Universitat Oberta de Catalunya

uoc.edu

Análisis de sentimientos aplicado a la opinión política en Twitter: un sistema de clasificación en tiempo real

Pablo Llorente Ayuso

Grado en ingeniería informática

Inteligencia artificial

David Isern Alarcón

Carles Ventura Royo

4 de junio de 2019

A mis padres y hermanos.

A los profesores del IES Tetuán de las Victorias, por inculcarme el amor por la informática.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis de sentimientos aplicado a la opinión política en Twitter: un sistema de clasificación en tiempo real*
Nombre del autor:	*Pablo Llorente Ayuso*
Nombre del consultor/a:	*David Isern Alarcón*
Nombre del PRA:	*Carles Ventura Royo*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	*Grado en ingeniería informática*
Área del Trabajo Final:	*Inteligencia artificial*
Idioma del trabajo:	*Castellano*
Palabras clave	*Sentiment analysis, political opinion, Twitter*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

En este trabajo se ha desarrollado una aplicación web que muestra, en tiempo real, la opinión polarizada de los usuarios en español de Twitter sobre cuatro temas políticos: feminismo, derechos LGTB, migraciones y servicios públicos. La aplicación se inscribe dentro de la disciplina del análisis de sentimientos, obteniendo las opiniones por medio de la clasificación del texto de los tweets utilizando un conjunto de algoritmos de aprendizaje supervisado.

La preparación de los datos de entrenamiento se ha llevado a cabo implementando técnicas simbólicas de procesamiento del lenguaje natural mediante el uso de funciones con cadenas y expresiones regulares sobre el texto de los tweets. Para la selección de atributos se ha escogido la frecuencia inversa de documento sobre las palabras del texto. La clasificación final de una opinión resulta en la selección de la clase mayoritaria del conjunto de clasificadores.

Los resultados de la evaluación muestran una eficacia de entre el 77% y el 94% en función de la métrica seleccionada y del tema político estudiado. El algoritmo con mejores resultados es el clasificador bayesiano, si bien su eficacia converge con el clasificador por mayoría a medida que la distribución de las clases del conjunto de entrenamiento se hace más uniforme.

La arquitectura de la aplicación web tiene tres niveles (presentación, lógica de negocio e integración) siguiendo un patrón modelo-vista-controlador en un entorno WAMP (Windows, Apache, MariaDB y PHP) en el que los resultados de la clasificación se ofrecen en formato JSON mediante una API.

Abstract (in English, 250 words or less):

In this work we have developed a web application that shows, in real time, the

polarized opinion of users on Twitter in Spanish about four political issues: feminism, LGBT rights, migrations and public services. The application is part of Sentiment Analysis discipline and opinions are obtained using a set of supervised machine learning algorithms.

The preparation of the training data has been carried out by implementing symbolic natural language processing techniques through the use of functions with strings and regular expressions on the text of the tweets. For the selection of attributes, the inverse document frequency has been chosen over the words in the text. The final classification of an opinion results in the selection of the majority class of the set of classifiers.

The results of the evaluation show an effectiveness between 77% and 94% depending on the selected metric and the political issue studied. The algorithm with the best results is the Bayesian classifier, although its efficiency converges with the classifier by majority as the distribution of the classes of the training set becomes more uniform.

The architecture of the web application has three levels: presentation, business logic and integration, and it follows a model-view-controller pattern in a WAMP environment (Windows, Apache, MariaDB and PHP) in which the results of the classification are offered in JSON format using an API.

Índice

Lista de figuras y tablas

1. Introducción

1.1. Contexto y justificación del Trabajo

En la última década ha surgido un interés creciente por el estudio de las opiniones políticas utilizando los datos a gran escala que se producen en las aplicaciones de red social (Arcila-Calderón et. al, 2017). Sin embargo, la mayoría de los trabajos realizados en este ámbito se han basado en la clasificación manual de la opinión o en el análisis automático mediante el uso de diccionarios. Aquellos en los que se utilizan técnicas de machine learning son, de momento, escasos, pero es evidente que ofrecen un amplio abanico de oportunidades para crear soluciones eficaces y creativas.

Por ello, se desarrollará una aplicación web que muestre el estado de la opinión en tiempo real de los usuarios en español de la red de microblogging Twitter respecto a algunos de los clivajes políticos más importantes de la actualidad: feminismo, derechos LGTB, servicios públicos, migración o integración europea, entre otros. El término clivaje, en ciencia política, se refiere a los conflictos políticos que atraviesan una sociedad y que dividen en posiciones ideológicas opuestas a sus miembros[1].

El trabajo se inscribirá dentro del campo del análisis de sentimientos, disciplina de la inteligencia artificial entre cuyos objetivos se encuentra la elaboración de técnicas que permitan clasificar con eficacia expresiones del lenguaje natural en positivas, negativas o neutras respecto a un determinado tema, idea o sentimiento.

Estas expresiones del lenguaje natural serán los mensajes publicados por los usuarios en Twitter y, para su extracción, limpieza y selección de atributos, se utilizarán algunas técnicas de procesamiento del lenguaje natural.

