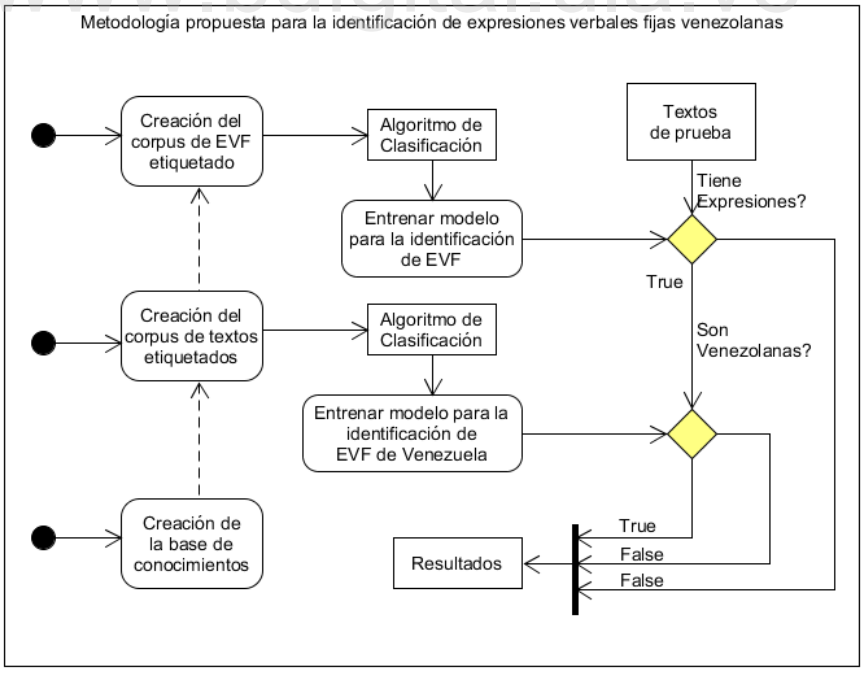
1.5.3. Etapa de Pruebas

- Pruebas de validación cruzada
- Evaluación del rendimiento de los modelos de clasificación
- Simulación de los modelos con textos informales.

1.5.4. Resultados y Conclusiones

- Análisis de resultados.
- Conclusiones
- Recomendaciones y trabajos futuros

Figura 1.1: Metodología propuesta para la identificación de expresiones verbales fijas venezolanas



La Figura 1.1 muestra la metodología propuesta para la identificación de expresiones verbales fijas venezolanas.

1.6. Justificación

En la actualidad se desarrolla un creciente interés por la automatización, en especial por el reciente auge del Internet y el impacto que ha tenido en la comunicación entre las diferentes partes del mundo. Las tecnologías de comunicación han permitido la disponibilidad de grandes cantidades de información, a veces más de la que se puede procesar, presentándose mayormente de forma textual [32]. La clasificación automática de textos permite el correcto manejo de esta información.

Una máquina necesita entender el lenguaje para poder analizarlo, por ello existe el NLP, un campo que junto a la inteligencia artificial se encarga de estudiar el entendimiento de los lenguajes viene de la mano de la comprensión de los idiomas, y cada idioma posee combinaciones gramaticales y semánticas que lo hacen único. Una de estas combinaciones lingüísticas son las unidades fraseológicas, estas son la parte de un idioma que se sale del estándar y que mas difícil se vuelve de definir. Este trabajo se centra en esta fracción semántica del lenguaje.

Entrando mas en el mundo de las unidades fraseológicas es posible encontrar las expresiones verbales fijas, estas constituyen una forma parcialmente fija de unidad fraseológica ya que su núcleo (el verbo), cambia constantemente [33]. En [34] se examina el rol de estas dentro del análisis de sentimientos, estimando que la inclusión de estas expresiones genera mejores resultados en comparación a los obtenidos en análisis de sentimientos estándares; asignando un tipo de emoción a una serie de expresiones idiomáticas pre-seleccionadas, por lo tanto, conocer de primera mano las expresiones idiomáticas con el uso de este módulo sería de gran utilidad en investigaciones futuras sobre análisis textual.

El dominio de las expresiones idiomáticas y modismos es uno de los principales obstáculos que presenta un estudiante de lenguas extranjeras, dadas las enormes diferencias que suelen presentar con respecto a las existentes en su lengua materna [35]; estas expresiones poseen una estructura peculiar que las convierte en singulares, y esto constituye un gran problema considerando que los diccionarios bilingües y monolingües de expresiones existentes no cubren todas las necesidades del usuario. Un estudio realizado en [36] revela que los diccionarios fraseológicos de inglés y español, deberían mejorar la forma en la que estas expresiones son tratadas de acuerdo a criterios de

selección e inclusión y las convenciones tipográficas más pertinentes para transmitirla e incluso se presentan algunas sugerencias para ese mejoramiento.

El desarrollo de este módulo permitirá la elaboración de trabajos futuros relacionados al aprendizaje del idioma español, haciendo uso de recursos venezolanos para esta investigación. Tendrá utilidad en otras aplicaciones relacionadas al análisis de sentimientos, traducción y procesamiento del lenguaje natural [37, 38, 39]

Capítulo 2

Marco Teórico

En éste capítulo se presentarán los conocimientos necesarios para comprender la recopilación de los datos y la implementación del identificador. Se describen con detalle las expresiones idiomáticas que se identificarán y se contextualiza al lector sobre las tecnologías y métodos utilizados.

2.1. Aprendizaje automático

También llamado "Aprendizaje de Máquina" o "Machine Learning", es una de las tecnologías más recientes y forma parte del campo de estudio de la Inteligencia Artificial. Al inicio de este documento, se describió lo que representa el aprendizaje automático para esta investigación, en particular para la clasificación automática de textos, el aprendizaje automático ofrece muchas ventajas ya que su empleo radica en una precisión equiparable con expertos humanos, además de no necesitar intervención continua de los mismos para tareas como el análisis y la clasificación.

Existen tres métodos de aprendizaje automático, estos son: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Para problemas de clasificación, es común el uso de algoritmos de aprendizaje supervisado ya que se necesita un conjunto de datos previamente etiquetado para clasificar [9].

En el aprendizaje supervisado, un conjunto de datos provee al algoritmo una función con entradas y salidas, con esta información el agente (el modelo de la IA)

corrige los parámetros de la función e intenta reducir la magnitud de la perdida global. Después de cada iteración, si el algoritmo es suficientemente flexible y los elementos del conjunto de datos son coherentes, la precisión aumenta y la diferencia entre los valores predichos y esperados se acerca a cero, el objetivo es entrenar un conjunto de datos que funcione genéricamente, pero también sirva para ejemplos nunca antes vistos. Las aplicaciones comunes para aprendizaje supervisado incluyen: clasificación en categorías, detección de spam, detección de patrones, NLP, análisis de sentimientos, procesamiento automático de secuencias, entre otros [26].

A continuación se presentan algunos algoritmo de aprendizaje automático que se utilizarán durante el trabajo.

2.1.1. Algoritmos de Naive Bayes

Los algoritmos de Naive Nayes son clasificadores de entrenamiento rápido que determinan las probabilidades de un resultado usando el teorema de Bayes, cuyo desempeño impera en situaciones donde la probabilidad de una etiqueta esté determinada por la probabilidad de algunos factores causales. En el procesamiento del lenguaje natural, la frecuencia relativa de un término provee información suficiente para inferir a que etiqueta pertenece un texto [26]. Los algoritmos de Naive Bayes han sido utilizados en análisis de sentimientos y en la clasificación de modismos de la lengua inglesa [34] y mandarín [40], asi como también en la identificación de unidades fraseológicas en español mexicano [41]. Esto sugiere que es una herramienta ideal para la clasificación de expresiones.

Teorema de Bayes

Considérense dos eventos probabilísticos A y B. Se pueden correlacionar las probabilidades marginales $P(A)$ y $P(B)$ con las probabilidades condicionales $P(A|B)$ y $P(B|A)$ usando la regla:

$$\begin{cases} P(A \cap B) = P(A|B)P(B) \\ P(B \cap A) = P(B|A)P(A) \end{cases} \tag{2.1}$$

Considerando que la operación intersección es conmutativa, se obtiene la expresión:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

Los elementos de un clasificador Naive Bayes se describen de la siguiente manera; considérese un conjunto de datos, que se quiere clasificar:

$$X = \bar{x_1}, \bar{x_2}, ..., \bar{x_n} \; donde \; \bar{x_i} \in \mathbb{R}_m \tag{2.3}$$

Donde cada vector de características es representado de la siguiente forma:

$$\bar{x_i} = [x_1, x_2, ..., x_m] \tag{2.4}$$

y el conjunto de salida es representado como:

$$Y = \{y_1, y_2, ..., y_n\} \; donde \; y_n \in (0, 1, 2, ...P) \tag{2.5}$$

Aqui, cada y puede pertenecer a una de las P diferentes etiquetas. Considerando el Teorema de Bayes bajo independencia condicional se puede escribir:

$$P(y|x_1, x_2, ..., x_m) = \alpha P(y) \prod_i P(x_i|y) \tag{2.6}$$

El valor de la mínima probabilidad $P(y)$ y de las probabilidades condicionales $P(x_i|y)$ es obtenido a travez de un contador de frecuencia; por lo tanto, dado un vector de entrada x, la etiqueta predecida es una de la cual su probabilidad es máxima.

Los clasificadores bayesianos están basados en las distintas distribuciones de probabilidad. Se pueden mencionar los siguientes modelos:

2.1.1.1. Modelo de Bernuoulli

Sea X una variable aleatoria y distribuida con Bernoulli, solo puede adquirir dos valores (simplificando, 0 y 1) y sus probabilidades son:

$$P(X) = \begin{cases} p & si \; X = 1 \\ q & si \; X = 0 \end{cases} \quad donde \; q = p - 1 \; y \; 0 < P < 1 \tag{2.7}$$

2.1.1.2. Modelo Multinomial

La distribucion multinomial es utilizada para modelar vectores de características donde cada valor representa un numero de eventos de un término. Si cada vector de características tiene n elementos y cada uno de ellos adquiere k valores diferentes con probabilidad p_k entonces:

$$P(X_1 = x_1 \cap X_2 = x_2 \cap ... \cap X_k = x_k) = \frac{n!}{\prod_i x_i!} \prod_i P_i^{x_i} \tag{2.8}$$

2.1.1.3. Modelo Gaussiano

Este algoritmo es altamente utilizado cuando se usan valores continuos cuyas probabilidades pueden ser modeladas usando una distribución Gaussiana:

$$P(X) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.9}$$

Las probabilidades condicionales ($P(x_i|y)$ son distribuciones Gaussianas; por lo tanto, es necesario estimar la media y la varianza de cada una de ellas usando la máxima verosimilitud. De esta forma, se obtiene:

$$L(\mu; \sigma^2; x_i|y) = \log \prod_k P(x_i^(k)|y) = \sum_k \log P(x_i^(k)|y) \tag{2.10}$$

Donde el indice k representa los ejemplos del conjunto de datos y $P(x_i|y)$ es un Gaussiano. Minimizando la inversa de esta expresión, se consigue la media y la varianza de cada Gaussiano asociado a $P(x_i|y)$ y por consiguiente el modelo es entrenado.

2.1.2. Máquinas de Vectores de Soporte

El algoritmo de Máquinas de Vectores de Soporte (SVM, del inglés Support Vector Machine) constituye la mejor opción al momento de clasificar conjuntos de datos [26]. Estos algoritmos funcionan bajo modelos lineales y no lineales, esto quiere decir que permiten un buen desempeño en distintos contextos. En el procesamiento del lenguaje natural, las SVM pueden manejar la clasificación de datos de entrada en

forma de palabras para meterlos dentro de una categoría; recientemente son usados en el campo de la clasificación de tokens, detección de spam y análisis de textos [42, 43].

El algoritmo SVM es un modelo que representa a los puntos de un conjunto de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación llamado vector soporte, esto en el caso bidimensional, pero las SVM pueden trabajar en n-dimensiones si se considera separar las clases mediante un conjunto de hiperplanos. Es decir, las SVM pueden ser tanto un hiperplano como un conjunto de hiperplanos en un espacio dimensional que puede ser utilizado en problemas de clasificación o regresión. De manera mas detallada, [26] describe los elementos de las SVM lineales de la siguiente manera:

Considérese un conjunto de vectores de características, que se quiere clasificar:

$$X = \bar{x_1}, \bar{x_2}, ..., \bar{x_n} \; donde \; \bar{x_i} \in \mathbb{R}_m \tag{2.11}$$

Se asume que se quiere realizar una clasificación binaria, cuyas etiquetas se denotan como -1 y 1:

$$Y = \{y_1, y_2, ..., y_n\} \; donde \; y_n \in \{-1, 1\} \tag{2.12}$$

Se intenta encontrar el hiperplano $\mathbf{H_0}$ que mejor separe el conjunto de datos dado, cuya ecuación es la siguiente:

$$\bar{w}^T \bar{x} + b = 0 \; donde \; \bar{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} \; y \; \bar{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \tag{2.13}$$

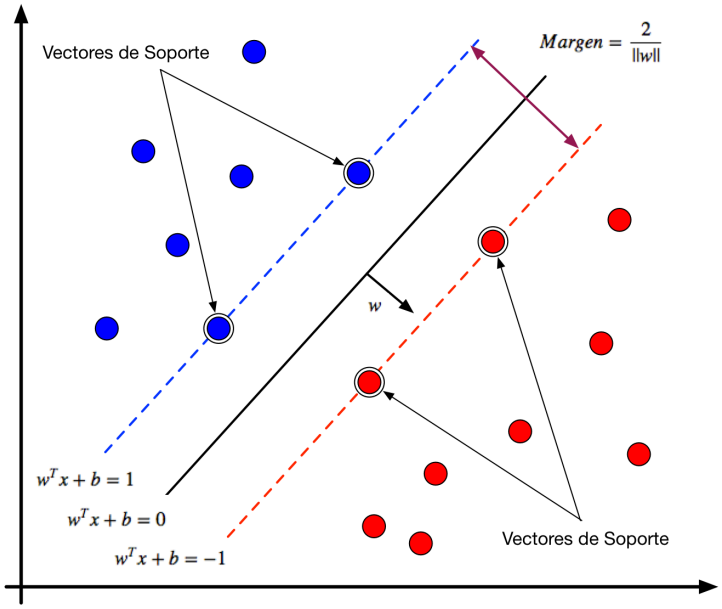
En esta ecuación (2.13), $\bar{w}w$ y $\bar{x}x$ son vectores y $\bar{w}^T\bar{x}$ representa el producto escalar de los dos vectores. El vector $\bar{w}$ se denomina a menudo vector de pesos.

De esta manera, el clasificador se puede escribir como:

$$\bar{y} = f(x) = sgn(\bar{w}^T \bar{x} + b) \tag{2.14}$$

En una aplicación común, las dos etiquetas son normalmente separadas por un margen con dos límites que dependen de algunos elementos. Estos elementos son

Figura 2.1: Máquina de Vectores de Soporte y sus elementos en scikit-learn



llamados vectores de soporte. La Figura 2.1 Muestra el márgen de separación entre las dos clases (espacio entre las lineas en rojo y azul), que sirve de límite para las clases dentro de un espacio de dos dimensiones y el hiperplano de separación.

Para una vista genérica de la expresión, se suele normalizar el conjunto de datos de tal manera que los vectores de soportes esten unidos por dos hiperplanos $\mathbf{H_1}$ y $\mathbf{H_2}$ equidistantes a $\mathbf{H_0}$ cuyas ecuaciones serían:

$$\begin{cases} \bar{w}^T \bar{x} + b = -1, \\ \bar{w}^T \bar{x} + b = 1 \end{cases} \tag{2.15}$$

El objetivo principal es maximizar la distancia entre los limites de los dos hiperplanos y para así minimizar la probabilidad de conseguir una clasificación incorrecta.

Considerando que los hiperplanos son paralelos, la distancia entre ellos está definida por la longitud de un segmento perpendicular a ambos que conecta dos puntos x_1 y x_2. Si se consideran estos puntos como vectores se obtiene el vector ortogonal:

$$\bar{x_1} - \bar{x_2} = t\bar{w}^T \tag{2.16}$$

Donde la longitud del segmento t es la distancia entre los puntos x_1 y x_2. Sustituyendo 2.16 en las ecuaciones 2.15 se obtiene:

$$\bar{w}^T\bar{x_2} + b = \bar{w}^T(\bar{x_1} + t\bar{w}^T) + b = (\bar{w}^T\bar{x_1} + b) + t\left\|\bar{w}^2\right\| = 1 \tag{2.17}$$

Como $\bar{w}^T\bar{x_1} + b = -1$ (2.15) se obtiene el valor de t y a su vez, la distancia entre los dos puntos:

$$t = \frac{2}{\|\bar{w}^2\|} \tag{2.18}$$

$$d(x_1, x_2) = t\,\|\bar{w}\| = \frac{2}{\|\bar{w}\|} \tag{2.19}$$

Como se había mencionado antes, las SVM también pueden funcionar bajo modelos no lineales, para entender un poco sobre eso, es necesario conocer un concepto detrás de estas máquinas denominado kernel o núcleo.

2.1.2.1. Kernel

Los kernels son funciones particulares que poseen la propiedad de calcular el producto de dos vectores proyectados. Esto permite, a nivel de complejidad computacional, beneficiarse de otras proyecciones no-lineales, incluso de dimensiones muy grandes. En algunos casos, los datos pueden no ser linealmente separables en dos dimensiones, los kernels solucionan este problema agregando otras dimensiones.

$$K(\bar{x}_i, \bar{x}_j) = \phi(\bar{x}_i^T)\phi(\bar{x}_j) \tag{2.20}$$

Las funciones kernel retornan el producto interno entre dos puntos en un espacio para ser interpretados como medida similitud entre los datos de entrada. Existen varios tipos de kernel y diferentes usos, se mencionaran algunos a continuación:

- Kernel polinomial

$$K(\bar{x}_i, \bar{x}_j) = (\gamma\bar{x}_i^T\bar{x}_j + r)^c \tag{2.21}$$

- Kernel sigmoide

$$K(\bar{x}_i, \bar{x}_j) = (\gamma \bar{x}_i^T \bar{x}_j + r)^c \tag{2.22}$$

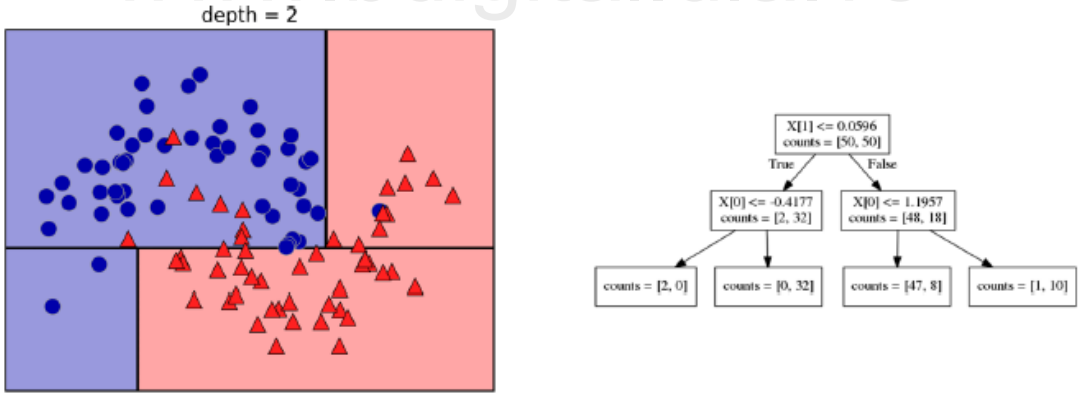
- Funcion de base radial o RBF kernel

$$K(\bar{x}_i, \bar{x}_j) = e^{\gamma |\bar{x}_i - \bar{x}_j|^2} \tag{2.23}$$

2.1.3. Árboles de Decisión Binarios

La estructura de los árboles de decisión (DT) se basa en un proceso secuencial de decisiones, empezando desde la raíz del árbol, una característica es evaluada y se selecciona una de dos posibles ramas. Este procedimiento se repite hasta que se alcanza la hoja final, la cual generalmente representa la clasificación que se estaba buscando. En la Figura 2.2 se puede encontrar la clasificación binaria de un árbol de profundidad 2.

Figura 2.2: Ejemplo de Árbol de decisión binario de profundidad 2



Considérese un conjunto de datos, que se quiere clasificar:

$$X = \bar{x_1}, \bar{x_2}, ..., \bar{x_n} \; donde \; \bar{x_i} \in \mathbb{R}_m \tag{2.24}$$

Cada vector está formado por m características, así que cada uno es un candidato para crear un nodo basado en la tupla (característica, umbral). Formalmente se define de la siguiente manera:

$$\sigma =< i, t_k > \tag{2.25}$$

En esta expresión, el primer elemento es el indice de la característica que se usa para separar el conjunto de datos, y el segundo es el umbral que determina las ramas izquierda y derecha del árbol. La decisión del mejor umbral es un elemento fundamental ya que determina la estructura del árbol, y por ende, su desempeño. El objetivo de este algoritmo es el de reducir la impureza en el el numero de divisiones, para tener un camino corto entre los datos de ejemplo y los resultados de la clasificación.