Por último, para la clasificación de la valoración de los mensajes respecto a los clivajes, se utilizará un conjunto de algoritmos de aprendizaje supervisado. Esta técnica forma parte de la disciplina del aprendizaje computacional y consiste, en este caso, en la predicción del valor de la clase a la que pertenece un objeto después de haber entrenado al sistema con un conjunto de objetos cuya clase es conocida de antemano.

1.2. Objetivos del Trabajo

El objetivo principal es la obtención de la aplicación final que implemente el sistema de clasificación de mensajes en Twitter basado en el análisis de sentimientos utilizando técnicas de aprendizaje supervisado. Esta aplicación mostrará a través de su interfaz web el estado de la opinión sobre determinados temas políticos en tiempo real.

[1] El término clivaje fue acuñado por los politólogos Seymour Martin Lipset y Stein Rokkan en su obra de 1967 Party Systems and Voter Alignments.

Como objetivos específicos del desarrollo de la aplicación podemos destacar:

- Desarrollo del módulo principal de la aplicación.
- Obtención del conjunto de datos de entrenamiento a través de la utilización de la API de Twitter.
- Clasificación manual del conjunto de datos de entrenamiento.
- Selección de las técnicas de procesamiento del lenguaje natural para el tratamiento previo de los datos y la obtención del conjunto de características para la clasificación.
- Desarrollo del módulo de obtención de datos en tiempo real.
- Selección de las métricas de evaluación y selección de los algoritmos de aprendizaje supervisado que obtengan los mejores resultados.
- Desarrollo del módulo de clasificación de la aplicación.
- Desarrollo de la interfaz web de visualización de los datos.

De manera paralela, se deberá elaborar la memoria final de trabajo donde se expliquen tanto el desarrollo técnico de la aplicación como todas las tareas de gestión del proyecto. También deberá incluir una exposición teórica sobre las disciplinas en las que se basa el trabajo: análisis de sentimientos y procesamiento del lenguaje natural, explicando sus conceptos principales, el estado actual de la técnica, con sus posibles aplicaciones y limitaciones, y su futuro.

1.3. Enfoque y método seguido

En cuanto al enfoque y a la metodología de trabajo, se ha decidido el desarrollo completo de una aplicación enmarcada en el campo del análisis de sentimiento debido a que esta área es muy joven, (apenas dos décadas de historia), y existen escasos ejemplos de desarrollos que puedan servir de punto de partida. A pesar de ello, se han seguido como guías fundamentales los trabajos de (Sobrino Sande, 2018) y de (Lage García, 2014) por sus características similares, ya que son, respectivamente, un trabajo de fin de máster y un trabajo de fin de grado.

El método de trabajo seguido ha consistido, principalmente, en la búsqueda y lectura de artículos de investigación publicados en universidades y revistas científicas del campo para establecer los fundamentos teóricos sobre los que se asienta la aplicación y organizar su proceso de desarrollo.

La falta de ejemplos ha permitido decidir de manera muy abierta la arquitectura y la tecnología de la aplicación, por tanto, se ha optado por utilizar un entorno de desarrollo WAMP, es decir, una combinación de sistema Windows, servidor web Apache, base de datos MariaDB y el lenguaje de programación PHP.

Pese a que los recursos en PHP para el desarrollo de herramientas de procesamiento del lenguaje natural y de aprendizaje automático son escasos y poco conocidos, se ha optado por este lenguaje por razones de competencias, habilidades y experiencia del autor.

Se descartó Java por conocimientos insuficientes para generar un entorno web y se decidió lo mismo con Python por un conocimiento insuficiente del lenguaje. La decisión final de seleccionar PHP como lenguaje para la aplicación estuvo motivada por la existencia de la librería PHP-ML (que cuenta con todas las herramientas fundamentales de machine learning) y de otras librerías necesarias para algunas operaciones del procesamiento del lenguaje natural en el tratamiento previo de los datos.

Por último, se seleccionó Netbeans (versión 10) como entorno de desarrollo integrado. Para la gestión de las dependencias externas se utiliza el software Composer y el control de versiones se lleva a cabo con Git.

Respecto a la obtención de los datos de entrenamiento, se ha utilizado una cuenta de desarrollador en Twitter para la extracción de los 120 tweets en texto completo más recientes desde el día 10 de abril de 2019, excluyendo retweets, de 23 cuentas de usuario que incluyen a los principales líderes de los 5 partidos políticos de ámbito estatal, a los líderes en el Congreso del PNV y de ERC, a las alcaldesas de Madrid y Barcelona y a otras cuentas que suelen generar contenido político polarizado. Con estos parámetros se ha obtenido 1170 tweets, de los que se han clasificado de manera manual 425 para utilizarse como conjunto de entrenamiento.

En cuanto a los clivajes políticos escogidos para formar las clases de los tweets, se han seleccionado los siguientes: feminismo, derechos LGTB, migraciones y servicios públicos. Las clases pueden tener tres valores diferentes, que se representan con un número entero: un 1 indica que el tweet es favorable al clivaje, un 0 indica que es neutral y un -1 indica que es desfavorable.

A continuación, se detalla cada uno de los clivajes escogidos para la clasificación:

- **Feminismo:** un tweet es clasificado como favorable si expresa su acuerdo respecto a las ideas y acciones principales de los movimientos feministas.
- **Derechos LGTB:** un tweet es clasificado como favorable si expresa su acuerdo con el cumplimiento y/o ampliación de los derechos de las personas LGTB.
- **Migraciones:** un tweet es clasificado como favorable si expresa su desacuerdo con el endurecimiento de las políticas, normativas y acciones gubernamentales relativas a la inmigración, residencia, nacionalidad o acogida de extranjeros.
- **Servicios públicos:** un tweet es clasificado como favorable si expresa su acuerdo en la defensa de los servicios y empresas públicas frente a su privatización o degradación, o si lo expresa respecto a su extensión a otros ámbitos de la economía, y al aumento de su financiación y calidad.

Obviamente, los clivajes seleccionados son generalizaciones de temas complejos, de enorme diversidad y que se pueden subdividir en varios niveles

jerárquicos en función del detalle hasta el que se pretenda llegar, sin embargo, esto queda fuera del alcance de este trabajo.

1.4. Planificación del Trabajo

La planificación temporal del desarrollo del trabajo ha seguido los hitos marcados por el calendario académico establecido y dividido en la entrega de varias pruebas de evaluación continua a lo largo del semestre, desde el 20 de febrero de 2019 hasta el 4 de junio del mismo año.

Tabla 1. Contenido y fecha de entrega de las PEC

Contenidos de las PEC		
PEC	**Fecha de entrega**	**Contenido**
PEC 2	22/04/2019	• Código fuente del módulo de obtención de datos y del módulo de preprocesamiento. • Capítulos 2 y 5 de la memoria.
PEC 3	20/05/2019	• Código fuente de la aplicación completa. • Capítulo 6 de la memoria.
PEC 4	04/06/2019	• Memoria de trabajo completa.
PEC 5	12/06/2019	• Presentación de los resultados del TFG.

1.5. Breve sumario de productos obtenidos

En este trabajo se presentan tres productos diferentes:

- El código fuente de la aplicación de clasificación de tweets en tiempo real, formado por un conjunto de scripts y clases en lenguaje PHP, scripts en lenguaje JavaScript y documentos HTML y otros ficheros necesarios para su funcionamiento relacionados con la persistencia de objetos En el código fuente no se incluyen las bibliotecas externas utilizadas ni ningún tipo de software de apoyo.
- Un fichero de extensión SQL que incluye la base de datos MariaDB donde se almacenan los tweets y clasificaciones manuales utilizados en el proceso de entrenamiento de los algoritmos de aprendizaje supervisado.

- La memoria de trabajo, que es el documento presente donde se realiza una exposición detallada de los resultados obtenidos durante todo el proceso de desarrollo de la aplicación.

1.6. Breve descripción de los otros capítulos de la memoria

La memoria del trabajo consta de siete capítulos: un primero introductorio, uno final de conclusiones y cinco intermedios de exposición del desarrollo de la aplicación. Son los siguientes:

1. **Introducción.** En la introducción se tratarán los aspectos obligatorios contenidos en la plantilla de la memoria de trabajo: la justificación, los objetivos, la metodología, la planificación temporal, el sumario de productos obtenidos y la descripción del contenido de la memoria.
2. **Procesamiento del lenguaje natural.** Exposición teórica de la disciplina del procesamiento del lenguaje natural, explicando su historia, sus aplicaciones y limitaciones, sus principales técnicas, el estado actual y sus posibles aplicaciones futuras.
3. **Análisis de sentimientos.** Se hará una exposición teórica similar al apartado de procesamiento del lenguaje natural y se explicarán las técnicas que serán utilizadas por la aplicación que se va a desarrollar.
4. **Arquitectura de la aplicación.** Se presentarán los documentos de requisitos, análisis y diseño de la aplicación a alto nivel que se seguirán para su desarrollo, utilizando las técnicas más comunes de ingeniería de software. Se presentará una visión general de la arquitectura de la aplicación y del flujo de operación.
5. **Preprocesamiento de los datos.** Se realizará una exposición detallada de las técnicas de preparación de los datos de entrenamiento y de las técnicas de procesamiento del lenguaje natural y de análisis de sentimientos que se aplicarán sobre dichos datos.
6. **Clasificación y evaluación de resultados**. En este capítulo se expondrán las principales técnicas y métricas de valoración de los resultados de los algoritmos de clasificación utilizados, así como la descripción de los propios algoritmos y del proceso que se ha seguido.
7. **Conclusiones.** Se expondrán de manera resumida los resultados principales de los algoritmos de clasificación y los resultados generales de los productos incluidos en el TFG (la aplicación y la memoria de trabajo). Además, se hará una breve crítica a las limitaciones de las disciplinas y técnicas utilizadas y se propondrán líneas de mejora y funcionalidades añadidas a la aplicación.

2. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural, a partir de ahora PLN, es una disciplina técnica y de investigación fruto de las aportaciones de la lingüística y las ciencias de la computación en el estudio de los lenguajes naturales. Los lenguajes naturales son todos aquellos lenguajes hablados y escritos para su utilización en la comunicación de propósito general entre los seres humanos. A diferencia de los lenguajes formales, que han sido diseñados de manera estructurada para un propósito específico, los lenguajes naturales son el resultado de la acción espontánea de los humanos en el transcurso de su actividad comunicativa. El objetivo último del PLN es la compresión completa de los fundamentos del entendimiento, la generación y la expresión de los lenguajes naturales para la creación de interfaces de comunicación entre los ordenadores y los seres humanos. Enmarcado en la actualidad como una subdisciplina de la inteligencia artificial, la finalidad del PLN es la construcción de una computadora capaz de comunicarse con otras computadoras y con los seres humanos utilizando cualquiera de los lenguajes naturales existentes. Un reto tremendamente complejo que, si bien no se ha conseguido todavía, ya ha generado numerosas aplicaciones prácticas cuyo uso se ha generalizado en la era de internet y que se abordarán más adelante en este capítulo.

Sin embargo, la resolución final del problema del PLN es una tarea que, pese a los avances recientes, todavía está lejos de ser alcanzada debido a la complejidad intrínseca que presentan los lenguajes naturales. La dificultad principal es que el lenguaje natural es localmente ambiguo (Carbonell, 1992), es decir, que una expresión de un lenguaje específico puede interpretarse de muy diversas maneras en función de las ambigüedades lingüísticas y del contexto en el que se inscriba dicha expresión. El contexto se refiere a toda aquella información (y cómo ésta se estructura) que no está incluida en la propia expresión del lenguaje pero que es necesaria para otorgar un significado a ésta. Por su parte, una expresión del lenguaje natural puede presentar tres tipos de ambigüedades: léxicas, referidas a la variabilidad del significado de una palabra según su uso en una expresión; referenciales, que son las relativas al uso de pronombres o sintagmas nominales para referirse a conceptos descritos con anterioridad; y estructurales, en las que el uso de una misma figura sintáctica puede modificar la estructura de la expresión en función del significado de las otras figuras con la que se utilice. La resolución de estas ambigüedades y la aplicación de un contexto correcto son tareas que, además de ser computacionalmente complejas, todavía no han sido sistematizadas, ya que no son comprendidos del todo los mecanismos teóricos para su resolución ni se ha conseguido diseñar un algoritmo universal para su procesamiento.

Pese a ello, cualquier sistema de PLN, independientemente de su ámbito de aplicación o de las técnicas que utilice, siempre debe llevar a cabo un conjunto de tareas de análisis de la expresión del lenguaje natural que quiera procesar para facilitar su objetivo posterior. Estas tareas se agrupan en cuatro niveles de abstracción que van desde la comprensión morfológica hasta el entendimiento del contexto. Según (Carbonell, 1992), estos niveles son:

- Análisis morfológico: es el análisis de cada una de las palabras que forman las oraciones para extraer raíces, rasgos reflexivos, unidades léxicas compuestas y otros fenómenos.
- Análisis sintáctico: es el análisis de la estructura sintáctica de la oración mediante el uso de la gramática formal del lenguaje natural en el que esté expresada o por medio de métodos estadísticos.
- Análisis semántico: se refiere a la obtención del significado de la frase y la resolución de ambigüedades léxicas y estructurales. Para ello se pueden utilizar métodos estadísticos o métodos de representación del conocimiento tales como las ontologías.
- Análisis pragmático: es el análisis del contexto de la expresión, es decir, toda aquella información no incluida en una oración y sin la cual no puede extraerse su significado ni resolver ambigüedades referenciales. En este nivel de análisis de incluye el tratamiento del lenguaje figurado y el conocimiento del dominio específico del que trata la expresión.

La existencia de estos cuatro niveles no implica que todas las aplicaciones prácticas del PLN los utilicen todos ni que lo hagan de manera jerárquica o lineal. Muchos sistemas utilizan de manera paralela o entremezclada los niveles de análisis y otros, como los traductores automáticos sólo utilizan alguno de los niveles, el morfológico con la gramática de la lengua y el sintáctico con diccionarios, en el caso de que no usen métodos estadísticos.

2.1. Historia del procesamiento del lenguaje natural

Antes de los primeros computadores modernos.

Es posible seguir el rastro a la idea de la construcción de una máquina mecánica de traducción automática hasta el siglo XVII, sin embargo, no es hasta la primera mitad del siglo XX cuando comienzan a hacerse intentos sistemáticos para conseguir este objetivo (Hutchins, 2005). A mediados de la década de los 30, el francés Georges Atrouni y el soviético Peter Troyanskii, cada uno de manera independiente, solicitaron en sus respectivos países la patente de una “máquina de traducción”. La propuesta de Troyanskii ya adelantaba varios de los métodos que serían utilizados en las décadas inmediatamente posteriores. Su máquina utilizaba un diccionario bilingüe automático y un esquema de codificación de las reglas gramaticales del esperanto.