El total de impureza se define como sigue:

$$I(D, \sigma) = \frac{N_{izquierda}}{N_D} I(D_{izquierda}) + \frac{N_{derecha}}{N_D} I(D_{derecha}) \tag{2.26}$$

Donde D, es el conjunto de datos completo. $D_{izquierda}$ y $D_{derecha}$ son los subconjuntos de datos resultante y I es la medida de impureza.

La parte más difícil de construir el árbol es seleccionar el atributo que produzca la mejor división. Hay tres medidas comunes de impureza usadas para medir la mejor división.

- Impureza de Gini

$$I_{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2 \tag{2.27}$$

- Entropía

$$I_{Entropia}(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t) \tag{2.28}$$

- Error de clasificación

$$I_{ErrorClasificacion}(t) = 1 - \max_i [p(i|t)] \tag{2.29}$$

2.1.4. Impulso Adaptativo (AdaBoost)

El impulso adaptativo (AB) es una técnica de aprendizaje combinado que se diferencia del árbol de decisión porque el conjunto de datos usado para el entrenamiento es continuamente adaptado para forzar al modelo en enfocarse en los ejemplos que son

erróneamente clasificados. Nuevos clasificadores son añadidos para mejorar al anterior, mejorando el desempeño en esas áreas donde no existió la precisión que se esperaba. En cada iteración se aplica un peso a cada ejemplo para incrementar la importancia de los resultados predichos erróneamente y se disminuye la importancia de los otros. La característica de importancia viene dada por:

$$Importancia(x_i) = \frac{1}{N_{Trees}} \sum_t \sum_k \frac{N_k}{N} \Delta I_{x_i} \tag{2.30}$$

2.2. Minería de textos en Twitter

La era digital ha hecho posible que la información digitalizada sea fácil de procesar, distribuir, y transmitir. La minería de textos es el proceso de extraer información significativa de texto no estructurado. Es un campo relativamente reciente con un gran valor comercial que se nutre de las áreas de recuperación de la información, minería de datos, aprendizaje automático, estadística y procesamiento del lenguaje natural [44].

La recopilación de información de redes sociales proporciona una fuente de datos con gran proyección a la hora de su análisis. Las redes sociales se han convertido en un recurso significativo de datos sin estructurar; y cuando se buscan datos en forma de texto, Twitter es una de las redes mas populares en la actualidad, con más de 300 millones de usuarios activos [45], cuyo atractivo principal son los textos de poco caracteres; a diferencia de otras redes como Instagram pues su atractivo, en cambio, son las imágenes. Esta enorme cantidad de datos provee información útil si es procesada de manera adecuada, y, debido a la naturaleza de Twitter, donde personas comparten sus opiniones, sentimientos y expresiones de su vida diaria, han sido usados en la construcción de modelos a partir de su información lingüística [46, 47].

2.2.1. Twitter

Twitter es un servicio de microblogging que permite enviar mensajes de texto con un máximo de 140 caracteres (doblado a 280 caracteres en 2017), estos mensajes se llaman Tweets, y aparecen en la página principal del usuario. Cada usuario puede seguir

a otros usuarios y ver sus tweets y se puede acceder desde la web (www.twitter.com) o desde aplicaciones para smartphones.

2.2.1.1. Estructura de un Tweet

Un Tweet posee información relacionada al Tweet, su origen, el usuario que lo publicó, entre otros campos clave:

- `text`, es el cuerpo del Tweet, el mensaje.
- `ID`, es la identificación del Tweet, es un número único.
- `created_date`, es la fecha de creación del Tweet.
- `lang`, es el idioma del Tweet, en formato ISO 639-1.
- `place, geolocation`, la información de lugar del Tweet, si se encuentra disponible.
- `user`, este campo involucra el perfil completo del usuario, donde se obtiene por ejemplo: su nombre, localización, entre otros.

2.2.2. Recopilación de datos

Para empezar a recopilar datos de Twitter primero es necesario acceder al API, para esto es necesario crear y registrar un app en la página web de desarrollo:

2.2.2.1. Registro de App: Protocolo de seguridad OAuth

OAuth [48] (Open Authorization) es un protocolo que permite autorización segura de una API de un modo estándar y simple para aplicaciones de escritorio, móviles y web. Este proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta. La creación del app se completa al registrarse en la página web https://apps.twitter.com/, en esta se reciben los tokens de autenticación: `consumer_key`, `consumer_secret_key`, `access_token` y `access_token_secret`, usados para el acceso privado a la cuenta.

2.2.2.2. API de Twitter

Existen dos formas de extraer información desde Twitter: API Rest y Streaming API

- **API Rest** Permite realizar todas las acciones a las que se tiene acceso desde la página web o las aplicaciones. Proporciona acceso a la información ya existente en Twitter en el momento de hacer la llamada.
- **Streaming API** Se inicia con Twitter abriendo una conexión entre su servidor y el sistema. A través del API se envian tweets que son publicados a partir de ese momento, siempre que cumplan los filtros indicados al inicio de la conexión hasta su cierre, es decir, es un proceso en tiempo real.

Para este trabajo solo se utiliza el API Rest. Específicamente, en las consultas utilizando el método `search` de la API de Twitter a través de la librería Tweepy. Ver Figura 2.3.

Figura 2.3: Método de búsqueda del API de Twitter

```
API.search(q, geocode, lang, locale, result_type, count, until, since_id,
    max_id, include_entities)
```

El método retorna una lista de Tweets relevantes que coinciden con la consulta especificada.

Parámetros más importantes

- `q`, es la consulta, una cadena de máximo 500 caracteres, incluyendo operadores de búsqueda avanzada
- `geocode`, retorna Tweets localizados en un radio, dadas unas coordenada. Especificado como "latitud, longitud, radio".
- `lang`, restringe los Tweets a un lenguaje, dado su codigo ISO 639-1. (Por ejemplo: es, para español)

- `result_type`, Especifica el tipo de resultados de la búsqueda: reciente, popular o mezclado.
- `count`, número de resultados que se intentan recuperar (el máximo es 100)
- `until`, retorna Tweets creados antes de una fecha dada
- `since_id`, retorna solo Tweets con ID mayor a la dada (si la ID es mayor, es mas reciente)

2.2.2.3. Geolocalización

El procesamiento de lenguaje natural y la geolocalización de los tweets son los puntos claves dentro de la recopilación de datos de este proyecto. La clasificación de los tweets según su zona geográfica ha estado fuertemente condicionada por la escasez de tweets que incluyan su geolocalización. La solución a este problema segun [49] consiste en la obtención del lugar del usuario que aparece en el perfil, con la pérdida de exactitud que esto supone. No obstante, dado que muchos usuarios no indican un lugar válido en su perfil, el número de tweets geolocalizados en territorio nacional sigue siendo pequeño.

Existen cuatro maneras de localizar un Tweet que pueden o no estar disponibles al extraer la información del mensaje:

1. **Localización exacta:** Viene dada por las coordenadas exactas del Tweet.
2. **País:** Es un metadato que poseen algunos Tweets, es menos preciso que las coordenadas pero mas confiable que la provista por el usuario. [1]
3. **Localización del usuario:** Es un dato impreciso de un Tweet, puesto que el usuario es libre de escribir cualquier cosa.
4. **Localización dentro del Tweet:** Es el caso mas extraño; cuando el nombre de un país o ciudad es mencionado dentro del mensaje.

[1] Sólo un 2 % de los tweets publicados globalmente poseen información sobre coordenadas exactas o país [49].

2.3. Procesamiento del Lenguaje Natural

El entendimiento de como funciona el lenguaje natural es una tarea que lleva años de estudio, de hecho, el Procesamiento del Lenguaje Natural (NLP, del Inglés, Natural Language Processing) empezó en los años 50 como una intersección entre la lingüística y la IA. Originalmente el NLP se distinguía de la IR porque esta última manejaba datos enormes usando técnicas basadas en estadísticas; no obstante, con el tiempo estos dos campos empezaron a converger obligando a los investigadores a ampliar sus conocimientos.

En el caso particular de la categorización automática de texto, el NLP constituye una herramienta de suma importancia, puesto que para el tratamiento de textos es necesario conocer ciertas técnicas, entre las cuales se encuentran la normalización, tokenización y etiquetado de palabras [50].

2.3.1. Normalización

La normalización consiste en el transformado del texto para que tenga consistencia; algunos ejemplos de esto incluye: eliminación de mayúsculas y signos de puntuación, transformación de la codificación del lenguaje o eliminación de stopwords (las stopwords son palabras que son filtradas de acuerdo al uso que puedan tener) [51]. La tokenización también forma parte del proceso de normalización, de la misma manera la conversión de números a letras; además, el proceso de normalización difiere para cada idioma, ya que los alfabetos pueden tener más o menos caracteres o símbolos [52].

2.3.1.1. Tokenización

Se entiende por tokenización a una técnica del NLP que se realiza sobre el texto, ésta consiste en dividir un cuerpo de texto en bloques llamados token. La separación se realiza con la finalidad de facilitar el análisis individual de cada elemento, así como su relación con los demás tokens resultantes. La tokenización se realiza cuando se identifican en una frase u oración sus signos de puntuación, caracteres especiales, entre otros. Todo este proceso permite la separación de textos con significados autónomos

para un posterior proceso de análisis [18].

Se puede ver un ejemplo (véase figura 2.4) de descomposición en bloques haciendo tokenización a una frase de Chomsky, uno de los padres de la lingüística, “La gente paga por su propia subordinación.”

Figura 2.4: Descomposición en bloques de una tokenización

```
[
“La”,
“gente”,
“paga”,
“por”,
“su”,
“propia”,
“subordinación”,
“.”
]
```


2.3.1.2. Lematización y Stemming

La lematización es un proceso importante que se realiza antes del proceso de minería de texto, también es usado en NLP y otros campos de la lingüística. El proceso de lematización consiste en asignar a una palabra flexionada su forma normalizada, cuál se considera la raíz morfológica de la palabra. Por ejemplo, la forma infinitiva del verbo “Estudiar”, constituye una forma normalizada de las palabras flexionadas “estudiar, estudiante, estudioso, estudió, etc” (véase figura 2.5) [51, 53]; esta forma de procesamiento de las palabras provee una manera productiva de generar palabras claves genéricas o etiquetas. La lematización se diferencia del Stemming porque no requiere de una “stem word” [53].

El proceso de Stemming consiste en reducir las palabras a su forma base o raíz, cuya forma base no es necesariamente idéntica a la raíz morfológica de la palabra. Este algoritmo se lleva a cabo eliminando los afijos de las palabras (elementos adicionales adjuntos para cambiar su sentido gramatical). Se puede observar como la

Figura 2.5: Proceso de lematización

$$\left.\begin{array}{l} Estudias \\ Estudio \\ Estudiabas \end{array}\right\} Estudiar$$

forma normalizada "Estudiar" ya no es raíz al existir una stem word, por el contrario, es una forma flexionada, en la Figura 2.6 también se muestra como el stemming podría ocacionar perdida del contexto ya que algunas de las palabras podrian no tener la misma raíz semántica, como "estudias" y "estudios"

Figura 2.6: Proceso de stemming

$$\left.\begin{array}{l} Estudias \\ Estudio \\ Estudiar \\ Estudiante \\ Estudios \end{array}\right\} Estud$$

2.3.2. Vectores de palabras

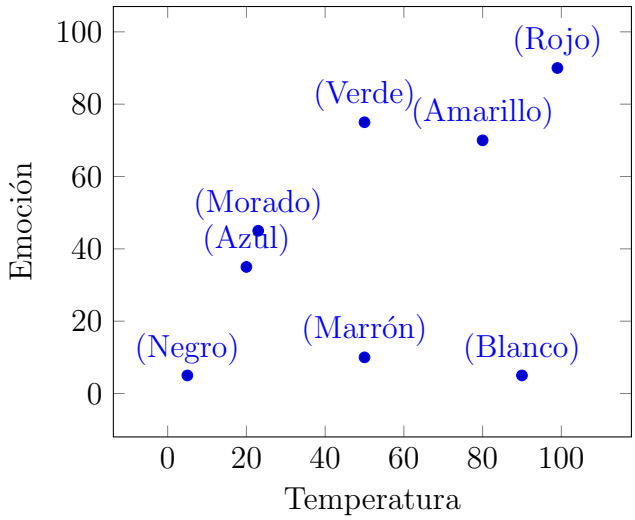
Entender el significado de una palabra es la tarea principal en el NLP; por esa razón, recientemente se han creado modelos en los que cada palabra es representada como un vector n-dimensional de numeros reales, donde cada dimensión es una característica de la palabra. Este acercamiento ha demostrado que la distancia matemática entre estos vectores, puede determinar la relación semántica entre los mismos, estos vectores reciben el nombre de vectores de palabras [54].

Para efectos gráficos. Considérese un conjunto de vectores de palabras bidimensionales:

$$\bar{V}(w : a, b) = \{\bar{v_1}, \bar{v_2}, ..., \bar{v_n}\} \tag{2.31}$$

donde w es la palabra y a, b es la dupla de características que representa esa palabra.

Figura 2.7: Vectores de palabras de 2 dimensiones



Como ejemplo se tomarán vectores de colores de dos características totalmente subjetivas: temperatura y emoción. Los valores de temperatura varían de 0 a 100, siendo 100 el mas caliente. Y los de emoción entre 0 a 100, siendo 0 el más triste, y 100 el mas alegre.

Se definen los valores de los vectores de cada color. Por ejemplo Rojo: (99,90), como se observan en la Figura 2.7. Estos valores subjetivos permiten hacer cálculos con las palabras, como si de números se tratase, gracias a las propiedades de los vectores.

2.3.2.1. Similitud

La medida de similitud de dos vectores viene dada por su Similitud Coseno, es una medida existente entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos, esto permite conocer la relación semántica entre dos palabras. Esta distancia se emplea frecuentemente en las técnicas IR representando las palabras (o documentos) en un espacio vectorial [55].

Tomando en cuenta el ejemplo anterior, se puede calcular que tan relacionados están dos colores.

Sea S la similitud entre dos palabras y $\bar{v}_1(x_1, y_1)$ y $\bar{v}_2(x_2, y_2)$, los vectores palabra a comparar. La similitud coseno se deriva de la fórmula del producto punto entre dos

vectores:

$$\mathbf{v_1} \cdot \mathbf{v_2} = \|\mathbf{v_1}\| \, \|\mathbf{v_2}\| \cos\theta \tag{2.32}$$

Despejando el coseno, se obtiene la ecuación de similitud:

$$S_{v_1,v_2} = \cos\theta = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\| \, \|\mathbf{v_2}\|} \tag{2.33}$$

Entonces S será igual a:

$$S = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2}} \tag{2.34}$$

Para obtener valores entre 0 y 1, aplicamos el Coseno a la similitud y con valor absoluto se descartan los valores entre -1 y 0:

$$S_{cos} = \left| \cos\left(\frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2}} \right) \right| \tag{2.35}$$

La similitud entre Morado y Azul, se calcula como sigue:

$$S_{morado,azul} = \frac{(20)(23) + (35)(45)}{\sqrt{20^2 + 35^2} + \sqrt{23^2 + 45^2}} = 22.3999 \tag{2.36}$$

$$S_{cos} = |cos(22.3999)| = |-0.91762| = 0.91762 \approx 91,76\,\% \tag{2.37}$$

Rojo y blanco, deberían resultar menos similares debido a que están mas lejos tanto en distancia como en dirección:

$$S_{rojo,blanco} = 41.7981 \tag{2.38}$$

$$S_{cos} = 0.57562 \approx 57,56\,\% \tag{2.39}$$

El mismo color debe ser similar a si mismo, supóngase que existe el color azul+ (21,45) que es un poco mas frío que el azul normal (23,45).

$$\bar{S}_{azul,azul+} = 25.0309 \tag{2.40}$$

$$\bar{S}_{cos} = 0.99482 \approx 99,48\,\% \tag{2.41}$$

2.4. Unidades Fraseológicas

Las Unidades Fraseológicas (UF), también llamadas "Expresiones Idiomáticas" [56], son construcciones lingüísticas cuya significación no es composicional, es decir, son un conjunto de palabras que tienen un significado unitario, y que no constituyen la suma total de sus significados individuales [57]. Corpas define la UF como:

> *"...combinaciones estables formadas por al menos dos palabras y cuyo límite superior se sitúa en la oración compuesta. Se caracterizan por la alta frecuencia de aparición en la lengua y de coaparición de sus elementos integrantes, así como por la institucionalización, la estabilidad, la idiomaticidad y la variación potencial que dichas unidades presentan en diverso grado."*
>
> [58, pp.23]

2.4.1. Tipología de las unidades fraseológicas

Ls unidades fraseológicas se dividen en tres grandes grupos: colocaciones, enunciados fraseológicos y locuciones.

2.4.1.1. Colocaciones

Son unidades fraseológicas apegadas a las normas de la lengua pero que tienen distinto grado de fijación según las normas de uso, ya que las palabras que la forman permiten la sustitución paradigmática en mayor o menor grado, esto las convierte en frases poco restrictivas.

La Tabla 2.1 presenta la poca restricción de las colocaciones, tanto verbos, como adjetivos y sustantivos pueden cambiar sin que la expresión pierda su significado.

2.4.1.2. Enunciados Fraseológicos

Son unidades fraseológicas que construyen oraciones completas y pueden dividirse en dos tipos: paremias y fórmulas rutinarias. La figura 2.8 muestra en detalle la clasificación formulada por Corpas en [58] en su manual de fraseología.

Tabla 2.1: Ejemplos de colocaciones

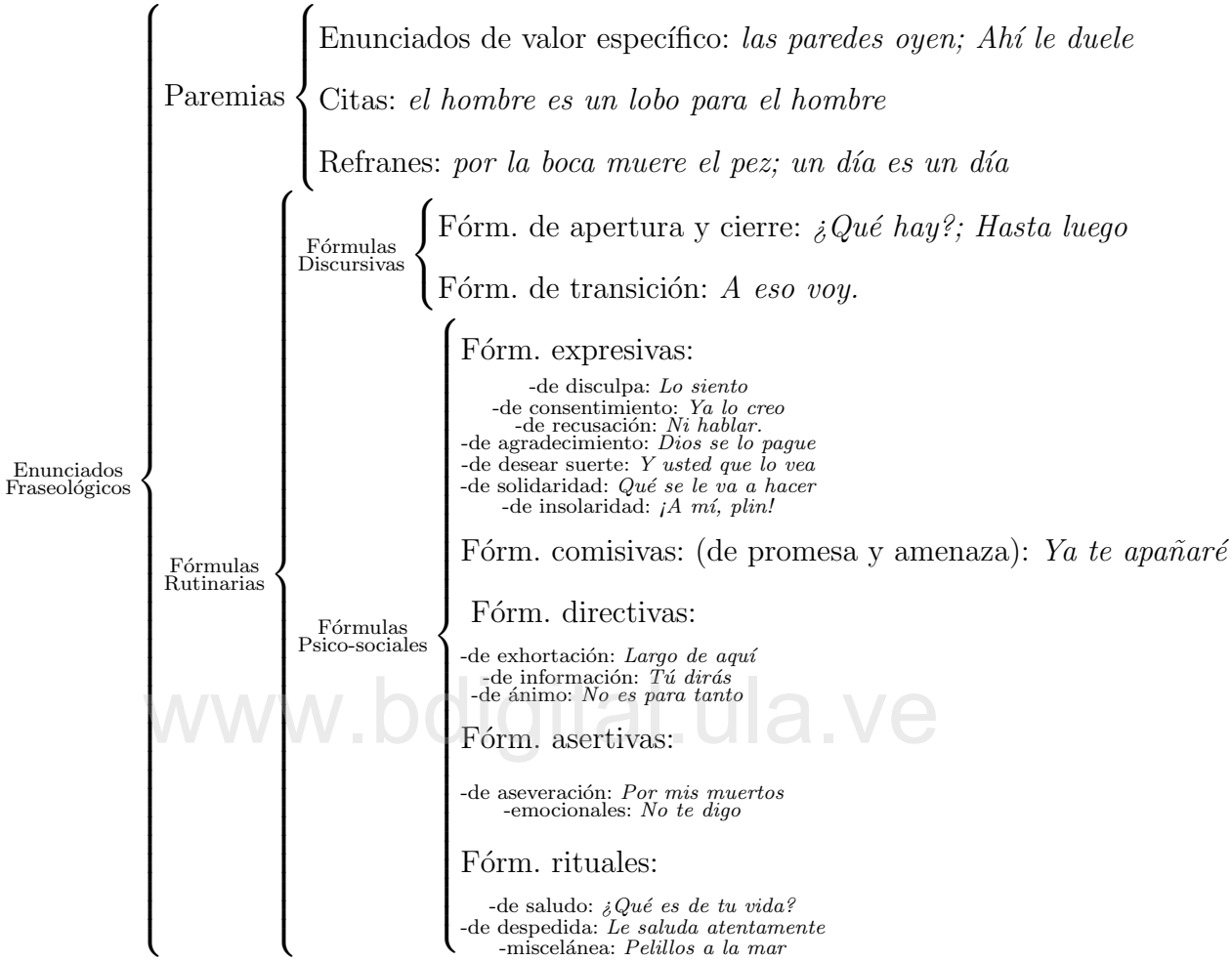
Estructura	**Colocacion**	Variante
VERB + SUST (sujeto)	correr un rumor	correr un chisme
VERB + PREP + SUST(objeto)	asestar un golpe	asestar un puñetazo
ADJ/SUST + SUST	visita relámpago	visita rápida
SUST + PREP	banco de peces	banco de sardinas
VERB + ADJ	negar rotundamente	negar completamente

2.4.1.3. Locuciones

Son expresiones idiomáticas fijas que presentan grandes restricciones de combinación. Una locución no se obtiene composicionalmente, es decir, combinando las palabras que la constituyen, sino que el conjunto léxico es lo que le dá significado; estas se caracterizan por su sentido metafórico, que en algunos casos dificulta su traducción a otra lengua [59]. Según [57] se clasifican de acuerdo a la función que ejercen en la oración en:

- **Locuciones nominales**: *el que diran, mosca muerta, patas de gallo, santo y seña,* entre otras.
- **Locuciones adjetivas**: *corto de medios, de armas tomar, más suave que el algodón, sano y salvo,* entre otras.
- **Locuciones adverbiales**: *boca con boca, con el corazón en la mano, de par en par, más de la cuenta, por lo pronto,* entre otras.
- **Locuciones verbales**: *agarrar con la manos en la masa, (no) quebrar un plato, meter la pata, ir y venir, pisar el peine,* entre otras.
- **Locuciones prepositivas**: *a pesar de, delante de, en lugar de,* entre otras.
- **Locuciones conjuntivas**: *asi que, como si, dado que, mientas tanto, puesto que, tan pronto como,* entre otras.

Figura 2.8: Clasificación de los enunciados fraseológicos



- **Locuciones clausales**: *como quien dice, como Dios manda, hacérsele a alguien agua la boca, subírsele a alguien, revolvérsele a alguien las tripas,* entre otras.

2.4.2. Expresiones Verbales Fijas

Las expresiones verbales fijas (EVF) constituyen toda expresión cuyo núcleo de sintagma es el verbo y además, posee un significado unitario apegado a su distribución geográfica. Para esta investigación se utilizarán las locuciones verbales, clausales con núcleo verbal y los enunciados fraseológicos que contengan verbo como su palabra inicial, contenidos en el diccionario de venezolanismos.

Las locuciones verbales tienen como elemento principal el sintagma verbal[2], ya que están compuestas por un nucleo verbal, acompañado por sus complementos. Estas presentan las mismas características del resto de las locuciones: Fijación interna y el significado unitario [33]. La forma fija significa que los elementos que forman parte de la locución, exceptuando el núcleo, no pueden modificarse, esto significa que no pueden introducirse o reemplazarse palabras. Sin embargo, la fijación de las locuciones no es absoluta, mas bien es relativa y es posible encontrar una locución que tiene dos o más formas y estas formas se consideran individualmente fijas. La unidad de significado es lo que convierte a las locuciones verbales en unidades fraseológicas.

En [33] destacan que la característica principal de las locuciones verbales, y de cualquier locución, es su distribución geográfica, y distingue dos formas mas de clasificar las locuciones: locuciones de ámbito general y locuciones locales. Estas últimas serán el objetivo de estudio de esta investigación, pues solo se tomarán en cuenta expresiones venezolanas. Para esta investigación se referirá a estas locuciones como Expresiones Verbales Fijas venezolanas (EVFV).

2.4.2.1. Verbo

El diccionario de la Real Academia Española lo define como "Clase de palabras cuyos elementos pueden tener variación de persona, número, tiempo, modo y aspecto" [3]. Di Tullio en su Manual de Gramática del Español describe estas cinco variaciones[3] y menciona que las dos primeras representan las características del sujeto mientras que las otras dos las propiedades de la cláusula[60]. Para la preparación de los datos de esta investigación se ha utilizado la combinación de las siguientes conjugaciones:

- **Persona:** Primera, segunda y tercera persona.
- **Número:** Singular, plural.
- **Tiempo:** Presente, pasado, futuro, condicional.

[2]Definiciones de DRAE [3]. **Sintagma**: palabra o conjunto de palabras que se articula en torno a un núcleo y que puede ejercer alguna función sintáctica. **Sintagma verbal**: que tiene por núcleo un verbo.

[3]En el Capitulo XIII Flexión Verbal de [60] Las variaciones son denominadas propiedades flexionadas.

- **Modo:** Subjuntivo, indicativo, imperativo.
- **Aspecto:** Perfecto, imperfecto, progresivo.

Al convertir el verbo en token, es necesario tomar en cuenta otra variación del verbo además de las cinco mencionadas: los complementos.

Complementos del verbo Son operaciones que permiten poner de manifiesto las relaciones del verbo con el sujeto o el objeto [60]. Estos complementos se encuentran antes del verbo, o como sufijo. Como un token es solo una palabra, para este trabajo solo los sufijos serán relevantes.

- Reflexivo: Indican que la acción la realiza el sujeto/objeto a si mismo. Se encuentran como sufijos "-se, -nos, -me, -te", denominados complemento indirecto, después del verbo en infinitivo, progresivo e imperativo. *Ponerse (se pone), poniéndose (se está poniendo), poniéndonos (nos estamos poniendo), ponte (pon a ti mismo)...* Los sufijos "-la, -las, -lo, -los", denominados complemento directo, no se incluyen en el desarrollo porque cambian la fijación (y el significado) de la expresión: *"echarse **tierra**"* → *"echar**sela**", "echarnos **tierra**"* → *"echar**nosla**".*
- No Reflexivo: Indican que la acción se realiza al objeto. Se encuentran como sufijos "-le, -les" (complemento indirecto), después del verbo en infinitivo, progresivo e imperativo. *Ponerle (poner a el/ella), poniéndoles (poniendo a ellos/ellas), ponle (pon a el/ella).* Los sufijos "-la, -las, -lo, -los" (complemento directo), no se incluyen en el desarrollo porque cambian la fijación (y el significado) de la expresión: *"echar **tierra**"* → *"echar**la**", "echando **tierra**"* → *"echando**la**".*

Capítulo 3

Preparación de los datos

En este capítulo se detalla la metodología utilizada para la extracción y verificación de los datos que conforman la base de conocimientos de expresiones verbales fijas y el corpus etiquetado, asimismo, se realiza un estudio de la vigencia de los datos y las dependencias utilizadas para su extracción.

3.1. Dependencias

La programación orientada a objetos introdujo un cambio drástico en el proceso de desarrollo de software. Partiendo de técnicas caracterizadas por su énfasis en la descripción algorítmica de la solución del problema, se cambió a la representación y manipulación de los objetos que caracterizan el problema. Este paradigma abrió las puertas a nuevas formas de desarrollo de software basado en la noción de reutilización de componentes [61]. Para la extracción de datos del Diccionario de Venezolanismos y para la creación del corpus es necesario el uso de herramientas de NLP. A continuación se describen las herramientas utilizadas y su rol en la preparación de los datos.

3.1.1. Tokenización y etiquetado con SpaCy

Para este trabajo, se utiliza el modelo *es_core_news_md* de SpaCy, este modelo en Español usa una red neuronal convolucional entrenada con textos de Wikipedia, y posee una exactitud sintáctica del 97.03 % en el etiquetado de categorías gramaticales.

La biblioteca SpaCy provee un objeto documento (DOC), un DOC es una secuencia de objetos Token; un objeto token posee atributos que permiten analizar las palabras de los documentos; la Tabla 3.1 muestra algunos de esos atributos, se resalta la lemmatización del documento "La gente paga por su propia subordinación".

Tabla 3.1: Atributos del objeto Token

Cadena	Lemma	Categoria Gramatical	Sufijo	Está en vocab.	Es stopword
token.text	token.lemma_	token.pos_	token.suffix_	token.is_alpha	token.is_stop
La	La	DET	La	True	True
gente	gente	NOUN	nte	True	False
paga	**pagar**	VERB	aga	True	False
por	por	ADP	por	True	True
su	su	DET	su	True	True
propia	**propio**	DET	pia	True	True
subordinación	subordinación	NOUN	ión	True	False
.	.	PUNCT	.	False	False

3.1.1.1. Vectores de palabras

El modelo en español de SpaCy posee vectores de 50 dimensiones (características) para cada token, entrenados con una red neuronal convolucional con artículos de Wikipedia. Algunos atributos y métodos de los token y documentos para trabajar con vectores de palabras son:

- `has_vector` devuelve True si el token se encuentra en el modelo, y por lo tanto tiene un vector.
- `vector` devuelve el vector de 50 dimensiones, si el token se encuentra en el modelo, en caso contrario, devuelve un vector de ceros.
- `vector_norm` devuelve la norma del vector de 50 dimensiones, si el token se encuentra en el modelo, en caso contrario, devuelve un cero.
- `similarity(token)` y `similarity(doc)` retorna el coseno similitud entre dos vectores 50-dimensionales.

Tabla 3.2: Similitud entre tokens

$Token_1$	$Token_2$	$Norma_1$	$Norma_2$	Similitud	Tiene vector
token.text		token.vector_norm		token1.similarity(token2)	token.has_vector
comerse	comerse	6.0765786	6.0765786	1.0	True
la	un	9.112849	8.556655	0.79384774	True
flecha	cable	5.121496	5.874019	0.3601505	True

Tabla 3.3: Similitud entre documentos

Doc_1	Doc_2	$Norma_1$	$Norma_2$	Similitud
doc.text		doc.vector_norm		token1.similarity(token2)
comerse la flecha	comerse la flecha	5.37776	5.37776	1.0
comerse la flecha	comerse un cable	5.37776	5.60048	0.83351
comerse la flecha	comer cuentos	5.37776	4.97515	0.73159
comerse un cable	comer cuentos	5.60048	4.97515	0.74044

3.1.2. Similitud de cadena con FuzzyWuzzy

La similitud de cadena se refiere al grado de semejanza entre dos cadenas con respecto a sus caracteres individuales, es una forma sintáctica de comparar texto y su uso radica en el emparejamiento. FuzzyWuzzy es una biblioteca ideal para esto, y provee de un método denominado `get_match(query, choices, limit)` que se utilizará en uno de los pasos de etiquetado.

Figura 3.1: Similitud de cadena entre una expresión y las expresiones de la base de conocimientos de EVF.

```
In [3]: from fuzzywuzzy import process

        def get_matches(query, choices, limit = 7):
            results = process.extract(query, choices, limit = limit)
            return results

        get_matches('dar la cola', expresiones)

Out[3]: [('dar la cola', 100),
         ('aflojar la lengua', 86),
         ('agarrar con las manos en la masa', 86),
         ('agarrar la mandarria', 86),
         ('agarrarle el gusto a la cosa', 86),
         ('agarrarlo la pelona', 86),
         ('agarrarse una oreja y no alcanzar la otra', 86)]
```

3.1.3. Conjugación con Pattern

Pattern es una biblioteca utilizada para conjugar verbos, con soporte en español. Será utilizada para las consultas en la creación del corpus de Tweets. El léxico para la conjugación de verbos contiene 600 verbos comunes y para verbos desconocidos posee una precisión del 84 % en la conjugación. [1]

3.1.4. Conexión con el API de Twitter con Tweepy

Para este proyecto solo se utiliza Tweepy para conectar con el API de Twitter:

Figura 3.2: Conexión al API de Twitter usando Tweepy

```
def twitter_setup():
    # Autenticación y acceso usando claves:
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)
    # Retornar API con autenticación, reconexión cuando se pasa tasa de
        consulta:
    api = tweepy.API(auth,
        wait_on_rate_limit=True,wait_on_rate_limit_notify=True)
    return api
```


3.2. Recursos lingüísticos

3.2.1. Diccionario de venezolanismos

Para esta investigación se hizo uso del Diccionario de Venezolanismos [31] como fuente primaria de las expresiones verbales fijas que conforman la base de conocimientos. Este diccionario comenzó como una idea en 1948, cuando Ángel Rosenblat propone la recopilación de los materiales para elaborar el Diccionario Histórico de Venezuela, que daría lugar después al Diccionario de venezolanismos. La

[1] Datos provistos por la página oficial. https://www.clips.uantwerpen.be/pages/pattern-es

recolección de materiales fué trabajo de la Instituto de Filología "Andrés Bello", donde el director de la institución, Rosenblat, impartía instrucciones de como elaborar fichas léxicas, las cuales fueron alimentando el diccionario de información. La mayor fuente de datos del Diccionario está consignada en fichas individuales, en número aproximado a 200.000 y está conformado por tres tomos, que suman 1654 páginas.

3.2.1.1. Artículos del diccionario

Cada ficha o artículo del diccionario posee la siguiente estructura (* indican que puede no estar en el artículo):

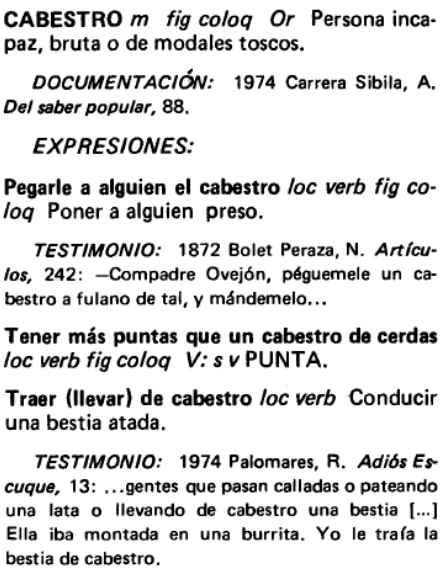
- El lemma: La forma ortográfica mas aceptada de cada unidad léxica.
- La definición: Significado de la entrada del diccionario.
- Las abreviaturas de las partes de la oración y de la localización geográfica.
- La sinonimia*: Si existen dos o más términos con un significado muy parecido o cercano, se remite al más usado.
- La documentación: indicación bibliográfica breve de obras venezolanas en las que esa acepción ha sido definida o estudiada.
- Los testimonios.
- Expresiones idiomáticas y refranes*.

3.2.1.2. Expresiones idiomáticas

En cada artículo del diccionario, si la palabra forma parte de expresiones idiomáticas aparece bajo el título de "Expresión" o "Expresiones". Se consideran expresiones idiomáticas aquellas formadas por mas de dos unidades léxicas. Estas incluyen: las locuciones, frases proverbiales y refranes.

Cada expresiones idiomática se considera como una unidad que permite una subentrada del artículo de la unidad léxica central de esa frase. El interés de este proyecto recae en las locuciones verbales (EVF). Identificadas con la clave *loc verb* dentro del apartado de "Expresiones". (Véase 3.3)

Figura 3.3: Estructura de un artículo del Diccionario de venezolanismos



CABESTRO *m fig coloq Or* Persona incapaz, bruta o de modales toscos.

DOCUMENTACIÓN: 1974 Carrera Sibila, A. *Del saber popular,* 88.

EXPRESIONES:

Pegarle a alguien el cabestro *loc verb fig coloq* Poner a alguien preso.

TESTIMONIO: 1872 Bolet Peraza, N. *Artículos,* 242: —Compadre Ovejón, péguemele un cabestro a fulano de tal, y mándemelo...

Tener más puntas que un cabestro de cerdas *loc verb fig coloq V: s v* PUNTA.

Traer (llevar) de cabestro *loc verb* Conducir una bestia atada.

TESTIMONIO: 1974 Palomares, R. *Adiós Escuque,* 13: ...gentes que pasan calladas o pateando una lata o llevando de cabestro una bestia [...] Ella iba montada en una burrita. Yo le traía la bestia de cabestro.

3.2.1.3. Vigencia de las expresiones verbales fijas

Con el fin de verificar el uso de las expresiones verbales fijas del Diccionario de venezolanismos en la actualidad (debido a la antigüedad del documento), se ha realizado una encuesta a tres distintos grupos de la población, divididos por su edad. Se les ha hecho la pregunta "¿Conoce el significado no literal de la expresión X?", para al menos 350 expresiones de la base de conocimientos; además, se ha preguntado el lugar de procedencia (estado) y si utiliza la red social Twitter.

La encuesta reflejada en la Tabla 3.4 muestra que en promedio un 42.7 % de una muestra aleatoria de 350 expresiones verbales fijas venezolanas fueron reconocidas por el grupo encuestado. 12/14 personas encuestadas utilizaban Twitter, a pesar de la diferencia entre edades. Del total de 750 expresiones, fueron reconocidas un cantidad de 484 expresiones en las muestras aleatorias. Se concluye que si existen expresiones del Diccionario de venezolanismos que siguen siendo conocidas hoy día.

Tabla 3.4: Encuesta de vigencia de las EVF

Persona	Sexo	Estado	Edad	Porcentaje de positivos a la pregunta: ¿Conoce la expresión X?	¿Utiliza Twitter?
1	Femenino	Zulia	62	61.34454 %	Si
2	Femenino	Zulia	61	45.65826 %	Si
3	Masculino	Zulia	63	50.56180 %	No
4	Femenino	Zulia	59	58.42697 %	Si
5	Femenino	Zulia	47	32.77310 %	Si
6	Masculino	Zulia	22	19.60784 %	Si
7	Masculino	Mérida	25	36.97480 %	Si
8	Masculino	Trujillo	29	22.40896 %	Si
9	Femenino	Trujillo	33	68.62745 %	Si
10	Masculino	Trujillo	24	42.01680 %	Si
11	Masculino	Trujillo	23	58.35694 %	Si
12	Masculino	Caracas	35	33.64389 %	No
13	Femenino	Mérida	62	35.57423 %	Si
14	Femenino	Mérida	27	32.77310 %	Si
Total	F:7, M:7			$\bar{x}$ =42.76776 %	

3.3. Creación de la Base de conocimientos de EVF

3.3.1. Extracción de datos en bruto

Se utiliza la biblioteca PyPDF2 para la extracción de los datos en bruto, desde tres documentos PDF, correspondientes a los tomos I, II y III del Diccionario de venezolanismos; debido a la naturaleza del formato PDF, la obtención de información vuelve necesaria la normalización de estos datos, pues se pierde mucha información en la extracción. Se crea una lista de cadenas de caracteres por cada salto de linea, desde el comienzo del Tomo I, y se convierte en su totalidad en minúsculas, para dar inicio al proceso de normalización(véase Figura 3.4).

3.3.2. Normalización de los datos en bruto

Para la limpieza del texto, ha sido necesario remover los términos linguísticos del diccionario que no se utilizarán para la base de conocimientos utilizando algoritmos

Figura 3.4: Lista de cadenas de caracteres después de la extracción

```
Out[16]: [' coloq  mr   v:  cambeto',
          'mano abierta   v: s v mano',
          'abrir gola   v: s v  gola',
          'abrir  un  agujero   para  tapar  otro    v:   s  vaguj e ro',
          'abrirse como un paraguas v: s v para\xad guas',
          'abrirse  la tripa  (cañera)   v:  s v  tr ipa',
          '  v: s   vcar ato',
          '  centr   v:  picu r e',
          '  ap  n esp  v:  acu r ito',
          '  v:acu r ito',
          '  v:  picu r e 1documentación: 1881 rojas, a',
          '  jerg  define  v: ag uantar 3•documentación: 1883 medrana, j',
          'achinar los dientes v: s v di e nte',
          'achiotar   tr   v: onotar',
          '   v:  onot01\nv: onot02•ciembre, a-6:  ',
          '}  escribió  buenos   relatos,  hoy\nv: onot03• 4',
          ' v: onot04• 5',
          'iadiós coroto(s)!   v:  s v  coroto',
          'ad iós luz que te apagaste v: s v luz',
```

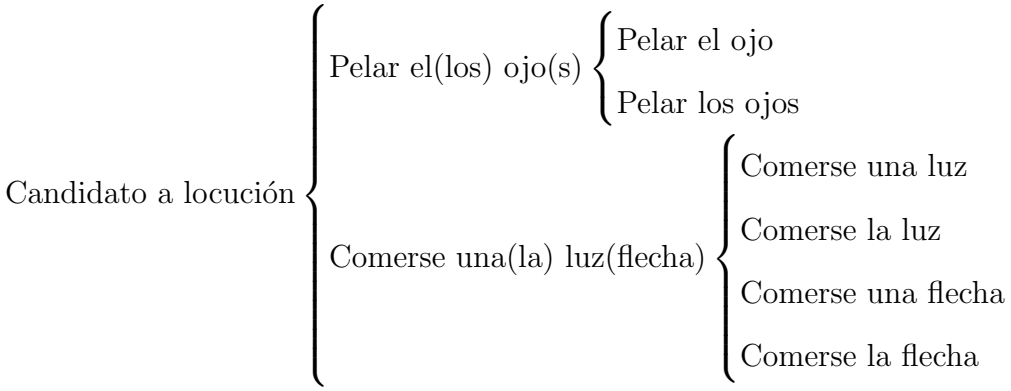
genéricos, de la misma forma se han removido símbolos, números, doble espaciados y signos de puntuación.

Figura 3.5: Primer proceso de normalización

```
Out[19]: ['mano abierta',
          'abrir gola',
          'abrir un agujero para tapar otro',
          'abrirse como un paraguas',
          'abrirse la tripa (cañera)',
          'centr',
          'ap n esp',
          'jerg define',
          'achinar los dientes',
          'achiotar',
          ') escribió buenos relatos, hoy',
          'iadiós coroto(s)!',
          'ad iós luz que te apagaste',
          'adiós paloma turca',
          'iadiós peroles!',
          'adulanta adj ú t e s rúst',
          'af lojar el coroto',
          'af lojar (sele) la lengua',
          'agal loso, a adj',
```

El segundo proceso de normalización (Figura 3.6) consiste en la combinación de las sinonimias de las locuciones semi-fijas para separarlas y convertirlas en expresiones individualmente fijas, siendo así más práctico y viable para su codificación y procesamiento (Véase Figura 3.6).