Los inicios, 1947 – 1954.

Tras la aparición de los primeros computadores electrónicos digitales, el PNL quedó ya ligado para siempre a las ciencias de la computación. En esta primera década de desarrollo de la disciplina, la atención estuvo puesta sobre el desarrollo de máquinas de traducción automática debido a la influencia del contexto de la Guerra Fría, en el que las dos superpotencias buscaban facilitar el trabajo de traducción entre el inglés y el ruso. En julio de 1949 se publicaba en Scientific American *The Mathematical Theory of Communication*, un artículo

de Warren Weaver y Claude E. Shannon donde se sentaban las bases de la teoría de la información moderna y se hacía una propuesta de los principios fundamentales de los lenguajes naturales. Desde entonces, comenzó la proliferación de la investigación en PLN en las universidades de Estados Unidos y en 1954 tuvo lugar la primera demostración de la factibilidad de un sistema de traducción automática construido por IBM en colaboración con la Universidad Georgetown. Pese a que el sistema utilizaba un diccionario y una gramática muy limitadas, los resultados fueron lo suficientemente aceptables como para iniciar toda una década de inversiones e investigación por todo el mundo.

Una década de optimismo, 1954 – 1966.

Las aplicaciones prácticas anteriores a esta década utilizaban diccionarios bilingües en los que las entradas de las palabras del lenguaje fuente tenían asociadas una o varias palabras equivalentes en el lenguaje objetivo junto con algunas reglas de orden sintáctico. Sin embargo, durante esta etapa se constató la complejidad creciente de estos diccionarios y su carácter ad hoc. Se hizo patente la necesidad de aplicar nuevos métodos más sistemáticos y generales de análisis sintáctico a la que las contribuciones teóricas sobre gramáticas formales dieron respuesta.

Durante los años posteriores a la demostración de IBM una serie de hitos impulsaron una visión de futuro optimista sobre el alcance y los avances del PLN. En 1954 inició su andadura la publicación Mechanical Translation, antecesora de la moderna Computational Linguistics. En 1956 tuvo lugar la conferencia en el Darmouth College de New Hampshire en la que se sentaron las bases fundacionales de la inteligencia artificial simbólica y que ligó desde entonces el desarrollo de esta disciplina con el PLN. Un año después, en 1957, Noam Chomsky publicaba Syntactic Structures, en el que revolucionaba los conceptos tradicionales de la lingüística estableciendo una jerarquía de tipos de gramática en la que se exponían las leyes universales de los lenguajes. Y en 1964, en el MIT, finalizaba el desarrollo de ELIZA, un programa capaz de mantener por un tiempo limitado y bajo ciertas condiciones estrictas una conversación real con un ser humano.

Pese a los avances iniciales, pronto se pudo constatar la lentitud de los progresos, en parte causados por la escasa capacidad computacional de la época. En 1964, el gobierno de Estados Unidos puso en marcha el Automatic Language Processing Advisory Comitee (ALPAC) que concluyó en su informe de 1966 que no existía una perspectiva futura de encontrar utilidad en el PLN, afirmando el lento, poco preciso y caro avance de la disciplina y recomendando el fin de las inversiones, el desarrollo de programas por computador de ayuda a los traductores humanos y la reubicación de los esfuerzos de inversión en la investigación básica.

Las repercusiones del informe del ALPAC.

Pese a que el informe del ALPAC fue ampliamente criticado por su parcialidad y por su falta de visión de futuro (Hutchins, 2005), la realidad es que la

inversión se redujo al mínimo y la investigación se ralentizó hasta la década de los 80. A pesar de estas dificultades, esta época fue un momento de consolidación de los conceptos, modelos y técnicas desarrollados hasta entonces y de expansión de la comunidad de investigadores interesados por la disciplina. Además, el vertiginoso aumento de la capacidad de computación de los ordenadores hizo que se avanzara en el desarrollo de algunas aplicaciones como SYSTRAN, en el campo de la traducción automática, o el programa SHRDLU, que consistía en la reorganización de figuras geométricas mediante órdenes introducidas por humanos en lenguaje natural textual. También es la era del inicio del desarrollo de los *bots* conversacionales.

Nuevos métodos de aproximación al problema.

A partir de la década de los 90 comienza la revolución metodológica en el PLN. Durante estos años, en paralelo al surgimiento de aplicaciones reales de los algoritmos de aprendizaje automático, se comienzan a utilizar métodos estadísticos y conexionistas para resolver los problemas de la disciplina frente al clásico enfoque simbólico. Coinciden dos grandes hechos que provocan este cambio de paradigma: la expansión explosiva de internet y de la Web, que pone a disposición de los investigadores cantidades ingentes de documentos de texto para el entrenamiento de los nuevos algoritmos; y la reducción del coste y la disponibilidad ubicua de componentes de hardware cada vez más potentes para ejecutar estos complejos algoritmos. Es la era de los avances determinantes en los campos de la búsqueda y extracción de información con la aparición de los grandes buscadores de internet. Los algoritmos de aprendizaje automático implementados por métodos estadísticos o con redes neuronales han supuesto la consecución de hitos impensables en las etapas anteriores en otros campos como la traducción automática o el reconocimiento del habla, lo que ha permitido el desarrollo de sistemas cada vez más precisos y fiables como Google Translate o los asistentes virtuales Siri o Cortana, de Apple y Microsoft, respectivamente.

2.2. Enfoques y aplicaciones del PLN

Tras el recorrido histórico por la evolución del PLN, se ha podido comprobar el surgimiento de nuevos enfoques aplicables al desarrollo de la disciplina desde el adoptado en sus inicios: el enfoque puramente simbólico. Como señala (Liddy, 2001), en la actualidad, conviven tres modos de aproximarse al campo del PLN, en ocasiones utilizados de manera híbrida: el simbólico, el estadístico y el conexionista:

- El enfoque simbólico es aquel que utiliza diccionarios, algoritmos y estructuras de datos para representar las reglas de las gramáticas formales y métodos de representación del conocimiento con el objetivo de la descomposición analítica completa de un lenguaje.
- Por su parte, el enfoque estadístico utiliza técnicas matemáticas con el objetivo de obtener un aprendizaje sobre qué palabras tienen más probabilidad de aparecer junto a otras a través del entrenamiento con un

corpus muy grande de documentos de texto. En este enfoque no se recrea un modelo estructurado del fenómeno lingüístico.

- El tercer enfoque, el conexionista, es muy similar al estadístico, ya que no se genera un conocimiento estructurado sobre una lengua, sino que se utilizan representaciones subsimbólicas del problema que se quiere resolver mediante el uso de redes neuronales artificiales.

Los tres enfoques no son contradictorios entre sí, ni se utilizan de manera aislada en las aplicaciones prácticas actuales del PLN, no existe un enfoque puramente simbólico, estadístico o conexionista. Por lo general, se combinan y se construyen soluciones híbridas, en las que se aplica una u otra opción en función de las ventajas que ofrezca cada una para una tarea determinada. Las principales desventajas de los sistemas estadísticos y conexionistas son la imposibilidad de obtener un conocimiento estructurado, ya que son enfoques subsimbólicos, y el escollo que supone la obtención de una gran cantidad de datos para que se produzca el aprendizaje. Por su parte, los sistemas simbólicos son poco robustos y poco generalizables cuando se presentan alteraciones inesperadas de las reglas y estructuras gramaticales programadas.

En cuanto a las aplicaciones del PLN, es obvio que la principal es la consecución de su objetivo fundamental, la comprensión completa de los lenguajes naturales para construir computadores capaces de entenderlos y expresarse con ellos, implicando la automatización de innumerables tareas que los seres humanos realizamos de manera cotidiana. Sin embargo, en el camino hacia este fin, ya se ha aplicado el PLN en numerosos campos para la resolución de tareas más específicas, de las cuales se pueden señalar:

- Recuperación de la información. Se trata de la creación de sistemas de búsqueda de información que ofrecen una interfaz en la que los seres humanos introducen los datos de entrada en forma de lenguaje natural, facilitando como respuesta una lista de documentos relevantes relacionados con la consulta realizada. Dentro de este campo se puede incluir también la indexación, utilizada para caracterizar a un documento en función de una serie de términos clave, y el filtrado, utilizado para descartar los documentos que no son relevantes en una consulta.
- Extracción de información. Consiste en extraer de manera automática información clave de una expresión del lenguaje natural para transformarla en un conocimiento estructurado.
- Análisis de sentimientos. Tiene como objetivo la clasificación de expresiones del lenguaje natural según su polaridad positiva, negativa o neutra respecto a un tema determinado. El PLN se aplica en las fases de tratamiento previo de la información que después servirá para alimentar el conjunto de entrenamiento de algún algoritmo de aprendizaje supervisado.
- Traducción automática. La traducción automática ha sido el primer y más importante campo de aplicación del PLN a lo largo de su historia. Como su nombre indica se trata de la traducción sin intervención humana entre

textos de cualquier combinación posible de idiomas. En la actualidad es uno de los campos en los que más fiabilidad se está obteniendo mediante la aplicación de enfoques estadísticos y conexionistas. El ejemplo paradigmático de traductor automático es el servicio de Google Translate.

- Asistentes virtuales personales. Son herramientas software que combinan varias de las aplicaciones del PLN para ofrecer ayuda a los usuarios del sistema y ejecutar una serie de tareas ordenadas por éstos. Combinan el reconocimiento del habla, la extracción y recuperación de información y técnicas de aprendizaje automático. En la actualidad el uso de los asistentes virtuales es masivo, ya que están incorporados en los dos grandes sistemas móviles: Android y iOS, con Siri y Google Aissistant, respectivamente. Además, existen los asistentes de Microsoft (Cortana) y de Amazon (Alexa).