Figura 3.6: Segundo proceso de normalización



3.3.3. Extracción de expresiones verbales fijas con SpaCy

Cada elemento de la lista de cadenas normalizada se convierte en un objeto DOC, y el primer elemento de la secuencia de tokens indicará si ese elemento es una locución verbal o no.

Para finalizar, se ha revisado la lista de EVF manualmente para arreglar cualquier desperfecto, desde un archivo TXT. Además, se han extraído dos características de la lista de EVF: el núcleo y la estructura o patrón de categorías gramaticales que siguen (Véase Tabla 3.5). La base de conocimientos final está constituida por 750 expresiones verbales fijas.

3.4. Creación del corpus etiquetado de Tweets

Un corpus es un conjunto de textos recopilados cuya función radica en convertirse en proporcionar ejemplos de uso (con sus respectivos contextos) [16]. Para este proyecto, se necesitan ejemplos del uso de las expresiones verbales fijas encontradas en el diccionario de venezolanismos con el objetivo de identificarlas dentro del contexto en texto no etiquetados utilizando aprendizaje automático.

Tabla 3.5: Base de conocimientos de Expresiones Verbales Fijas

Expresión	Núcleo	Estructura
abrir gola	abrir	[VERB, NOUN]
abrir un agujero para tapar otro	abrir	[VERB, DET, NOUN, ADP, VERB, PRON]
abrir un hueco para tapar otro	abrir	[VERB, DET, NOUN, ADP, VERB, PRON]
abrirse como un paraguas	abrirse	[VERB, SCONJ, DET, NOUN]
abrirse la tripa	abrirse	[VERB, DET, NOUN]
abrirse la cañera	abrirse	[VERB, DET, NOUN]
achinar los dientes	achinar	[VERB, DET, NOUN]
aflojar el coroto	aflojar	[VERB, DET, NOUN]
aflojar la lengua	aflojar	[VERB, DET, NOUN]
⋮	⋮	⋮

3.4.1. Metodología propuesta para la extracción de datos en bruto

Se ha utilizado la biblioteca Tweepy para la extracción de los datos en bruto desde Twitter. El corpus inicial está constituido por una lista de 67.445 Tweets, estos fueron obtenidos al realizar 100 consultas exactas[2] de cada expresion verbal fija de la base de conocimiento, y 100 consultas de 46 variaciones del núcleo de la expresión (Véase tabla 3.6), esto se realiza con el fin de conseguir mas datos, ya que no se podrían lemmatizar las búsquedas. Las conjugaciones se obtuvieron con la biblioteca Pattern de Python, y los sufijos mediante algoritmos genéricos.

La estructura del corpus inicial (Tabla 3.7), está formada por la ID, texto, localización e idioma del Tweet, y una etiqueta inicial. Se construye a partir de un objeto Tweet:

- **ID** Se le asigna atributo `tweet.id`, el cual es un entero de 64 bits del objeto Tweet.

[2]Cada consulta se hizo con comillas, para buscar la expresión completa y en orden y no sus palabras por separado.

- **Text** Con el atributo truncated = False del objeto tweet para conseguir el texto completo, si el Tweet tiene retweets, se consigue a partir del atributo `tweet.retweeted_status.full_text`, en caso contrario, `tweet.full_text`.

Tabla 3.6: Variación del núcleo de una expresión verbal fija para las consultas

	Consulta Exacta	**Variación del núcleo**	**Sufijo**
	echar tierra	Infinitivo (Sin variación)	
(le)	echó tierra	Tercera persona-singular-pasado	
	écha**le** tierra	Imperativo	Complemento indirecto
	echar**se** tierra	Infinitivo	Reflexivo
(está)	echándo**se** tierra	Progresivo-tercera persona	Reflexivo
(que)	eches tierra	Subjuntivo-segunda persona-singular	
	⋮	⋮	⋮

- **Localización** Se asigna dependiendo de la existencia de ciertos atributo dentro de un tweet.

El punto de inicio para clasificar un tweet con una expresion verbal fija venezolana (EVFV) es su ubicación. Para esto se toma en primer lugar las coordenadas ofrecidas por el tweet, si las tiene, en segundo lugar, la información de país del tweet, y por último lugar la localización ofrecida por el usuario, siendo esta la menos precisa ya que puede contener cualquier texto. Más detalles en el diagrama de la Figura A.3 en apéndices.

- **Idioma** A partir `tweet.lang`, que normalmente sera "es" ya que las consultas contienen palabras en español.

3.4.1.1. Resultados de la extracción y tiempo invertido

Solo se obtuvieron resultados a la consulta exacta de 393 EVFV (contando las conjugaciones del núcleo como la misma expresión). La figura A.4 detalla la metodología usada para la creación del corpus sin etiquetar.

Es posible calcular un estimado del tiempo de extracción, partiendo del número de consultas al API de Twitter:

Tabla 3.7: Estructura del Corpus de textos inicial

ID	**Tweet(Texto)**	Localización	Idioma	Etiqueta
1181713509668868096	cuando esa monja muera y llegue a puertas del ...	barrancabermeja	es	Untagged
1234567890987654321	soy un tweet creado en el centro de Venezuela	6.4237499, -66.5897293	es	Untagged
0000000000000000000	tweet sin coordenadas pero con tweet.place	Honduras	es	Untagged
6666666666666666666	tweet con solo la localización escrita por el usuario	In the hell	es	Untagged
⋮	⋮	⋮	⋮	⋮

$$C = \sum_{i=0}^{n} 100 \times EV_i \tag{3.1}$$

donde C es el numero de consultas, n es el número de expresiones en la base de conocimientos; EVi es el numero de variaciones de la expresión, el cual es distinto para los verbos reflexivos y no reflexivos, transitivos e intransitivos[3]; 100 es el número máximo de consultas que permite twitter por búsqueda. Para saber las consultas con resultados:

$$C_R = \sum_{i=0}^{n} (R_i \times EV_i) \qquad 0 \leq R \leq 100, R \in \mathbb{Z} \tag{3.2}$$

donde C_R es el numero de consultas que arrojaron resultados, R es el número de resultados obtenidos de 100 consultas.

$$T_e(t_c) = C \times t_c + t_w \qquad t_c \in (0, t_{max}] \tag{3.3}$$

donde T_e es el tiempo de extracción, t_c es el tiempo de consulta, t_{max} es el tiempo máximo que tarda una computadora en hacer una consulta y t_w es la suma de los tiempos de espera al alcanzar la tasa de consultas por hora del API de Twitter.

[3]Para este proyecto se asumió que todos los verbos son transitivos. Un ejemplo de verbo intransitivo es morir, donde variaciones como "morirles" o "morirte", no existen, ya que no tienen complemento indirecto. Naturalmente, estos no arrojan resultados en la búsqueda, así que solo añaden mas tiempo de ejecución. En [60] se explica con gran detalle la diferencia entre verbo transitivo y no transitivo, pero es irrelevante para este trabajo.

El número de consultas realizadas fue de 3.525.000 aproximadamente, y la extracción tardó 3.8875 días o 93.3 horas (335880.56 segundos). Al eliminar tweets repetidos se obtuvo la cantidad de 35.825 tweets.

3.4.1.2. Normalización

La normalización de un Tweet difiere de la normalización estándar ya que hay que tomar en cuenta los componentes de un Tweet, se siguió el siguiente proceso de normalización:

1. Remoción de emojis, caracteres unicode (excepto banderas) y signos de puntuación.
2. Reemplazo de tildes y diéresis (á→a,ü→u) y conversión de las cadenas a minúsculas.
3. Eliminación total de menciones (el símbolo @ y el nombre de usuario), hashtags (símbolo # y la palabra clave) e hipervínculos (utilizando expresiones regulares). Solo para la columna "Tweet" en el corpus.
4. Reemplazo de las banderas unicode por el nombre de país. Solo para la columna "Localización".
5. Se utilizó la biblioteca OpenCageGeocode para convertir las coordenadas en una dirección exacta. Solo para la columna "Localización".

3.4.2. Metodología propuesta para el etiquetado del corpus de textos

Después del proceso de normalización, el corpus pasa por un proceso de etiquetado utilizando la localización de los tweets y algunas palabras claves. La localización es esencial para el etiquetado del conjunto de datos, puesto que las expresiones poseen un contexto regional, por otro lado, las palabras claves han sido objeto de uso para el etiquetado de información en distintas investigaciones [62, 63] y han resultado útiles para refinar las etiquetas.

3.4.2.1. Etiquetado

Cada elemento del corpus está etiquetado como Untagged por defecto. Luego se cambian las etiquetas, en orden, de acuerdo a los siguientes casos:

1. Etiqueta True, si al menos una palabra clave de lugar de Venezuela aparece dentro de la cadena en la columna "Localización". Las palabras claves incluyen:
   - Nombre de Venezuela: *venezuela, vzla, vnzla.*
   - Nombres de estados: *aragua, zulia, merida.*
   - Nombres de capitales y ciudades importantes: *maracay, valera, margarita, caracas.*
2. Etiqueta False, si al menos una palabra clave de lugar internacional aparece dentro de la cadena en la columna "Localización". Este paso evita que se etiqueten erróneamente elementos como "verdadero" cuando tienen localizaciones como "Mérida, México". Las palabra claves incluyen:
   - Nombres de países hispanos: *mexico, mx, colombia.*
   - Nombres de capitales de paises hispanos: *bogota, quito.*
3. Etiqueta True, si al menos una palabra clave de modismo de Venezuela aparece dentro de la cadena en la columna "Tweet". Son tomadas del diccionario de venezolanismos. Este paso evita que se etiqueten como falsos elementos donde los Tweets son de venezolanos en el exterior y las palabras claves incluyen:
   - Expresiones de una palabra: *naguara, chamo...*
   - Vulgaridades regionales.
4. Etiqueta False, si al menos una palabra clave de modismo internacional aparece dentro de la cadena en la columna "Tweet". Las palabras claves incluyen:
   - Palabras ajenas a la región: *orale, fome, pibe, boludo.*[4,5,6,7]

[4]https://www.thisischile.cl/modismos-chilenosde-la-a-a-la-z/
[5]http://espanolyohablo.blogspot.com/2013/08/modismos-argentinos_698.html
[6]https://www.elblogdeyes.com/modismos-mexicanos/
[7]Solo se tomaron 30 palabras.

5. Las filas que quedan con la etiqueta untagged han sido eliminadas. Quedando el corpus con 15051 elementos. La proporción de verdaderos y falsos es de 9:11.

En la ultima sección del proyecto se puede ver gráficamente el proceso (A.5).

Tabla 3.8: Estructura del corpus de textos etiquetado

ID	**Tweet(Texto)**	Etiqueta	Expresión
1181713509668868096	cuando esa monja muera y llegue a puertas del ...	True	Pelar Cacao
1234567890987654321	soy un tweet creado en el centro de Venezuela	True	EVF
0000000000000000000	tweet sin coordenadas pero con tweet.place	False	EVF
6666666666666666666	tweet con solo la localización escrita por el usuario	False	EVF
2222222222222222222	ayer me dieron la cola de conejo que usan para la suerte	False	EVF
⋮	⋮	⋮	⋮

3.5. Creación del corpus de EVF

3.5.1. Metodología propuesta para la creación del corpus de EVF

Para saber si un documento posee una expresión verbal fija venezolana mediante su contexto, es necesario saber si el documento posee expresiones fijas. En esta sección se propone una metodología para la caracterización de estas expresiones con el fin de entrenar un modelo que las identifique como EVF a priori de clasificarlas como EVF venezolanas.

3.5.1.1. Extracción de patrones de categorías gramaticales

Con el corpus de Tweets mostrado en la tabla 3.8, y utilizando la estructura de las expresiones de la base de conocimientos (Tabla 3.5) es posible crear una lista de

los patrones que aparecen (véase Tabla 3.9). Este proceso se inspira en la metodología propuesta en [33] en 2015[8]. En este caso, para la identificación de los patrones se ha utilizado el modelo en español de SpaCy que permite extraer la categoría gramatical de un token.

Tabla 3.9: Lista de patrones de categorías gramaticales encontrados en la base de conocimientos.

#	**Patrón de categorías gramaticales**
1	[VERB, NOUN]
2	[VERB, DET, NOUN, ADP, VERB, PRON],
3	[VERB, SCONJ, DET, NOUN]
4	[VERB, DET, NOUN]
5	[VERB, ADP, DET, NOUN, ADP, DET, NOUN]
6	[VERB, DET, NOUN, ADP, DET, NOUN]
7	[VERB, ADP, DET, NOUN]
8	[VERB, DET, NOUN, CONJ, ADV, VERB, DET, PRON]
⋮	⋮
105	[VERB, ADV, AUX, VERB]

Para evaluar cada patrón con cada Tweet con el objetivo de adquirir candidatos a EVF para el corpus de EVF etiquetado, el proceso empieza convirtiendo cada Tweet en una lista de duplas. Como el coste computacional de convertir el corpus de Tweets, es demasiado alto se ha utilizado una fracción del mismo, asegurandose que contiene al menos una de las expresiones que ahí aparecen.

3.5.1.2. Extracción de patrones de un Tweet

Considérense el conjunto de Tweets:

$$T(w) = \{t_1, t_2, ..., t_n\} \; donde \; w \in \Sigma \tag{3.4}$$

[8]En el trabajo de Sánchez se identifican etiquetas morfo-sintácticas utilizando una herramienta llamada FreeLing.

Donde t_i es un Tweet, w una palabra y Σ es el conjunto de palabras del idioma Español. Y el conjunto de patrones:

$$P(k) = \{p_1, p_2, ..., p_n\} \; donde \; k \in \Omega \tag{3.5}$$

Donde p_i es un patrón, k una categoría gramatical y Ω el conjunto de todas las categorías gramaticales. L es un conjunto de duplas:

$$L(k, w) = \{(k_1, w_1), (k_2, w_2), ..., (k_n, w_m)\} \; donde \; \forall w \exists k \tag{3.6}$$

Entonces el corpus será un conjunto de expresiones $E(w) = \{e_1, e_2, ..., e_j\}$:

$$Corpus = E(w) \Big| \Big(p_i(k), e_j(w)\Big) \in L(k, w) \; si \; i = j \tag{3.7}$$

Ejemplo de extracción de patrón del conjunto de Tweets

Sea el conjunto unitario $T(w) = \{t_1\}$

$$t_1 = \text{“Hace rato fui a una panadería y le pregunté a mí mamá si me podía dar la cola hasta el central”} \tag{3.8}$$

$P(k)$ es el conjunto de patrones en la lista de patrones de la Tabla 3.9. Entonces L será:

$L(k, w)$ = {(AUX, Hace), (NOUN, rato), (AUX, fui), (ADP, a), (DET, una), (NOUN, panadería), (CONJ, y), (PRON, le), (VERB, pregunté), (ADP, a), (PRON, mí), (NOUN, mamá), (SCONJ, si), (PRON, me), (AUX, podía), (VERB, dar), (DET, la), (NOUN, cola), (ADP, hasta), (DET, el), (NOUN, central)}

El conjunto de patrones pertenecientes a L y a $P(k)$ es:

$\widehat{P}(k)$ = {(VERB, DET, NOUN), (VERB, DET, NOUN, ADP, DET, NOUN), (AUX, NOUN), (AUX, VERB, DET, NOUN)}

Por lo tanto, el conjunto de expresiones pertenecientes a L es:

corpus $= E(w)$ = {dar la cola, dar la cola hasta el central, hace rato, podía dar la cola} para $\Big(p_i(k), e_j(w)\Big) \in L(k, w)$.

3.5.2. Metodología propuesta para el etiquetado del Corpus de EVF

Cada documento del corpus sin etiquetar, es etiquetado como EVF si el valor de Similitud de Expresión (S_E) es mayor a 90 % siguiendo el siguiente proceso de validación:

1. **Lemmatización del núcleo**: Ya que el núcleo de cada expresión es el que suele cambiar, se lemmatiza el núcleo (verbo) del candidato a expresión, de esta forma, las conjugaciones ya no resultan un problema al momento de calcular la similitud. (Véase Figura 2.5)

2. **Similitud Coseno**: S es el resultado de conseguir el valor máximo de todos los valores devueltos por `doc1.similarity(doc2[i])` donde DOC1 es el candidato a EVF del corpus etiquetado y DOC2 son cada una de las expresiones de la base de conocimientos.

$$S = S_{MAX} \tag{3.9}$$

3. **Cardinalidad del documento:** Si DOC1 posee mas o menos tokens que DOC2, el valor de similitud se castiga en un 5 % si la diferencia de tamaño es 1 y 10 % si la diferencia de tamaño es mayor a 1. Se premia un 1 % si son del mismo tamaño.

$$S^* = \begin{cases} S - S(0.05) & si\ |Card(doc1) - Card(doc2)| = 1 \\ S - S(0.1) & si\ |Card(doc1) - Card(doc2)| > 1 \\ S + S(0.01) & si\ Card(doc1) = Card(doc2) \end{cases} \tag{3.10}$$

   Esta regla permite comparar el tamaño de las expresiones, el candidato a expresión podría contener una palabra agregada que no pertenece a la locución verbal pero que fue extraída debido a un patrón equivocado.

4. **Similitud de cadena de los componentes**: La similitud de la expresión será el promedio entre la similitud de cadena de los componentes (documento sin núcleo) de DOC1 entre los componentes de DOC2 y la similitud S^*

$$S_E = \frac{S^* + S_{cadena}}{2} \tag{3.11}$$

Esta regla permite comparar la fijación de dos expresiones; ya que el núcleo no se incluye y se supone que las locuciones no pueden variar sus componentes.

Tabla 3.10: Proceso de etiquetado de un candidato a EVF

Candidato a Etiquetar	Núcleo	Componentes	S^*	Expresión Posible	S_{cadena}	S_E	EVF
dió la cola	dar	la cola	0.9748	dar la cola	1.0	0.9874	True
dar la cola hasta el central	dar	la cola hasta el central	0.9849	coger ese trompo en la uña	0.35	0.6674	False
hace rato	hacer	rato	0.9750	hacer cachapas	0.36	0.6675	False
podía dar la cola	poder	dar la cola	0.9702	dar con la espuela	0.53	0.7501	False
dar el rabo	dar	el rabo	1.0697	dar el palo	0.71	0.8899	False
dar las colas	dar	las colas	1.0002	dar la cola	0.88	0.9401	True
darnos la cola	darnos	las colas	0.9844	dar la cola	1.0	0.9922	True
kkkkkk sdfsdf	kkkkkk	sdfsdf	0	Null	0	0	False

Este proceso de etiquetado ya constituye una forma de identificación de expresiones verbales fijas, sin embargo, jamás podría identificar expresiones nuevas debido a su cualidad restrictiva. Al finalizar el etiquetado se obtuvieron 385 expresiones clasificadas como EVF que no están incluidas en el corpus etiquetado. Con el corpus ya terminado, el siguiente capítulo tratará sobre el entrenamiento de modelos de clasificación haciendo uso del corpus de EVF y el corpus de texto.

Capítulo 4

Construcción del modelo

En este capítulo se explican los pasos necesarios para al entrenamiento de los datos del corpus etiquetado, las dependencias utilizadas y el desempeño de los algoritmos. Así como también el procedimiento que deben seguir los datos desconocidos para ponerse a prueba.

4.1. Técnicas de extracción de características

Para entrenar un conjunto de datos, es necesario recopilar las características que distinguen a esos datos. Para este proyecto, se deben extraer características a los documentos del corpus de textos y del corpus de EVF. Se ha utilizado la biblioteca scikit-learn que provee de varios métodos para conseguir los modelos para los vectores de características [64]. Primero considérense dos documentos D_1 y D_2 a evaluar:

D_1 = "se acaba el tiempo"

D_2 = "el tiempo de Dios es perfecto"

4.1.1. Vector binario

Con este método, para todas las palabras dentro del vocabulario, si la palabra aparece en un documento al menos una vez, es contada con un positivo, si la palabra no aparece, es contada como negativo. Como no toma en cuenta la frecuencia en la que una palabra aparece, existe la posibilidad de perder información [65]

Tabla 4.1: Vectores binarios.

Documento	se	acaba	el	tiempo	de	Dios	es	perfecto
se acaba el tiempo	1	1	1	1	0	0	0	0
el tiempo de Dios es perfecto	0	0	1	1	1	1	1	1

4.1.2. Vector de conteo (CV)

Este método funciona como el vectorizador binario, pero cuenta también la frecuencia en la que aparece una palabra en el documento. Para efectos demostrativos se ha cambiado D_1 para que tenga una palabra repetida.

Tabla 4.2: Vectores de conteo.

Documento	a	el	se	le	acaba	tiempo	de	Dios	es	perfecto
a el se le acaba el tiempo	1	2	1	1	1	1	0	0	0	0
el tiempo de Dios es perfecto	0	1	0	0	0	1	1	1	1	1

4.1.3. TF-IDF

Es una de las técnicas más usadas para el proceso de datos en forma de texto, el nombre viene del inglés Term Frecuency-Inverse Data Frecuency, traducido al español como Frecuencia de Términos - Frecuencia Inversa de Documentos [66, 67]. Su utilidad radica en covertir un documento en un formato estructurado, de manera que refleje numéricamente que tan importante es una palabra para un documento del corpus [68]. Se calcula a partir de dos conceptos distintos:

Frecuencia de términos (TF) Esta definición resuelve la frecuencia de una palabra dentro de un documento en el corpus, el valor de TF aumenta cuando el numero de ocurrencia de una palabra dentro de un documento aumenta, y cada documento tiene su propia TF. Se define de la siguiente manera:

$$tf_{i,j} = \frac{n_{i,j}}{\sum n_{i,j}} \tag{4.1}$$

donde $tf_{i,j}$ es el número de ocurrencias de i en j

Frecuencia Inversa de Documento (IDF) Se utiliza para calcular el peso de palabras poco frecuentes dentro de todos los documentos del corpus, estas palabras tendran un IDF alto. Se define de la siguiente forma:

$$idf(w) = \log\left(\frac{N}{df_i}\right) \tag{4.2}$$

donde df_i es el número de documentos que contienen i y N es el número de documentos.

Combinando ambas definiciones se obtiene el TF-IDF de una palabra en un documento del corpus:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{4.3}$$

Ejemplo de TF-IDF

Sean D_1 y D_2:

El valor TF-IDF de las palabras comunes será cero si son poco significativas. Ver Tabla 4.3

Tabla 4.3: Cálculo de TF-IDF.

Palabra	TF		IDF	TF-IDF	
	D_1	D_2		D_1	D_2
se	1/4	0	$\log(2/1) = 0.301$	0.075	0
acaba	1/4	0	$\log(2/1)$	0.075	0
el	1/4	1/6	$\log(2/2)$	0	0
tiempo	1/4	1/6	$\log(2/2)$	0	0
de	0	1/6	$\log(2/1)$	0	0.075
Dios	0	1/6	$\log(2/1)$	0	0.075
es	0	1/6	$\log(2/1)$	0	0.075
perfecto	0	1/6	$\log(2/1)$	0	0.075

4.1.4. Vector de características híbrido

En [69] se presenta una técnica para la clasificación de textos utilizando características semánticas. Esto se realiza mediante la concatenación de un vector de características creado a partir de técnicas de conteo y frecuencia, con un vector de palabras pesado con TF-IDF.

Para este proyecto se utilizará el modelo en español de SpaCy de vectores de 50 dimensiones de manera similar, pero mucho mas simple:

Sea WV la representación en vectores de palabras de un documento, y XV el vector de características de un documento obtenido con TF-IDF o CV. El vector de características híbrido será la concatenación de WV con XV

Tabla 4.4: Vectores híbridos con vectores de palabras de dos dimensiones.

Documento	a	el	se	le	acaba	tiempo	de	Dios	es	perfecto	**Vector de palabras**	
a el se le acaba el tiempo	1	2	1	1	1	1	0	0	0	0	-1.345	2.345
el tiempo de Dios es perfecto	0	1	0	0	0	1	1	1	1	1	1.342	1.111

4.2. Creación de los conjuntos de datos

El conjunto de datos se diferencia del corpus porque el corpus ofrece información lingüistica, mientras que el conjunto de datos contiene valores numéricos. Sin embargo, es necesario aclarar que suelen ser utilizados como sinónimos, para este proyecto se justifica la distinción ya que los modelos utilizan distintos conjuntos de datos, pero el mismo corpus. En [70] realizan esta distinción al definir el corpus como un ejemplo representativo de un idioma para un propósito general, mientras que el conjunto de datos representa investigaciones sobre preguntas específicas.

Se han utilizado las técnicas de extracción de características de la sección anterior para ambos corpus. El primer conjunto de datos se obtiene a partir del corpus de EVF, con una dimensión de 10689 observaciones x 6137 características con bloques de 1 token. El segundo conformado por la extracción de características del corpus de Tweets, con 14405 observaciones x 30538 características con bloques de 1 token.

Los subconjuntos III-i corresponden a algunas de las expresiones individuales del corpus de Tweets con más observaciones. Mientras que los subconjuntos IV-i corresponden a expresiones individuales etiquetadas manualmente del corpus sin etiquetar de 35.825 Tweets. Las dimensiones y proporción de las etiquetas para cada conjunto de datos son detalladas en la Tabla 4.5.

Tabla 4.5: Conjuntos de datos. Características extraidas con vectores de conteo.

Conjunto de datos	Expresiones	Dimensión		Etiquetas	
		Observaciones	Características	True	False
I	n/a	10689	6137	37 %	63 %
II	Todas las encontradas	14405	30538	45 %	55 %
III-1	parar bola	862	3980	53 %	47 %
III-2	meterse a	592	3822	29 %	71 %
III-3	caer encima	349	2484	38 %	62 %
III-4	mentar la madre	302	1931	30 %	70 %
III-5	estar hecho	278	2159	38 %	62 %
III-6	dar el golpe	236	2131	27 %	73 %
III-7	sacar la piedra	157	1027	44 %	56 %
III-8	estar limpio	154	1330	32 %	68 %
III-9	dar la cola	95	772	83 %	17 %
III-10	poner la torta	70	512	81 %	19 %
IV-1	caer encima	881	4705	74 %	26 %
IV-2	estar limpio	154	1330	12 %	88 %
IV-3	dar la cola	162	1157	78 %	22 %

4.3. Modelos propuestos

Se proponen dos modelos distintos, correspondientes al modelo de caracterización de EVF y al modelo de identificación de EVFV.

- **Modelo para caracterizar EVF**, entrenado a partir del conjunto de datos I. Este modelo predice si un candidato (el cual es previamente extraído con la ayuda

del modelo entrenado de SpaCy, de acuerdo a sus categorías gramaticales) es una expresión verbal fija.

- **Modelo para identificar EVFV**, toma de base el corpus de Tweets. Este modelo predice si una EVF dentro de un contexto dado es venezolana. Y se proponen tres maneras de construirlo:
  - **Generalizado:** Incluye todas las expresiones encontradas, entrenado con el conjunto de datos II.
  - **Particularizado:** se han utilizado 10 expresiones frecuentes del corpus. Se entrena con cada subconjunto de datos III
  - **Particularizado etiquetado manualmente:**, es igual al anterior, pero es etiquetado por personas, y no con la metodología descrita en el capítulo 3. Se han utilizado solo 3 expresiones, corresponden al entrenamiento de los subconjuntos de datos IV.

4.4. Validación del modelo

La construcción de un modelo depende de la elección del algoritmo que mejor rendimiento tiene para una determinada tarea. El éxito de esta elección recae en el correcto ajuste de los parámetros que constituyen un modelo, uno de los más utilizados en la ingeniería de datos es la validación cruzada [26].

4.4.1. Variación de hiperparámetros para la extracción de características

Las métodos de extracción de características poseen parámetros que cambian su comportamiento y que pueden o no mejorar el desempeño del algoritmo de clasificación. Para los modelos propuestos se han utilizado tres de las técnicas mencionadas en el apartado anterior, estos son algunos de los parámetros que se han tomado en cuenta:

- **Rango de n-gramas:** `ngram_range = (min,max)`, los n_gramas son los bloques de tokens que se cuentan. Es posible utilizar varios tamaños de gramas en una

sola extracción. El valor mín y max indica los tamaños mínimo y máximo de cada bloque respectivamente. La ventaja del uso de n-gramas es que permite tomar en cuenta patrones continuos, pero las características tendrán un grado de fijación mayor.

Tabla 4.6: Variación del rango de n-gramas para la tokenización de D_1

`ngram_range`	Bloque de tokens						
(1,1)	se	acabo	el	tiempo			
(1,2)	se acabo	acabo el	el tiempo	se	acabo	el	tiempo
(2,3)	se acabo el	acabo el tiempo	se acabo	acabo el	el tiempo		

- **Stopwords:** Indica si se incluyen o no las palabras vacías. Este parámetro permite extraer características tomando en cuenta solo las palabras con significado único.
- **Usar IDF:** Es un parámetro binario que indica la inclusión de la frecuencia inversa de documentos (IDF) dentro de TF-IDF. La exclusión de IDF permite tomar en cuenta las palabras comunes dentro del corpus. Solo aplica para TF-IDF.
- **Máx. características:** Controla el número máximo de características a usar. Si el número de características es mucho mas grande que el de observaciones el modelo puede sobreajustarse.

4.4.2. Variación de hiperparámetros de los clasificadores

Análogo a los métodos de extracción de características, los algoritmos de clasificación DT y SVM poseen una configuración paramétrica que puede ajustarse con el fin de mejorar la tasa de clasificación, los principales son:

- **Máquinas de Vectores de Soporte**
  - **Kernel:** Lineal, Radial o Polinomial.
  - **Grado:** Para con el kernel polinomial, cuando el grado es mayor la superficie de decisión es mas flexible.

- **C:** Es el parámetro de regularización. Regula el compromiso entre una mala clasificación de los ejemplos de entrenamiento contra la simplicidad de la superficie de decisión. Si el valor es bajo generaliza más, y cuando el valor es alto, restringe.
- **Gamma:** Para los kernels polinomial y radial. Define que tan lejos llega la influencia de un solo ejemplo de entrenamiento, con valores bajos que significan 'lejos' y valores altos que significan 'cerca'.

▪ **Árboles de decisión binarios**

- **Profundidad máxima:** Mientras mas profundo es el árbol, mas divisiones tiene y captura más información sobre los datos. Pero si el valor es demasiado alto el modelo puede sobreajustarse.

▪ **Impulso Adaptativo**

- **Estimador base** clasificador a adaptar, normalmente un árbol.
- **Numero de estimadores:** determina el número de árboles secuenciales (estimador base) a ser modelados
- **Tasa de Aprendizaje:** Indica el impacto del estimador en el resultado final de entrenamiento. Cuando los valores son altos es posible que el modelo se sobreajuste, por eso se prefieren valores pequeños.

4.4.3. Validación cruzada

La validación cruzada consiste en una prueba de rendimiento de los algoritmos de acuerdo a distintos ajustes de parámetros. No solo se han tomado en cuenta los hiperparámetros de los clasificadores, sino también los de los métodos de extracción de características. Para esta prueba el conjunto de datos se dividió en cinco segmentos.

4.4.3.1. Validación del modelo de caracterización de EVF

Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

- **SVM:** Kernel: radial, C = 1, Gamma = 0.5.
- **DT:** Profundidad Máxima = 42, Criterion: 'gini'.
- **AB:** Criterion (Estimador base): 'entropy', Splitter(Estimador base): 'best', Tasa de aprendizaje = 1, Número de estimadores = 50.

Tabla 4.7: Exactitud de cada segmento de prueba de la validación cruzada para el modelo de caracterización de EVF

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	**CV(1,2)**	**76.35 %**	**75.48 %**	**75.94 %**	**74.67 %**	**76.54 %**	**75.80 %**	**±2 %**
DT	CV(1,2)	74.48 %	73.55 %	72.93 %	72.39 %	73.33 %	73.34 %	±1 %
AB	CV(1,2)	74.08 %	71.68 %	73.13 %	71.06 %	72.93 %	72.57 %	±2 %

4.4.3.2. Validación del modelo de identificación de EVFV: Generalizado

Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

- **SVM:** Kernel: radial, C = 100, Gamma = 0.001.
- **DT:** Profundidad Máxima = 42, Criterion: 'gini'.
- **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.5, Número de estimadores = 50.

Tabla 4.8: Exactitud de cada segmento de prueba de la validación cruzada para el modelo generalizado de identificación de EVFV.

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	66.73 %	65.89 %	66.93 %	63.54 %	63.59 %	65.34 %	±3 %
DT	CV(1,1)	61.08 %	65.05 %	61.77 %	61.46 %	62.30 %	62.04 %	±1.7 %
AB	**CV(1,1)**	**65.05 %**	**65.39 %**	**66.33 %**	**63.99 %**	**65.67 %**	**65.47 %**	**±1 %**

4.4.3.3. Validación del modelo de identificación de EVFV venezolanas: Particularizado

- **Expresión "parar bola"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 10, Gamma = 0.01.
  - **DT:** Profundidad Máxima = 2, Criterion: 'entropy'.
  - **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.05, Número de estimadores = 50.

Tabla 4.9: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "parar bola".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	63.11 %	65.29 %	59.17 %	60.00 %	62.50 %	62.02 %	±4.4 %
DT	CV(1,1)	66.39 %	74.38 %	58.33 %	65.00 %	70.83 %	67.00 %	±10.7 %
AB	CV(1,1)	66.39 %	66.94 %	58.33 %	62.50 %	72.50 %	65.34 %	±9 %

- **Expresión "meterse a"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.0001.
  - **DT:** Profundidad Máxima = 2, Criterion: 'entropy'.
  - **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.05, Número de estimadores = 150.

- **Expresión "caer encima"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

Tabla 4.10: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "meterse a".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	68.67 %	68.67 %	68.67 %	68.67 %	69.51 %	68.84 %	±1 %
DT	CV(1,1)	69.88 %	68.67 %	68.67 %	68.67 %	67.47 %	69.51 %	±1.3 %
AB	CV(1,1)	63.86 %	60.24 %	60.24 %	60.24 %	74.39 %	63.77 %	±10 %

- **SVM:** Kernel: polinomial, C = 10, Gamma = 0.01, Grado = 3.
- **DT:** Profundidad Máxima = 2, Criterion: 'gini'.
- **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.1, Número de estimadores = 100.

Tabla 4.11: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "caer encima".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	60.00 %	57.14 %	61.22 %	66.67 %	60.42 %	61.07 %	±6.2 %
DT	CV(1,1)	62.00 %	59.18 %	61.22 %	62.50 %	62.59 %	61.48 %	±2.5 %
AB	CV(1,1)	66.00 %	59.18 %	57.14 %	64.58 %	58.33 %	61.07 %	±7.1 %

- **Expresión "mentar la madre"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: polinomial, C = 1000, Gamma = 0.01, Grado = 5.
  - **DT:** Profundidad Máxima = 5, Criterion: 'entropy'.
  - **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'random', Tasa de aprendizaje = 1, Número de estimadores = 300.

- **Expresión "estar hecho"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

Tabla 4.12: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "mentar la madre".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	74.42 %	76.19 %	73.81 %	76.19 %	73.81 %	74.88 %	±2.2 %
DT	CV(1,1)	72.09 %	73.81 %	73.81 %	71.43 %	73.81 %	72.99 %	±2 %
AB	CV(1,1)	69.77 %	66.67 %	66.67 %	69.05 %	66.67 %	67.77 %	±2.7 %

- **SVM:** Kernel: polinomial, C = 1000, Gamma = 0.01, Grado = 5.
- **DT:** Profundidad Máxima = 5, Criterion: 'entropy'.
- **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'random', Tasa de aprendizaje = 1, Número de estimadores = 300.

Tabla 4.13: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "estar hecho".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	74.42 %	76.19 %	73.81 %	76.19 %	73.81 %	74.88 %	±2.2 %
DT	CV(1,1)	72.09 %	73.81 %	73.81 %	71.43 %	73.81 %	72.99 %	±2 %
AB	CV(1,1)	69.77 %	66.67 %	66.67 %	69.05 %	66.67 %	67.77 %	±2.7 %

- **Expresión "dar el golpe"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.0001.
  - **DT:** Profundidad Máxima = 3, Criterion: 'gini'.
  - **AB:** Criterion (Estimador base): 'entropy', Splitter(Estimador base): 'best', Tasa de aprendizaje = 1, Número de estimadores = 100.

- **Expresión "sacar la piedra"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

Tabla 4.14: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "dar el golpe".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	73.53 %	73.53 %	72.73 %	75.00 %	75.00 %	73.94 %	±1.8 %
DT	CV(1,1)	73.53 %	70.59 %	69.70 %	75.00 %	78.12 %	73.33 %	±6.1 %
AB	CV(1,1)	70.59 %	73.53 %	66.67 %	75.00 %	68.75 %	70.91 %	±6.1 %

- **SVM:** Kernel: radial, C = 10, Gamma = 0.5.
- **DT:** Profundidad Máxima = 17, Criterion: 'entropy'.
- **AB:** Criterion (Estimador base): 'entropy', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.1, Número de estimadores = 100.

Tabla 4.15: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "sacar la piedra".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	56.52 %	63.64 %	59.09 %	61.90 %	57.14 %	59.63 %	±5.5 %
DT	CV(1,1)	34.78 %	54.55 %	63.64 %	76.19 %	66.67 %	58.72 %	±28.3 %
AB	CV(1,1)	52.17 %	50.00 %	54.55 %	61.90 %	66.67 %	56.88 %	±12.4 %

- **Expresión "estar limpio"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.1.
  - **DT:** Profundidad Máxima = 3, Criterion: 'gini'.
  - **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'best', Tasa de aprendizaje = 0.5, Número de estimadores = 150.

- **Expresión "dar la cola"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

Tabla 4.16: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "estar limpio".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	68.18 %	72.73 %	68.18 %	76.19 %	70.00 %	71.03 %	±6.1 %
DT	CV(1,1)	68.18 %	77.27 %	72.73 %	71.43 %	70.00 %	71.96 %	±6.2 %
AB	CV(1,1)	63.64 %	81.82 %	81.82 %	61.90 %	65.00 %	71.03 %	±18.1 %

- **SVM:** Kernel: radial, C = 1000, Gamma = 0.0001.
- **DT:** Profundidad Máxima = 23, Criterion: 'entropy'.
- **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'best', Tasa de aprendizaje = 1, Número de estimadores = 50.

Tabla 4.17: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "dar la cola".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	85.71 %	84.62 %	84.62 %	84.62 %	84.62 %	84.85 %	±0.9 %
DT	CV(1,1)	71.43 %	84.62 %	100.00 %	84.62 %	76.92 %	83.33 %	±19.3 %
AB	CV(1,1)	78.57 %	84.62 %	84.62 %	76.92 %	84.62 %	81.82 %	±6.8 %

- **Expresión "poner la torta"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.0001.
  - **DT:** Profundidad Máxima = 39, Criterion: 'gini'.
  - **AB:** Criterion (Estimador base): 'entropy', Splitter(Estimador base): 'random', Tasa de aprendizaje = 1, Número de estimadores = 100.

Tabla 4.18: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular de identificación de la expresión "poner la torta".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	80.00 %	80.00 %	80.00 %	80.00 %	77.78 %	79.59 %	±1.7 %
DT	CV(1,1)	80.00 %	90.00 %	90.00 %	90.00 %	88.89 %	87.76 %	±7.9 %
AB	CV(1,1)	80.00 %	80.00 %	90.00 %	90.00 %	88.89 %	85.71 %	±9.5 %

4.4.3.4. Validación del modelo de identificación de EVFV: Particularizado etiquetado manualmente

- **Expresión "caer encima"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.2.
  - **DT:** Profundidad Máxima = 4, Criterion: 'gini'.
  - **AB:** Criterion (Estimador base): 'entropy', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.5, Número de estimadores = 50.

Tabla 4.19: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular etiquetado manualmente de identificación de la expresión "caer encima".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	75.00 %	74.19 %	74.80 %	73.98 %	74.59 %	74.51 %	±0.8 %
DT	CV(1,1)	88.78 %	72.58 %	74.19 %	73.17 %	64.23 %	74.59 %	±7.7 %
AB	CV(1,1)	69.35 %	75.00 %	72.36 %	71.54 %	74.59 %	72.56 %	±4.1 %

- **Expresión "estar limpio"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.2.

- **DT:** Profundidad Máxima = 2, Criterion: 'gini'.
- **AB:** Criterion (Estimador base): 'entropy', Splitter(Estimador base): 'random', Tasa de aprendizaje = 0.5, Número de estimadores = 50.

Tabla 4.20: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular etiquetado manualmente de identificación de la expresión "estar limpio".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	90.91 %	90.91 %	90.91 %	90.48 %	95.00 %	91.59 %	±3.3 %
DT	CV(1,1)	90.91 %	90.91 %	90.91 %	90.48 %	95.00 %	91.59 %	±3.3 %
AB	CV(1,1)	90.91 %	90.91 %	90.91 %	90.48 %	95.00 %	91.59 %	±4.1 %

- **Expresión "dar la cola"**

  Para cada algoritmo los mejores parámetros encontrados para el conjunto de validación son:

  - **SVM:** Kernel: radial, C = 1, Gamma = 0.2.
  - **DT:** Profundidad Máxima = 3, Criterion: 'gini'.
  - **AB:** Criterion (Estimador base): 'gini', Splitter(Estimador base): 'best', Tasa de aprendizaje = 0.1, Número de estimadores = 100.

Tabla 4.21: Exactitud de cada segmento de prueba de la validación cruzada para el modelo particular etiquetado manualmente de identificación de la expresión "dar la cola".