2.3. Retos del PLN y el futuro de la disciplina

Comprensión del lenguaje natural

Una buena parte de la comunidad de investigación del PLN argumenta que el reto principal de la disciplina es la comprensión completa del lenguaje natural, ya que es el prerrequisito fundamental para su generación artificial. El consenso es que ninguno de los sistemas actuales exhibe un entendimiento real del lenguaje, marcando el objetivo primario de ser capaces de desarrollar un modelo que lo comprenda como lo hacen los seres humanos. Para ello se plantean varias cuestiones. La primera es hasta qué punto se deberían incorporar modelos simbólicos de representación del conocimiento como punto de partida de los sistemas estadísticos y conexionistas o si, por el contrario, se debería partir desde cero, dejando a los algoritmos aprenderlo todo por sí solos. La segunda es que el lenguaje natural en los humanos se aprende y desarrolla de manera inseparable de las experiencias sensoriales del cuerpo interactuando con su entorno y que, por tanto, aunque exista un algoritmo universal del lenguaje natural, este debe ser desarrollado en paralelo a un sistema sensorial o, al menos, a un entorno simulado. La última de las cuestiones es la de la relación de la inteligencia artificial y todas sus subdisciplinas con la neurociencia y otras ciencias cognitivas, y qué es posible trasladar de unas a las otras y viceversa.

Procesamiento del lenguaje natural en situaciones de escasos recursos

Uno de los problemas principales a los que se enfrenta el PLN en la actualidad es la falta de documentos escritos para alimentar los algoritmos de aprendizaje para algunos idiomas o dialectos minoritarios. Para ello se proponen algunas soluciones como la formación de especialistas en PLN en dichos lenguajes o la construcción colaborativa de bases de datos abiertas de recursos, como openAFRICA para los más de mil idiomas vivos del continente. Por otro lado, también se propone la construcción de un modelo de lenguaje universal capaz de reconocer representaciones translingüísticas.

Tratar con expresiones del lenguaje de gran tamaño

El tratamiento de contextos grandes requiere un escalado ingente de los sistemas de PLN actuales hasta que puedan ser capaces de comprender, por ejemplo, un libro entero. El dilema de la comunidad de investigación se encuentra en que, si, sencillamente, esto va a ser posible aumentando la cantidad de entrenamiento de los algoritmos y la capacidad de computación de los sistemas o si, por el contrario, es necesario el desarrollo de nuevos modelos y algoritmos.

Definición precisa de los retos y problemas a los que se enfrenta el PLN

Todavía no se ha alcanzado un consenso sobre los problemas y retos fundamentales a los que se enfrenta el PLN entre la comunidad investigadora y, por tanto, no se han desarrollado definiciones formales de estos problemas ni técnicas de evaluación precisas que permitan medir de manera concisa el progreso alcanzado hacia la consecución de un objetivo concreto.

3. Análisis de sentimientos

Una vez repasados los fundamentos del PLN, es posible ofrecer una visión general de una de sus aplicaciones más importantes, el análisis de sentimientos, área en la que se inscribe la herramienta desarrollada con este trabajo. El análisis de sentimientos o minería de opiniones (término más adecuado por las características del trabajo) es, según (Liu, 2010), el estudio computacional de las opiniones, sentimientos o emociones expresadas en texto. Aunque esta definición sea ambigua, ya que no se especifica una descripción formal del objeto de estudio, y reducida, ya que lo limita a las expresiones textuales (dejando a un lado cualquier otro método de expresión del lenguaje natural) es válida para el desarrollo del trabajo, pues se estudiarán las opiniones políticas expresadas en mensajes de Twitter en formato de texto.

Para una definición más precisa, es necesario exponer una descripción formal del concepto de opinión. Para empezar, una de las primeras cuestiones que se abordan en el análisis de sentimientos es que en un mismo documento de texto pueden convivir partes de él que expresen opiniones diferentes sobre entidades diferentes, sobre un conjunto de ellas o sobre partes de ellas, o no expresar opinión en absoluto. Por ello, se debe distinguir la entidad objetivo sobre la que se expresa una opinión.

Esta entidad puede ser un concepto, un producto, una persona, un evento, una institución, un proceso, etc. Y puede formar parte de una jerarquía de composición como parte de una entidad mayor. Por ejemplo: en un mismo texto se puede expresar una opinión sobre un teléfono móvil y sobre su batería. (Liu, 2010) denomina objetos a estas entidades y atributos a sus partes constituyentes. Por lo tanto, una opinión se expresa sobre el atributo de un objeto. Y en un documento pueden aparecer diversas opiniones.