Algoritmo	Método de extracción	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
SVM	CV(1,1)	75.00 %	78.26 %	77.27 %	77.27 %	77.27 %	76.99 %	±2.2 %
DT	CV(1,1)	79.17 %	86.96 %	81.82 %	77.27 %	81.82 %	81.82 %	±6.6 %
AB	CV(1,1)	79.17 %	91.30 %	81.82 %	72.73 %	68.18 %	78.76 %	±15.8 %

4.4.3.5. Resultados de las pruebas de validación cruzada

Las pruebas de validación cruzada con K=5 permitieron estimar la exactitud de los algoritmos sobre datos nuevos. El comportamiento de los algoritmos en la prueba

fue diferente para cada modelo:

- Para el primer modelo, en la Tabla 4.8 se observó un mejor desempeño para el algoritmo SVM frente a los otros dos, sin embargo, la diferencia es poco notable y en los tres algoritmos no se observó sobreajuste.
- Para el segundo modelo, se observó un mejor rendimiento para el algoritmo AB, frente a los otros dos. La exactitud es menor a la de los modelos particulares.
- El comportamiento de los modelos particulares durante la prueba fue bastante variado y resulta difícil escoger hiperparámetros que optimicen a todos los modelos, sin embargo, esto no fue el caso para los que fueron etiquetados manualmente, pues estos comparten la mejor configuración del algoritmo SVM. En el modelo para la expresión "sacar la piedra" se observó inconsistencia para encontrar datos nuevos, esto se debe a la mala calidad de las etiquetas, esto sucede cuando la expresión se encuentra en el mismo contexto para varias observaciones pero las etiquetas se contradicen. El algoritmo SVM presentó menos sobreajuste durante las pruebas de validación de los modelos particulares.

Tabla 4.22: Comparación de exactitudes para cada segmento de la misma expresión con distinto etiquetado.

Expresión	Algoritmo	Iteración K					Media	Desviación Estándar
		1	2	3	4	5		
estar limpio	SVM	68.18 %	72.73 %	68.18 %	76.19 %	70.00 %	71.03 %	±6.1 %
estar limpio (manual)	SVM	90.91 %	90.91 %	90.91 %	90.48 %	95.00 %	91.59 %	±3.3 %
dar la cola	SVM	85.71 %	84.62 %	84.62 %	84.62 %	84.62 %	84.85 %	±0.9 %
dar la cola (manual)	SVM	75.00 %	78.26 %	77.27 %	77.27 %	77.27 %	76.99 %	±2.2 %

- El último modelo explica el comportamiento distinto de algunas expresiones, al ser etiquetado manualmente las etiquetas se contradicen mucho menos y el modelo predice fácilmente datos nuevos. No obstante, esto no significa que el etiquetado manual provea una mayor exactitud, puesto que para la expresión "dar la cola" se obtuvo mejor exactitud con el etiquetado automático. En la Tabla 4.22 se observa como para la expresión "estar limpio" la exactitud varía más en las iteraciones para el conjunto de datos etiquetado con la metodología del capitulo 3.

Se ha definido la mejor configuración para cada algoritmo, el siguiente capítulo corresponde a la simulación de los modelos.

Capítulo 5

Evaluación y resultados

En este capítulo se realiza el entrenamiento final de los modelos, habiendo obtenido ya los requisitos necesarios para su construcción, se evalúa el rendimiento de cada clasificador versus las técnica de extracción de características y se compara con otros algoritmos mencionados en el capítulo 2.

5.1. Métricas para la evaluación de los modelos de clasificación

A continuación se presentan las métricas para la evaluación de rendimiento de los algoritmos. Se han utilizado los parámetros óptimos obtenidos en la prueba de validación cruzada para cada algoritmo, y se incluye el modelo de Bernoulli del algoritmo de Naive Bayes para fines comparativos. Además, se comparan los métodos de extracción de características mencionados en la sección 4:

5.1.1. Rendimiento de los modelos de clasificación para la caracterización de EVF

En la Tabla 5.1, se puede observar una mayor exactitud en el conjunto de prueba para el algoritmo SVM en relación a los otros tres, asimismo, las características que combinaban los métodos tradicionales sumados a los vectores de palabra permitieron

Tabla 5.1: Valores de exactitud del modelo para caracterización de EVF

Métodos de extracción	Algoritmos			
	NB	SVM	DT	AB
CV	69.41 %	70.75 %	68.97 %	64.86 %
TF-IDF	69.41 %	70.41 %	69.16 %	65.98 %
CV+WV	69.54 %	**75.12 %**	64.27 %	64.05 %
TF-IDF+WV	69.54 %	**75.46 %**	65.05 %	63.83 %
WV	65.23 %	**75.87 %**	63.74 %	69.19 %

un mejor rendimiento del algoritmo.

Tabla 5.2: Métricas para evaluar el rendimiento del modelo para caracterización de EVF

Configuración del modelo		Precisión	Recuperación	F1 Score
SVM	CV+WV	74.18 %	51.26 %	60.63 %
SVM	TF-IDF+WV	74.66 %	51.09 %	60.67 %
SVM	WV	71.97 %	57.07 %	63.66 %

En la Tabla 5.2 se encuentran las mejores métricas obtenidas para el modelo de caracterización, correspondientes al algoritmo SVM. Existe un balance entre los documentos relevantes y los recuperados ya que las medidas de precisión y recuperación no difieren demasiado.

5.1.2. Rendimiento de los modelos de clasificación para la identificación de EVFV (generalizado)

Al igual que para el modelo anterior, se presentan las métricas obtenidas:

En la Tabla 5.3, se puede observar una mayor exactitud en el conjunto de prueba para los algoritmos SVM y NB, nuevamente, las características que combinaban los métodos tradicionales sumados a los vectores de palabra se desempeñaron mejor dentro de esos algoritmos.

En la Tabla 5.4, se encuentran las métricas obtenidas a partir del conjunto de

Tabla 5.3: Valores de exactitud del modelo generalizado de identificación de EVFV

Métodos de extracción	Algoritmos			
	NB	SVM	DT	AB
CV	65.69 %	**66.34 %**	63.49 %	62.31 %
TF-IDF	65.69 %	64.23 %	62.63 %	63.81 %
CV+WV	65.73 %	**66.13 %**	56.22 %	62.38 %
TF-IDF+WV	65.73 %	64.46 %	58.47 %	62.89 %
WV	54.91 %	56.5 %	53.1 %	54.51 %

Tabla 5.4: Métricas para evaluar el rendimiento del modelo generalizado de identificación de EVFV. Basadas en máquinas de vectores de soporte (SVM)

Vectores de características	Precisión	Recuperación	F1 Score
CV	64.83 %	55.91 %	60.04 %
TF-IDF	78.6 %	28.75 %	42.1 %
CV+WV	64.18 %	56.83 %	60.28 %
TF-IDF + WV	74.16 %	32.89 %	45.57 %
WV	76.22 %	5.58 %	10.39 %

prueba para el algoritmo SVM, los vectores de características de conteo, y los mismos combinados con vectores de palabras de 50 dimensiones se desempeñaron mejor con este algoritmo.

Tabla 5.5: Métricas para evaluar el rendimiento del modelo generalizado de identificación de EVFV. Basadas en el modelo de Bernoulli de Naive Bayes (NB)

Vectores de características	Precisión	Recuperación	F1 Score
CV	68.24 %	45.17 %	54.36 %
TF-IDF	68.24 %	45.17 %	54.36 %
CV+WV	68.06 %	45.68 %	54.67 %
TF-IDF + WV	68.06 %	45.68 %	54.67 %
WV	50.31 %	24.76 %	33.18 %

En la Tabla 5.5, se encuentran las métricas obtenidas a partir del conjunto de prueba para el algoritmo NB, los vectores de características no cambiaron tanto los cálculos de las métricas.

Tabla 5.6: Métricas para evaluar el rendimiento del modelo generalizado de identificación de EVFV. Basadas en un árbol de decisión binario (DT)

Vectores de características	Precisión	Recuperación	F1 Score
CV	69.49 %	33.55 %	45.26 %
TF-IDF	66.41 %	35.81 %	46.53 %
CV + WV	52.93 %	49.41 %	51.11 %
TF-IDF + WV	48.95 %	48.95 %	48.95 %
WV	50.31 %	24.76 %	33.18 %

En la Tabla 5.6, se encuentran las métricas obtenidas a partir del conjunto de prueba para el algoritmo DT, este algoritmo arrojó resultados muy pobres considerando que se calculó apenas un 51.11 % como máximo porcentaje de f1 score.

Tabla 5.7: Métricas para evaluar el rendimiento del modelo generalizado de identificación de EVFV. Basadas en impulso adaptativo (AB) con árbol de decisión como estimador base

Vectores de características	Precisión	Recuperación	F1 Score
CV	60.95 %	52.69 %	56.52 %
TF-IDF	64.77 %	45.88 %	53.71 %
CV + WV	61.51 %	49.05 %	54.58 %
TF-IDF + WV	59.05 %	48.54 %	53.28 %
WV	48.38 %	42.1 %	45.02 %

En la Tabla 5.7, se encuentran las métricas obtenidas a partir del conjunto de prueba para el algoritmo AB, este algoritmo mejoró el rendimiento del árbol de decisión, como se esperaba, y su mejor desempeño se registró con el uso de vector de conteo.

5.1.3. Rendimiento de los modelos de clasificación para la identificación de EVFV (particularizado)

Para los modelos particulares, se presentan las mejores métricas obtenidas y la configuración óptima del modelo:

Tabla 5.8: Valores de exactitud del modelo particularizado de identificación de EVFV

Expresión	Mejores vectores de características	Algoritmos			
		NB	SVM	DT	AB
parar bola	CV + WV	59.07 %	64.48 %	65.25 %	62.93 %
meterse a	CV + WV	69.66 %	70.22 %	70.22 %	66.29 %
caer encima	TF-IDF + WV	60.95 %	58.1 %	57.14 %	60.0 %
mentar la madre	TF-IDF + WV	73.63 %	72.53 %	56.04 %	58.24 %
estar hecho	TF-IDF	64.29 %	58.33 %	64.29 %	55.95 %
dar el golpe	TF-IDF + WV	76.06 %	76.06 %	80.28 %	63.38 %
sacar la piedra	CV + WV	45.83 %	70.83 %	52.08 %	54.17 %
estar limpio	CV + WV	68.09 %	65.96 %	55.32 %	61.7 %
dar la cola	CV + WV	86.21 %	89.66 %	58.62 %	79.31 %
poner la torta	TF-IDF	85.71 %	85.71 %	85.71 %	80.95 %
Promedios		68,95 %	71,19 %	64,5 %	64,29 %

En la Tabla 5.8, se puede observar una mayor exactitud promedio en el conjunto de prueba para los algoritmos SVM y NB. Los vectores de características híbridos predominan en la selección de métodos de extracción, la mayor exactitud obtenida fue de 89.66 % para el modelo de la expresión “dar la cola”, con el algoritmo SVM, mientras que la minima obtenida fué de 45.83 % para el modelo de la expresión “sacar la piedra”, con el algoritmo de Bayes. Esta ultima expresión, al igual que en la validación cruzada, registró los peores resultados.

En la Tabla 5.9, se encuentran las mejores métricas para cada modelo particular, y la configuración del modelo utilizada. El algoritmo DT, predominó entre los que obtuvieron mejor rendimiento. Se observó un porcentaje de recuperación promedio bastante bajo en comparación al encontrado en el modelo generalizado. Esto se debe a

Tabla 5.9: Métricas para evaluar el rendimiento del modelo particularizado de identificación de EVFV

Expresión	Configuración del modelo		Precisión	Recuperación	F1 Score
parar bola	DT	CV+WV	64.46 %	77.54 %	70.39 %
meterse a	DT	CV+WV	50.0 %	3.77 %	7.02 %
caer encima	NB	CV+WV	50.0 %	11.43 %	18.6 %
mentar la madre	NB	TF-IDF+WV	100.0 %	2.86 %	5.56 %
estar hecho	DT	TF-IDF	71.43 %	12.2 %	20.83 %
dar el golpe	DT	TF-IDF + WV	100.0 %	4.76 %	9.09 %
sacar la piedra	SVM	CV+WV	100.0 %	8.7 %	16.0 %
estar limpio	AB	TF-IDF	50.0 %	35.29 %	41.38 %
dar la cola	SVM	CV + WV	82.14 %	95.83 %	88.46 %
poner la torta	DT	TF-IDF	94.74 %	100.0 %	97.3 %
Promedios			76,28 %	35,24 %	37,46 %

el desbalance de los conjuntos de datos particulares.

5.1.4. Rendimiento de los modelos de clasificación para la identificación de EVFV (particularizado, etiquetado manualmente)

Para los modelos particulares, se presentan las mejores métricas obtenidas y la configuración óptima del modelo:

En la Tabla 5.10 se encuentran los valores de exactitud máximos encontrados para cada algoritmo por expresión etiquetada manualmente.

En la Tabla 5.11, se encuentran las mejores métricas para cada modelo particular, y la configuración del modelo utilizada.

Tabla 5.10: Valores de exactitud del modelo particularizado de identificación de EVFV a partir de un conjunto de datos etiquetado manualmente

Expresión	Mejores vectores de características	Algoritmos			
		NB	SVM	DT	AB
caer encima	TF-IDF + WV	72.45 %	73.21 %	74.34 %	60.0 %
estar limpio	WV	87.23 %	87.23 %	74.47 %	85.11 %
dar la cola	TF-IDF + WV	73.47 %	73.47 %	75.51 %	71.43 %

Tabla 5.11: Métricas para evaluar el rendimiento del modelo particularizado de identificación de EVFV para el conjunto de datos etiquetado manualmente

Expresión	Configuración del modelo		Precisión	Recuperación	F1 Score
caer encima	DT	TF-IDF + WV	74.8 %	97.94 %	84.82 %
estar limpio	AB	WV	20.0 %	50.0 %	28.57 %
dar la cola	DT	TF-IDF + WV	78.95 %	83.33 %	81.08 %

5.2. Resultados del cálculo de métricas

Las pruebas basadas en cálculo de métricas permitieron estimar el costo de los falsos negativos y falsos positivos en la clasificación de cada uno de los modelos. En estas evaluaciones se consiguieron los siguientes resultados:

- En el modelo de caracterización de EVF, el algoritmo de máquinas de vectores de soporte (SVM) registró el mejor rendimiento con un aumento de la exactitud de más del 5 % con respecto a los demás algoritmos. Además, los vectores de palabras fueron suficientes para obtener un resultado alto, aunque la métrica de precisión fue mayor cuando se utilizaba el vector de características híbrido.

- En el modelo de identificación de EVFV, el algoritmo de máquinas de vectores de soporte (SVM) registró el mejor rendimiento de entre todos los algoritmos, como se puede observar en la Tabla 5.12, el algoritmo AB mejoró los resultados del algoritmo DT, aún asi, la mejor opción para modelo de clasificación sigue siendo SVM, utilizando vectores de características híbridos.

Tabla 5.12: Mejores resultados para el cálculo de métricas del modelo generalizado

Algoritmo	Vector de Características	Precisión	Recuperación	F1 Score
NB	TF-IDF + WV	68.06 %	45.68 %	54.67 %
SVM	**CV+WV**	**64.18 %**	**56.83 %**	**60.28 %**
DT	TF-IDF + WV	48.95 %	48.95 %	48.95 %
AB	CV	60.95 %	52.69 %	56.52 %

- El modelo particularizado consiguió una mejor precisión que el modelo generalizado pero la recuperación fue mucho mas baja, esto se debe a que se requieren mucho más datos para obtener buen resultado al dividir el problema en pequeños modelos.

Tabla 5.13: Comparación de las métricas de los modelos particulares para la misma expresión.

Expresión	True	False	Precisión	Recuperación	F1 Score
caer encima	38 %	62 %	50.0 %	11.43 %	18.6 %
caer encima (et. manual)	74 %	26 %	74.8 %	97.94 %	84.82 %
estar limpio	32 %	68 %	50.0 %	35.29 %	41.38 %
estar limpio (et. manual)	12 %	88 %	20.0 %	50.0 %	28.57 %
dar la cola	83 %	17 %	82.14 %	95.83 %	88.46 %
dar la cola (et. manual)	78 %	22 %	78.95 %	83.33 %	81.08 %

En la Tabla 5.13, se comparan los modelos para las expresiones etiquetadas manualmente, se pueden observar tres escenarios:

- Para la expresión “caer encima”, cuyas etiquetas están invertidas, se encontró un aumento considerable de la métrica de recuperación, esto se debe a que el conjunto etiquetado manualmente posee mas observaciones.
- Para la expresión “estar limpio”, también hubo un aumento de la recuperación, aunque las observaciones fueron las mismas, pero las etiquetas están mas desbalanceadas en el etiquetado manual, lo que bajó la precisión

del modelo.

- Para la expresión “dar la cola”, la cual está similarmente etiquetada para los dos modelos, hubo una diferencia menor en el cálculo de métricas con respecto a las otras dos expresiones.

5.3. Simulación de los modelos en un texto informal.

Para probar los modelos finales se ha utilizado un texto venezolano rico en expresiones verbales fijas. [1]

- Cargar modelos

Figura 5.1: Documento normalizado

```
 'suponiamos muchos que serto algun dia se componia ser decir le empezari
amos a ver el qusero a la tostada solucionandose de verdad los problemas
socioeconomicos del pueblo y de las comunidadser por fin se construiran v
iviendas trabajo digno sin tener que jalar bolas  tener fe que la razon s
er poderosa y con sera fe avanzar hasta el fin hacer la parte que nos toc
ar seguir siempre la verdad'
```


- Cargar texto, en la Figura 5.1 se puede observar el primer párrafo (documento) del texto informal normalizado y con los verbos lematizados.
- Extracción de candidatos y predicción con el modelo de EVF. En la Figura 5.2, se pueden observar dos expresiones de la base de conocimientos “Jalar bolas” y “Ver el queso a la tostada”. El 1 en la columna predicción indica que la expresión sí es una expresión verbal fija. Ya que existen expresiones verbales fijas se prosigue al siguiente paso.
- Predicción de EVFV: En la Figura 5.3 se pueden observar las predicciones para los primeros tres docuementos del texto con el modelo de EVFV.

[1] Puede ser encontrado en http://www.opinionynoticias.com/opinionpolitica/5318-iesto-es-una-revolucion

Figura 5.2: Predicción de EVF

	pseudo_exp	prediccion
0	construiran viviendas	0
1	jalar bolas	1
2	tener fe	0
3	ver el qusero	1
4	hacer la parte	1
5	ver el qusero a la tostada	1
6	avanzar hasta el fin	1
7	seguir siempre	0
8	suponiamos muchos	0
9	decir le	0
10	ser poderosa	0
11	ser decir	0
12	tocar seguir	0

Figura 5.3: Predicción de EVF

	documento	prediccion
0	suponiamos muchos que serto algun dia se compo...	1
1	abraham lincoln... me disculpar ciertos camarad...	1
2	dar mucho dolor ver como se mofar en la cara d...	0

Capítulo 6

Conclusiones

En este trabajo de investigación se tocaron tres áreas relacionadas: la minería de textos en documentos y en redes sociales, el procesamiento del lenguaje natural y la clasificación de textos utilizando aprendizaje supervisado, siendo este último parte del objetivo general. Con el fin de obtener una base de conocimientos de expresiones verbales fijas con la cual empezar, se utilizó como recurso lingüístico, el Diccionario de Venezolanismos, un documento de 1994 que posee las diferentes expresiones y modismos utilizadas en la historia de Venezuela, con toda su documentación y ejemplos, este diccionario, a pesar de su antigüedad, tuvo lugar en la obtención de datos recientes, puesto a que fue posible hacer uso de la información extraída para realizar minería de texto a través de búsquedas. Twitter fué la red social utilizada para la minería de texto, debido a su naturaleza personal, ya que las expresiones verbales fijas son utilizadas en entornos coloquiales y Twitter ofrece ese ambiente informal. Además, es una de las redes sociales que ofrece más datos a través de texto ya que su premisa es la de escribir una idea en menos de 280 caracteres.

Las expresiones verbales fijas se caracterizan por su naturaleza regional, y es por esto que este trabajo se centra en las expresiones originales de Venezuela, por lo que la geolocalización constituyó un punto importante al momento de escoger Twitter como la fuente de datos principal. Esto generó ciertos problemas al decidir a que región pertenece la data, puesto que apenas el 2 % de la información de Twitter se encuentra precisamente localizada, por ello se utilizaron otras técnicas de localización como la

información que el usuario ofrecía o las palabras claves que indicaban la procedencia de la información. De hecho, a pesar de que se obtuvieron más de 30000 Tweets durante la extracción de los datos, solo la mitad fue relevante para la investigación.

Fue necesario procesar los datos en crudo, esto implicó la eliminación de elementos propios de Twitter que no se necesitaban durante el desarrollo, como los hashtags o las menciones, asi como también la limpieza de cada texto de emojis, signos de puntuación o caracteres unicode; y la utilización de técnicas de procesamiento como la lematización. Para hacer uso del corpus textual se propuso dividir el problema en dos partes: la caracterización de una expresión verbal fija, y la clasificación de la misma como expresión venezolana. Por ello fue necesario también dividir el conjunto de datos en dos conjuntos para distintos usos. Un cuerpo de textos constituidos por patrones de expresiones verbales que servirían para diferenciar si clasifican como expresion verbal fija o no, creado a partir de la información obtenida en Twitter y otro conjunto de datos constituido por textos que incluían las expresiones de la base de conocimiento dentro de un contexto.

El etiquetado de los corpus textuales siguió metodologías distintas. Para el corpus de expresiones verbales fijas se utilizó una combinación de similitud entre vectores de palabras y similitudes de cadena con respecto a cada expresión con el fin de encontrar similitudes semánticos y sintácticos. Estos que poseían un promedio de similitud mayor al 90 % calificarían como expresión verbal fija. Para el corpus de expresiones verbales fijas dentro de un contexto, o sea, el corpus de Tweets, se utilizó la geolocalización como se mencionó primero. Estos dos corpus etiquetados constituirían el paso inicial para la clasificación de textos utilizando aprendizaje supervisado.

Los algoritmos utilizados para la clasificación de textos fueron escogidos basándose en su frecuente utilización en problemas relacionados como la detección de expresiones multipalabra, análisis semántico de documentos e identificación de secuencias verbales en otros idiomas en otras investigaciones, estos fueron los algoritmos de Naive Bayes, las maquinas de vectores de soporte, los arboles de decisión binarios y el aprendizaje combinado del impulso adaptativo. Para cada algoritmo se propusieron 2 modelos: el de caracterízación de expresiones verbales fijas y el identificación de expresiones verbales fijas venezolanas, este último se estudió mas a fondo probando que

la separación del modelo general que incluye todas las expresiones en pequeños modelos individuales por expresión podría mejorar el rendimiento de la clasificación si se tienen suficientes datos, aunque empeorarlo si las etiquetas están muy desbalanceadas.

Otro enfoque de esta investigación fue la combinación de la sintaxis y la semántica al momento de extraer las características de los textos. Se probó que un modelo entrenado utilizando las cualidades combinadas de las técnicas de conteo y frecuencia junto a la de vectores de palabras de 50 dimensiones pueden lograr hasta un aumento del 4 % en la exactitud de la clasificación. Para el modelo final los resultados muestran que el algoritmo indicado para el problema de identificación de expresiones verbales fijas venezolanas y la caracterización de expresiones fueron las máquinas de vectores de soporte. A pesar de haber utilizado el impulso adaptativo para mejorar el desempeño de los árboles de decisión, las SVM se mantuvieron arriba en el cálculo de métricas durante las evaluaciones de rendimiento. Por otra parte, para los modelos para cada expresión el algoritmo DT consiguió mejores resultados, sin la ayuda de el AB ya que este se sobreajustaba.

Al obtener los modelos finales se realizó una simulación probando el modelo con un texto informal extraído de una página venezolana, rico en expresiones. Se observó que los modelos lograron, con una exactitud del 75.87 %, caracterizar las EVF encontradas; y con una exactitud del 66.34 %, identificar dichas expresiones como expresiones verbales venezolanas. Para la identificación de EVFV los modelos particulares obtuvieron un máximo de exactitud de 89.66 % para la expresión “dar la cola” y una exactitud promedio de 71.19 % para las 10 expresiones frecuentes utilizando máquinas de vectores de soporte y vectores de características híbridos. Debido a la escasez de datos para los modelos particulares solo se obtuvo un promedio de 35.24 % de recuperación versus 56.83 % de recuperación del modelo general. Se ha concluido que es posible conseguir mejores resultados para la identificación individual de las expresiones, siempre y cuando existan suficientes datos, de lo contrario, el modelo general es mejor.

Las expresiones son un instrumento utilizado en el aprendizaje de estudiantes de idiomas de una lengua extranjera, y su traducción es uno de los desafíos mas grandes dentro de la traducción automática pues su cualidad semántica resulta difícil

de tratar para una inteligencia artificial. Esta investigación propuso una metodología para identificar solo una parte de estas expresiones, aquellas cuyo núcleo es el verbo.

6.1. Aportes

Las principales contribuciones de esta investigación son las siguientes:

- Se utilizaron de técnicas de minería de texto en redes sociales para la recopilación de ejemplares de expresiones verbales fijas regionales dentro de un contexto para su uso en el procesamiento del lenguaje natural.
- Se construyó un corpus de textos en español etiquetado con el uso de palabras claves y geolocalización con información extraída de Twitter.
- Se estudiaron de técnicas de extracción de características combinadas para mejorar el rendimiento de un modelo supervisado en la identificación de expresiones verbales fijas regionales (venezolanas) dentro de un contexto.
- Se propuso una metodología para la identificación de expresiones verbales fijas venezolanas, esto permitirá resolver problemas aplicados a la enseñanza del español como lengua extranjera mediante inteligencia artificial, ya que las expresiones verbales fijas constituyen un reto al momento de ser traducidas.

6.2. Recomendaciones

A continuación se presentan algunas recomendaciones en la identificación de expresiones verbales fijas venezolanas utilizando aprendizaje automático.

- El Diccionario de Venezolanismos, a pesar de poseer expresiones que son utilizadas hoy día, sigue siendo un documento bastante desactualizado, por lo tanto, existen muchas expresiones verbales fijas que se usan en la actualidad que no se encuentran en la base de conocimientos. Una fuente de información más actualizada permitiría expandir la base de conocimientos de EVF.

- Cuando se extrae información de Twitter por medio de búsquedas, las consultas no siempre arrojan los resultados deseados, por ende, una fuente alterna de información resolvería ese problema. Ya que debido a esto no todas las expresiones de la base de conocimiento fueron encontradas en Twitter y algunas arrojaron muy pocos resultados.

- Existen otros algoritmos de aprendizaje supervisado utilizados en la clasificación de textos como la regresión logística y las redes neuronales. Esta investigación solo se centró en tres algoritmos, pero se debe ahondar en el uso de otras técnicas para verificar si su rendimiento es mayor.

- Los vectores de palabras utilizados correspondientes al modelo en español de SpaCy solo tienen 50 dimensiones. Existen modelos con más dimensiones, y estos podrían aumentar el rendimiento de los clasificadores considerablemente.

- Algunos modelos particularizados demostraron un mayor indice de exactitud con respecto al modelo generalizado. Con la información suficiente un modelo por expresión podría superar a un modelo generalizado ya que las expresiones son naturalmente diferentes entre si y se encuentran en distintos marcos.

6.3. Trabajos Futuros

A continuación se presentan los trabajos futuros que surgieron a raíz de esta investigación.

- Crear un diccionario bilingüe para las expresiones verbales fijas para los dos casos: cuando es una expresión literal o el significado es distinto al de Venezuela y cuando el significado es el que tiene en la región.

- Expandir el conjunto de datos con el uso de recursos lingüisticos alternativos y balancear las etiquetas.

- Usar la metodología propuesta para identificar expresiones verbales de una región diferente.

- Identificar las expresiones fijas con distinto núcleo: adjetivos, adverbios, sustantivos.
- Aplicar la identificación de expresiones dentro de un traductor real.
- Evaluar modelos entrenados con distintos algoritmos de clasificación.

Apéndice A

A.1. Acrónimos y abreviaciones

- NB: Naive Bayes
- SVM: Máquinas de vectores de soporte
- DT: Árbol de decisión binario
- AB: Impulso adaptativo
- CV: CountVectorizer o Vector de conteo
- TF-IDF: Frecuencia de términos × Frecuencia inversa de documento
- WV: Vector de palabras
- EVF: Expresion(es) Verbal(es) Fija(s)
- EVFV: EVF venezolanas
- DOC: Documento
- NLP: Procesamiento del Lenguaje Natural

A.1.1. Categorías gramáticales o tipo de palabra

- NOUN: Sustantivo. *Ej. perro, casa, persona*
- VERB: Verbo *(Ej. dar, tomar, abrir)*

- ADP: Adposición (preposición, postposición). *(Ej. desde, hacia, de)*
- DET: Artículo. *(Ej. el, los, un, unos)*
- ADJ: Adjetivo. *(Ej. rojo, lento, grande)*
- PROPN: Nombre propio *(Ej. David, Mérida)*
- ADV: Adverbio *(Ej: bien, bastante)*
- AUX: Verbo auxiliar *(ser, estar, haber, ir)*
- SCONJ: Conjunción subordinante *(Ej: que, cuantos)*
- PRON: Pronombre *(Ej: yo, tu, vos)*
- CONJ: Conjunción *(Ej: y, o, de)*

A.2. Algoritmos más importantes

A.2.1. Geolocalización de un Tweet

```
def get_location (tweet):
   try:
      if tweet.coordinates:
         lon = tweet.coordinates['coordinates'][0]
         lat = tweet.coordinates['coordinates'][1]
         location = str(lat) + ',' + str(lon)
      else:
         if tweet.place:
            location = tweet.place.country
         else:
            location = tweet.user.location
   except:
      if tweet.place:
         location = tweet.place.country
```

```
      else:
         location = tweet.user.location
   return location
```

A.2.2. Similitud Spacy

```
def max_similitud (pseudo_exp, lista_exp):
   if pseudo_exp in lista_exp:
      return [pseudo_exp, 1]
   doc = nlp(pseudo_exp)
   similitudes = [[expresion, doc.similarity(nlp(expresion))] for expresion
       in lista_exp]
   return sorted(similitudes, key=lambda sublist: sublist[1],
       reverse=True)[0]

#Candidato a expresion, lista de expresiones
def similitud_spacy(pseudo_exp, lista_exp):
   max_sim = max_similitud(pseudo_exp,lista_exp)
   similitud = max_sim[1]
   if max_sim[1]==1:
      return similitud, -1
   doc1 = nlp(pseudo_exp)
   doc2 = nlp(max_sim[0])
   #Búsqueda del más similar
   if abs(len(doc1)-len(doc2)) == 1:
      similitud-= similitud*0.05
   if abs(len(doc1)-len(doc2)) > 1:
      similitud-= similitud*0.1
   #Premio por tamaño igual
   if len(doc1)==len(doc2):
      similitud+= similitud*0.01
```

```
   similar_kernel = doc1[0].similarity(doc2[0])
   if(similar_kernel>0.95):
      similitud+= similitud*0.1
   return similitud, doc2.text
```

A.2.3. Extracción de candidatos a EVF de un documento

```
def remove_signos(cadena):
   import re
   import string
   signos= "\"¿¡\""
   for s in signos:
      cadena = cadena.replace(s,"")
   cadena = cadena.replace("&lt","").replace("&gt","")
   return re.sub('['+string.punctuation+']', '', cadena)



#Convierte la cadena entera en dupla de categoria gramtical,palabra
def dupla_list (sentence):
   tokesen = nlp(sentence)
   sentencex = [[token.pos_,token.text] for token in tokesen]
   return sentencex

#Extrae las sublistas, de acuerdo a subpatrones de la lista de duplas
def extract_sublist (candidato, etiquetas):
   resultado = []
   main_result = []
   #Para cada patron..
   for pattern in etiquetas:
      for i in range(len(candidato)):
      #Conseguir dupla de patron
      patron_i = candidato[i:i+len(pattern)]
      #Patron en forma de categorias gramaticales
```

```
      patron_f = [item[0] for item in patron_i]
#Si la primera palabra del candidato es la misma que la primera del patron
# Y el patron en forma de categorias es igual al patron
      if candidato[i][0] == pattern[0] and patron_f == pattern:
         #La expresion será la dupla
         expresion = candidato[i:i+len(pattern)]
      #La expresion final contiene solo las palabras
         expresionf = [item[1] for item in expresion]
         resultado.append(expresionf)
      if resultado != []:
         main_result+=resultado
      resultado = []
   return main_result

#La cadena deja de ser dupla y se obtienen las expresiones en patrones
def lista_cadena(lista_lista):
   lista_cadenas = []
   for lista in lista_lista:
      lista_cadenas.append(" ".join(lista))
   return lista_cadenas

def lemmatizar(cadena):
   doc = nlp(cadena)
   for token in doc:
      if token.pos_ == 'VERB' or token.pos_ == 'AUX':
         cadena = cadena.replace(token.text, token.lemma_)
   return cadena

def preparar_cadena(cadena, patrones):
   #cadena = lemmatizar(remove_signos(cadena))
   return lista_cadena(extract_sublist(dupla_list(cadena), patrones))
```

A.3. Especificaciones de Hardware

- Procesador: Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz 4 cores
- Tarjeta de Video: Intel(R) HD Graphics 4000
- RAM: 4.0 GB
- Sistema Operativo: Windows 10

A.4. Referencias

- Jupyter Notebook [71]
- Scikit-Learn [64]
- Spacy [72]
- Pattern [73]
- Tweepy [74]
- OpenCage Geocoder [75]
- Fuzzy Wuzzy [76]
- UMLet [77]

A.5. Diagramas de flujo

En esta sección se presenta el contenido adicional del proyecto que pudiese aclarar alguna duda, mejorar la perspectiva de los procesos o simplemente demostrar conceptos con muchísimo detalle.

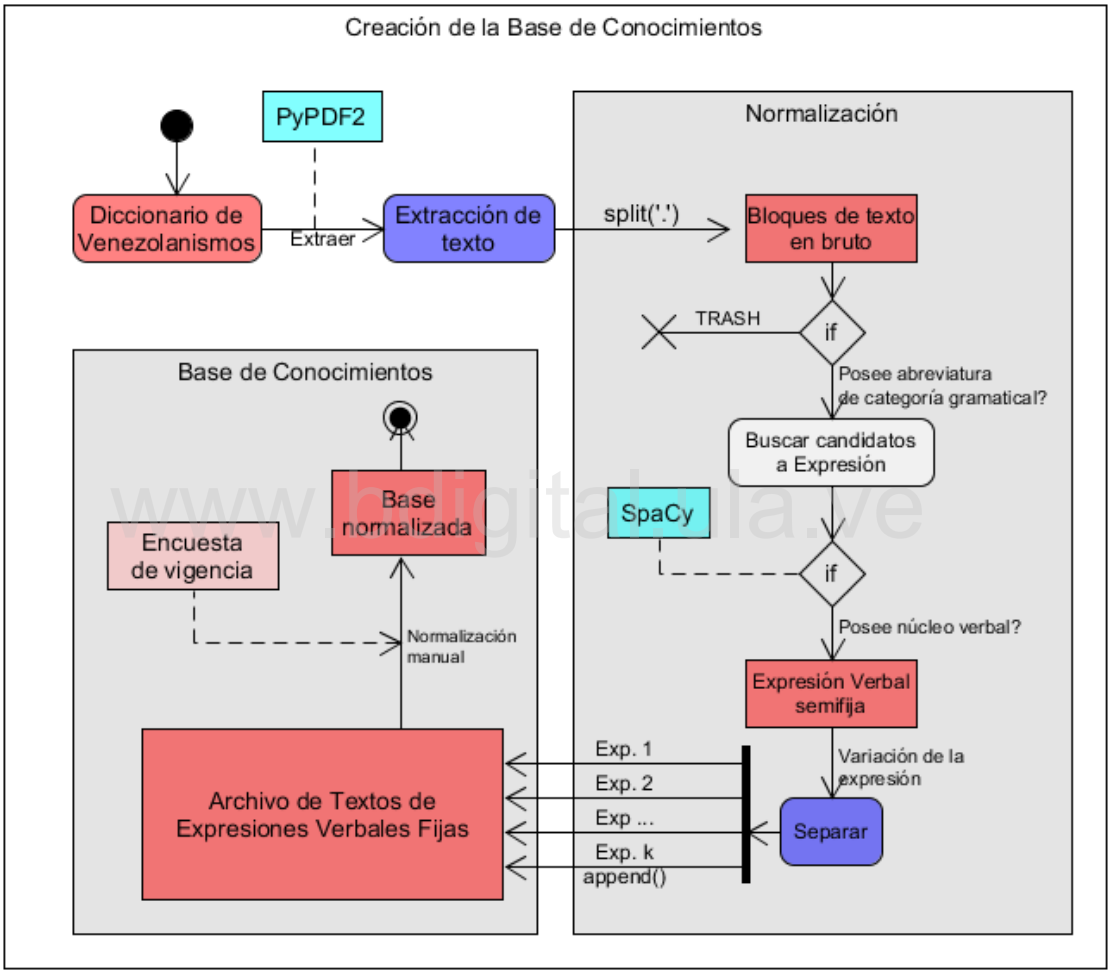


Figura A.1: Diagrama de flujo para la creación de la base de conocimientos de Expresiones Verbales Fijas

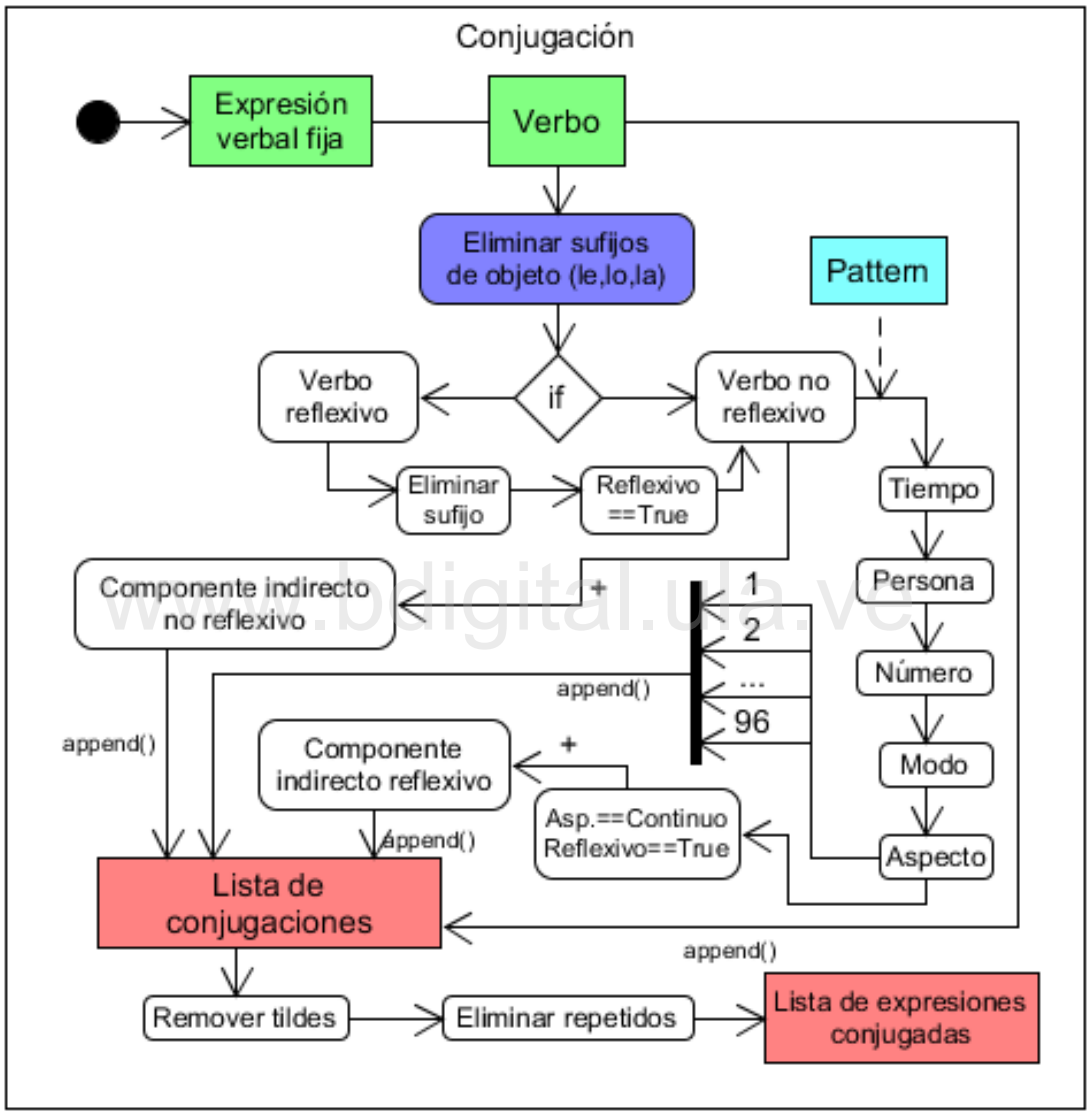


Figura A.2: Diagrama de flujo para la conjugación del verbo de las expresiones para la consulta

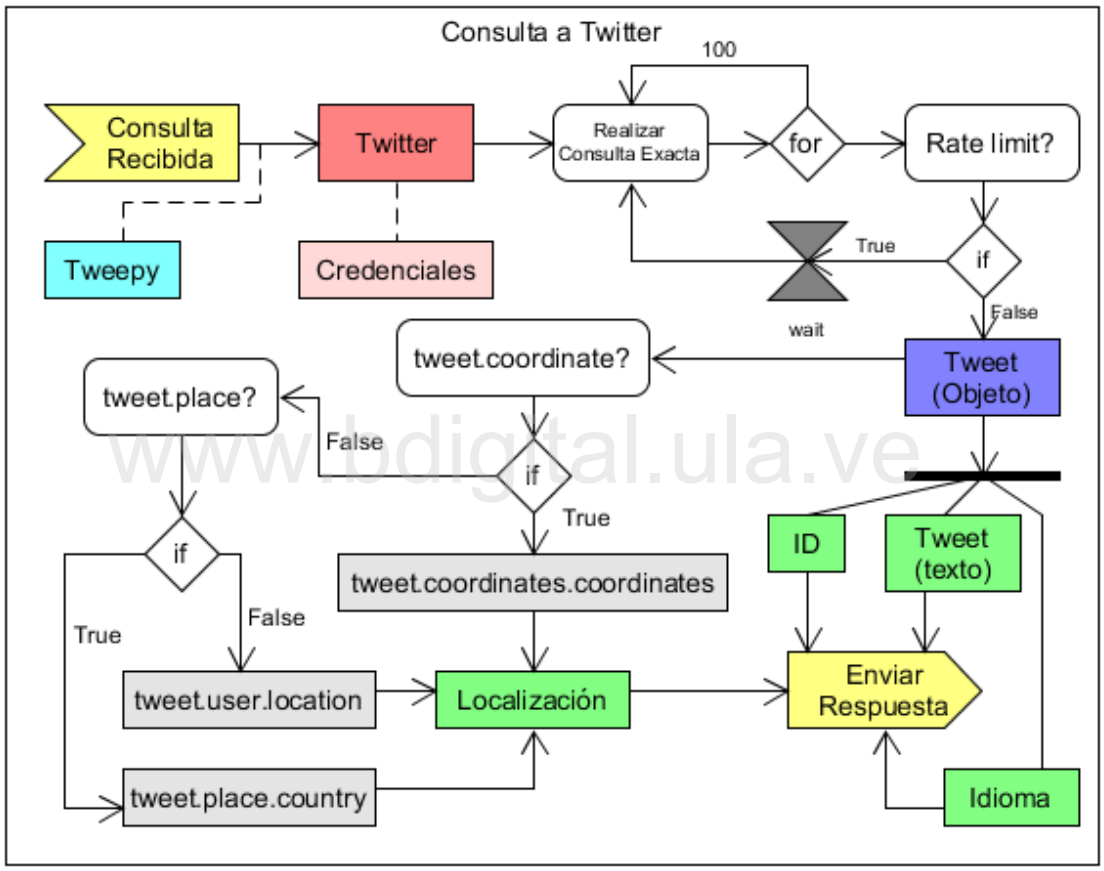


Figura A.3: Diagrama de flujo para la extracción de los datos de twitter utilizando Tweepy

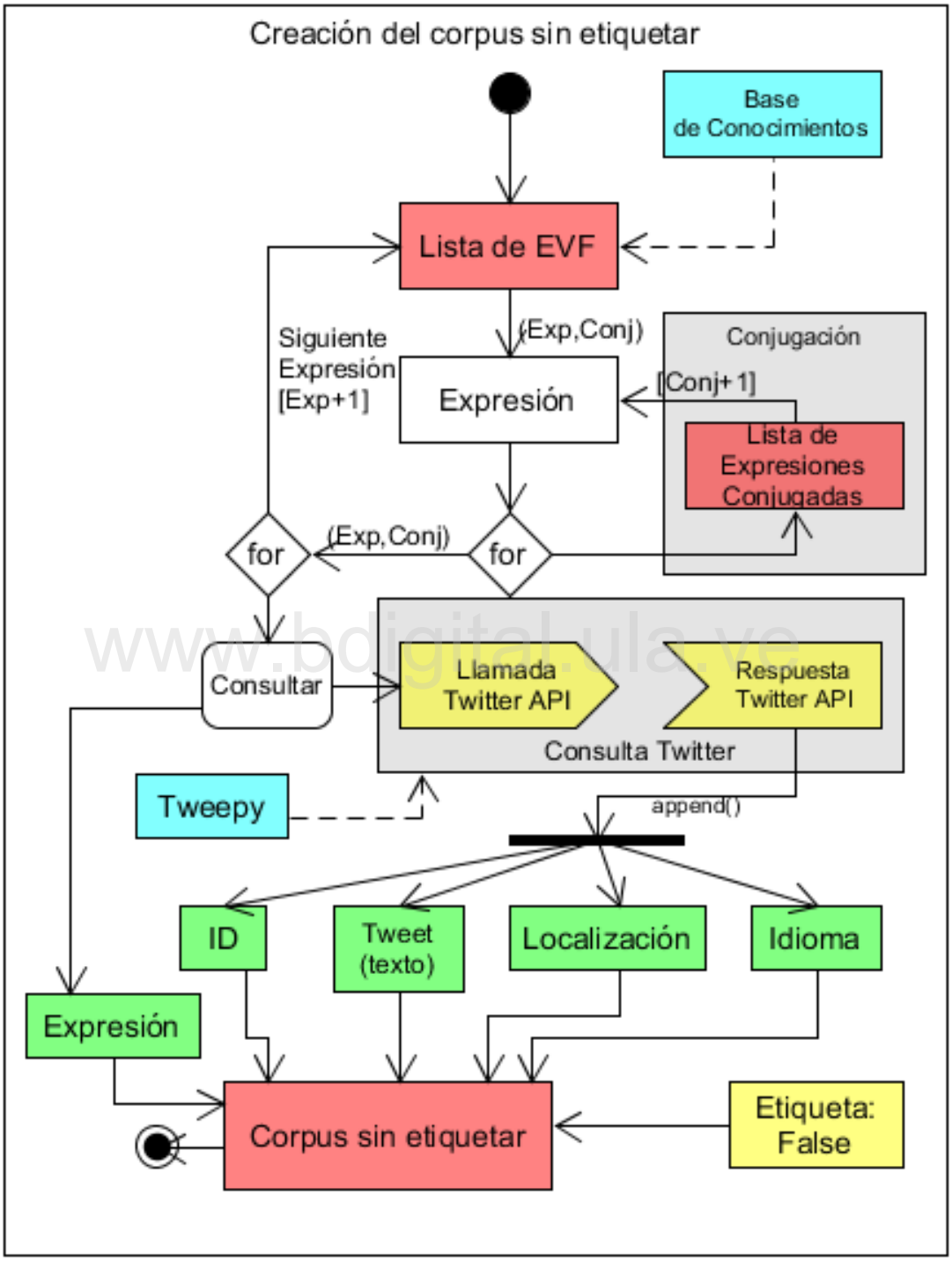


Figura A.4: Diagrama de flujo para la creación del corpus inicial

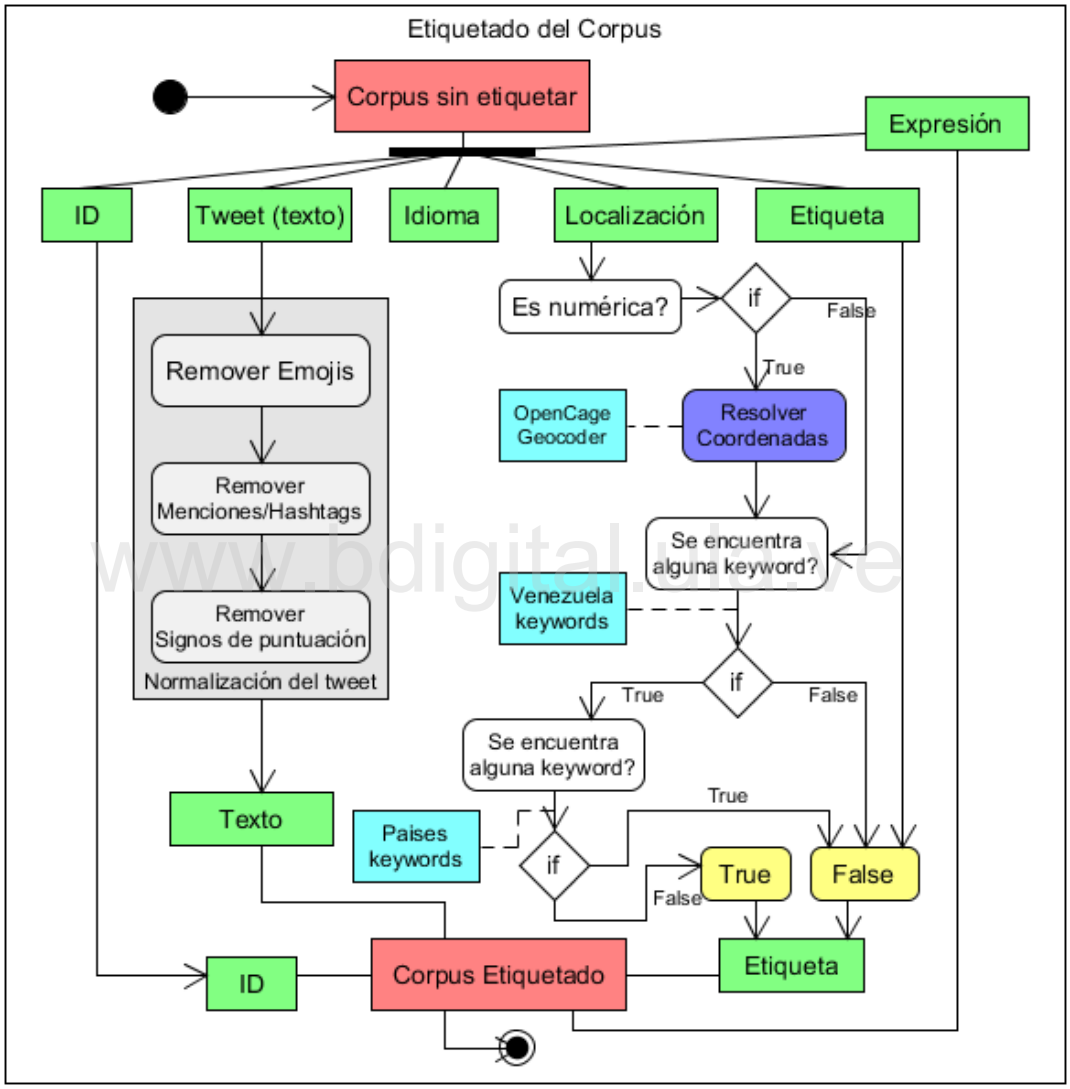


Figura A.5: Diagrama de flujo para el etiquetado del corpus

Bibliografía

[1] J. E. Aguilar Ruiz, "Diseño e integración de un módulo para detectar y categorizar opiniones de reclamo en un sistema de análisis web aplicado al rubro de las telecomunicaciones," 2017.

[2] E. K. Jacob, "Classification and categorization: a difference that makes a difference," 2004.

[3] R. A. Española, "Cuadrado. diccionario de la lengua española," 2019.

[4] S. Gutiérrez Ordóñez, "Sobre las categorías, las clases y la transposición," 1985.

[5] Á. C. Rodríguez, "La expresión fraseológica verbal en el discurso informativo del español actual," *Revista de Filología y Lingüística de la Universidad de Costa Rica*, vol. 43, no. 2, pp. 107–121, 2017.

[6] M. E. Maron, "Automatic indexing: an experimental inquiry," *Journal of the ACM (JACM)*, vol. 8, no. 3, pp. 404–417, 1961.

[7] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR?94*. Springer, 1994, pp. 3–12.

[8] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[9] Y. Yang, X. Liu *et al.*, "A re-examination of text categorization methods," in *Sigir*, vol. 99, no. 8, 1999, p. 99.

[10] T. L. Ngo-Ye, A. P. Sinha, and A. Sen, "Predicting the helpfulness of online reviews a scripts-enriched text regression model," *Expert Systems with Applications*, vol. 71, pp. 98–110, 2017.

[11] S. Bahassine, A. Madani, and M. Kissi, "An improved chi-sqaure feature selection for arabic text classification using decision tree," in *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*. IEEE, 2016, pp. 1–5.

[12] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[13] M. Goudjil, M. Koudil, M. Bedda, and N. Ghoggali, "A novel active learning method using svm for text classification," *International Journal of Automation and Computing*, vol. 15, no. 3, pp. 290–298, 2018.

[14] W. Gharbieh, V. Bhavsar, and P. Cook, "A word embedding approach to identifying verb-noun idiomatic combinations," in *Proceedings of the 12th Workshop on Multiword Expressions*, 2016, pp. 112–118.

[15] M. Farahmand and R. Martins, "A supervised model for extraction of multiword expressions, based on statistical context features," in *Proceedings of the 10th workshop on multiword expressions (MWE)*, 2014, pp. 10–16.

[16] B. P. Sánchez, D. Pinto, and S. Mejri, "Metodologıa para la identificación de secuencias verbales fijas," *Avances en la Ingeniería del Lenguaje y del Conocimiento*, p. 45, 2014.

[17] W. A. Koza, "Análisis automático de textos: Reconocimiento de incisos," *Revista Infosur. http://www. infosurrevista. com. ar/biblioteca/INFOSUR-Nro2-2008-Koza. pdf*, 2008.

[18] J. Tuberquia Benitez and J. A. Cartagena, "Diagnostico del estado de la cuestion y retos en el analisis de sentimientos basado en el procedimiento del lenguaje natural," 2018.

[19] J. Mariani, G. Francopoulo, P. Paroubek, and F. Vernier, "The nlp4nlp corpus (ii): 50 years of research in speech and language processing," *Frontiers in Research Metrics and Analytics*, vol. 3, p. 37, 2018.

[20] J. Vilares, "Aplicaciones del procesamiento del lenguaje natural en la recuperación de información en español." *Procesamiento del lenguaje natural*, vol. 36, 2006.

[21] A. Ferreira and G. Kotz, "Ele-tutor inteligente: Un analizador computacional para el tratamiento de errores gramaticales en español como lengua extranjera," *Revista signos*, vol. 43, no. 73, pp. 211–236, 2010.

[22] I. S. Paute Cárdenas, "Evaluación de las herramientas de minería de textos en los mensajes de la red social twitter," B.S. thesis, Universidad del Azuay, 2018.

[23] E. Olarte, M. D. Panizzi, and R. A. Bertone, "Segmentación de mercado usando técnicas de minería de datos en redes sociales," in *XXIV Congreso Argentino de Ciencias de la Computación (La Plata, 2018).*, 2018.

[24] P. Fornacciari, M. Mordonini, and M. Tomaiuolo, "Social network and sentiment analysis on twitter: Towards a combined approach." in *KDWeb*, 2015, pp. 53–64.

[25] G. V. Cormack and M. R. Grossman, "Scalability of continuous active learning for reliable high-recall text classification," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 1039–1048.

[26] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[27] I. N. Alousque, "La traducción de las expresiones idiomáticas marcadas culturalmente," *Revista de lingüística y lenguas aplicadas*, no. 5, pp. 133–140, 2010.

[28] J. Sevilla Muñoz and A. Gonzalez Rodriguez, "La traducción y la didáctica de las expresiones idiomáticas (francés-español)," *Equivalences*, vol. 24, no. 2, pp. 171–182, 1994.

[29] N. Ponce Márquez, "El arte de traducir expresiones idiomáticas: la finalidad de la funcionalidad," 2011.

[30] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, "A brief survey of text mining: Classification, clustering and extraction techniques," *arXiv preprint arXiv:1707.02919*, 2017.

[31] M. J. Tejera, *Diccionario de venezolanismos*. Academia Venezolana de la Lengua: Universidad Central de Venezuela, Facultad ?, 1983, vol. 2.

[32] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 562–570.

[33] B. P. Sánchez, "Análisis de la diversidad morfosintáctica en las locuciones verbales." *Research in Computing Science*, vol. 97, pp. 113–125, 2015.

[34] L. Williams, C. Bannister, M. Arribas-Ayllon, A. Preece, and I. Spasić, "The role of idioms in sentiment analysis," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7375–7385, 2015.

[35] F. Detry, "Retos iniciales en la comprensión y memorización de las expresiones idiomáticas en clase de ele," *marcoELE. Revista de Didáctica Español Lengua Extranjera*, no. 21, pp. 1–18, 2015.

[36] E. F. Vicente, "Sugerencias para mejorar el tratamiento de las expresiones idiomáticas en los diccionarios fraseológicos en inglés y español," *Trans. Revista de traductología*, no. 12, pp. 123–148, 2017.

[37] T. Baviera, "Técnicas para el análisis de sentimiento en twitter: Aprendizaje automático supervisado y sentistrength," *Revista Dígitos*, vol. 1, no. 3, pp. 33–50, 2017.

[38] J. Liu, "A research into the application of machine translation in the translation teaching," in *2017 International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2017)*. Atlantis Press, 2017.

[39] G. Salton, "Representations of idioms for natural language processing: Idiom type and token identification, language modelling and neural machine translation," 2017.

[40] Y.-J. Su, H.-W. Huang, and W.-C. Hu, "Using idiomatic expression for chinese sentiment analysis," in *2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)*. IEEE, 2017, pp. 1–4.

[41] B. Priego Sánchez and D. Pinto, "Identification of verbal phraseological units in mexican news stories," *Computación y Sistemas*, vol. 19, no. 4, pp. 713–720, 2015.

[42] M. Diale, C. Van Der Walt, T. Celik, and A. Modupe, "Feature selection and support vector machine hyper-parameter optimisation for spam detection," in *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*. IEEE, 2016, pp. 1–7.

[43] G. Salton, R. J. Ross, and J. Kelleher, "Idiom token classification using sentential distributed semantics," 2016.

[44] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[45] "Twitter overcounted active users since 2014, shares surge on profit hopes," 2019, [Online; accessed 29-November-2019]. [Online]. Available: https://www.usatoday.com/story/tech/news/2017/10/26/twitter-overcounted-active-users-since-2014-shares-surge/801968001/

[46] M. Jansche and M. E. Epstein, "Building language models for a user in a social network from linguistic information," Aug. 29 2017, uS Patent 9,747,895.

[47] A.-P. Bográn, J.-L. Alonso-Berrocal, and C. G. Figuerola, "Análisis léxico sobre los tweets de twitter," 2013.

[48] "Oauth 2.0," 2019, [Online; accessed 17-November-2019]. [Online]. Available: https://oauth.net/

[49] E. Blasco Ascencio, "Aplicación de técnicas de minería de datos en redes sociales/web," 2015.

[50] R. Lapeña, J. Font, Ó. Pastor, and C. Cetina, "Analyzing the impact of natural language processing over feature location in models," in *ACM SIGPLAN Notices*, vol. 52, no. 12. ACM, 2017, pp. 63–76.

[51] Y. Lu *et al.*, "Performing sentiment analysis on tweets: A comparison of machine learning algorithms across large data sets," Ph.D. dissertation, California State University Channel Islands, 2019.

[52] N. Hanafiah, A. Kevin, C. Sutanto, Y. Arifin, J. Hartanto *et al.*, "Text normalization algorithm on twitter in complaint category," *Procedia computer science*, vol. 116, pp. 20–26, 2017.

[53] J. Storby, "Information extraction from text recipes in a web format," 2016.

[54] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[55] A. Singhal *et al.*, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[56] A. G. Montes, A. L. H. Miralles, L. L. García, and M. L. Martínez, "Aplicación de las unidades fraseológicas a la enseñanza de e/le," *Foro de profesores de E/LE*, no. 13, p. 5, 2017.

[57] F. Núñez-Román, "Translating phraseology," *Translation and Translanguaging in Multilingual Contexts*, vol. 2, no. 1, pp. 106–123, 2016.

[58] G. Corpas Pastor, "Manual de fraseología," *Gredos, Madrid*, 1996.

[59] L. L. Toro and R. Luque, *Léxico español actual IV*. Cafoscarina, 2017.

[60] Á. Di Tullio, *Manual de gramática del español*. La isla de la luna Buenos Aires, 2005.

[61] J. Montilva, N. Arapé, and J. Colmenares, "Desarrollo de software basado en componentes," in *Actas del IV. Congreso de Automatización y Control. Mérida, Venezuela*, 2003.

[62] J. M. Kommers, D. Freed, and D. P. Kennedy, "Information retrieval from a collection of information objects tagged with hierarchical keywords," Apr. 11 2006, uS Patent 7,028,024.

[63] J.-S. Kim, D.-S. Jin, K.-Y. Kim, and H.-S. Choe, "Automatic in-text keyword tagging based on information retrieval," *Journal of Information Processing Systems*, vol. 5, no. 3, pp. 159–166, 2009.

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[65] A. Basarkar, "Document classification using machine learning," 2017.

[66] Tripathi, Mayank, "How to process textual data using tf-idf in python," https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/, 2018, [Online; accessed 6-October-2019].

[67] Y. Wu, W. Wu, C. Xu, and Z. Li, "Knowledge enhanced hybrid neural network for text matching," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[68] P. Bafna, D. Pramod, and A. Vaidya, "Document clustering: Tf-idf approach," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 61–66.

[69] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and word2vec for text classification with semantic features," in *2015 IEEE 14th International*

Conference on Cognitive Informatics & Cognitive Computing (ICCI CC)*. IEEE, 2015, pp. 136–140.

[70] Tom, "Corpora versus datasets," 2018, [Online; accessed 14-November-2019]. [Online]. Available: https://corpuslinguisticmethods.wordpress.com/2013/12/28/corpora-versus-datasets/

[71] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows." in *ELPUB*, 2016, pp. 87–90.

[72] A. Explosion, "spacy-industrial-strength natural language processing in python," *URL: https://spacy. io*, 2017.

[73] T. D. Smedt and W. Daelemans, "Pattern for python," *Journal of Machine Learning Research*, vol. 13, no. Jun, pp. 2063–2067, 2012.

[74] J. Roesslein, "Tweepy," *Python programming language module*, 2015.

[75] "Open cage geocoder," 2019, [Online; accessed 23-November-2019]. [Online]. Available: https://opencagedata.com/about

[76] A. Cohen, "Fuzzywuzzy: Fuzzy string matching in python," *ChairNerd Blog*, 2011.

[77] M. Auer, J. Poelz, A. Fuernweger, L. Meyer, and T. Tschurtschenthaler, "Umlet, uml tool for fast uml diagrams."