Por otro lado, una opinión puede ser de varios tipos: cuando su intención es calificar un atributo o un objeto, decimos que es directa, frente a cuando su intención es establecer una relación de orden cualitativo entre dos o más objetos, siendo entonces comparativa. Además, las opiniones pueden ser explícitas, cuando se expresan sin ambigüedad en una oración subjetiva (por ejemplo: el nuevo iPhone es genial) o implícitas, cuando se deben deducir de una oración, en principio, objetiva (por ejemplo: el nuevo iPhone se me ha roto en dos semanas).

Por último, como definición formal, se puede establecer que una opinión es una valoración polarizada positiva, neutra o negativa, que puede ser expresada de diferentes maneras, sobre un atributo de un objeto o sobre un objeto completo. El tipo de variable para medir dicha polaridad puede ser categorial, en el caso de que se establezcan las tres opciones anteriores, o numérico, tanto continuo como discreto.

El análisis de sentimientos no sólo se utiliza para detectar la polaridad expresada sobre objetos, sino que, también, como se explica en la definición dada al principio del capítulo, se aplica a la detección del tipo de emoción

contenida en un texto (rabia, miedo, tristeza, alegría, etc.) o a la intención del emisor implícita en él, aunque este tipo de aplicaciones quedan fuera del alcance de este trabajo.

3.1. Historia, aplicaciones y retos del análisis de sentimientos.

Para (Pang y Lee, 2008) el año 2001 puede servir como fecha en la que comienza a extenderse en el mundo de la investigación la consciencia sobre los problemas y las oportunidades que el análisis de sentimientos trae consigo. Desde ese año se han publicado cientos de estudios relacionados con la disciplina. Para estos autores, esta expansión se debe a varios factores, entre ellos: el crecimiento del uso de las técnicas de *machine learning* en el PLN en general y en las aplicaciones de recuperación de la información en particular, la disponibilidad para los investigadores de enormes conjuntos de datos para el entrenamiento de los algoritmos de aprendizaje, debido al auge de la web, y el creciente interés en los retos científicos y las oportunidades comerciales que ofrece el desarrollo de este campo del conocimiento.

Respecto a las aplicaciones del análisis de sentimientos, éstas son múltiples, sin embargo, la que goza de una mayor expansión es la que se utiliza para conocer la valoración de los clientes y usuarios sobre un producto o servicio que ofrezca una empresa, especialmente aquellas valoraciones vertidas en redes sociales, permitiendo cambios y mejoras en sus acciones comerciales o de promoción casi en tiempo real, sustituyendo a las tradicionales encuestas de opinión. En el terreno de la opinión política, sin embargo, el análisis de sentimientos es una disciplina que todavía no se ha implementado de manera amplia. Según (Arcila-Calderón et al., 2017), la extensión de este campo se reduce al trabajo de clasificación manual que realizan algunos institutos de investigación sociológica y al análisis automatizado con diccionarios de etiquetas de polaridad sobre palabras. Sin embargo, la posibilidad de la aplicación de algoritmos de aprendizaje supervisado en la minería de opinión ha llevado a un creciente interés de empresas, organismo públicos y organizaciones de la sociedad civil por la disciplina, ya que les puede permitir conocer de una manera rápida y precisa la opinión de la ciudadanía sobre acciones e ideas políticas.

Por último, es fundamental destacar que el análisis de sentimientos, como campo de aplicación del PLN, adolece de los mismos problemas que esta disciplina. El manejo de la ambigüedad de los lenguajes naturales, fruto de su complejidad y de la aplicación de un contexto en su expresión, supone un enorme reto para la minería de opinión. El significado de una palabra y, por tanto, su polaridad, pueden variar en función del dominio en el que se inscribe o del contexto en el que se expresa. El uso de la negación o de símbolos como la exclamación pueden cambiar completamente el sentido de una oración o la intensidad con la que se expresa una opinión. El recurso a la ironía o las frases hechas y los modismos son complejos de entender incluso para los humanos.

3.2. Métodos de clasificación para el análisis de sentimientos

El análisis de sentimientos es, según lo expuesto en los apartados anteriores, un problema de clasificación de documentos de texto, en el que la clase a la que pertenecen es una valoración polarizada sobre una entidad determinada. En el caso concreto de este trabajo, se tratará la clasificación de documentos completos de texto corto (tweets) en la polaridad positiva, neutra o negativa que expresen sobre los cuatro clivajes políticos explicados en el capítulo introductorio como entidades sobre las que recae la opinión. Para llevar a cabo esta clasificación existen diversos métodos con los que se pueden obtener resultados aceptables y que se pueden agrupar en dos grandes grupos: de aprendizaje supervisado y los que están basados en características lingüísticas.

Los métodos de aprendizaje supervisado utilizan algoritmos de aprendizaje computacional supervisado para detectar la opinión expresada sin necesidad de construir una estructura lingüística formal del documento de texto que se pretende clasificar. Esto es así debido a que predicen la clase a la que pertenece dicho elemento por su semejanza respecto a un conjunto de objetos de ejemplo en los que su clase es conocida de antemano y con los que se ha entrenado el algoritmo en una fase previa a dicha predicción. En los métodos de aprendizaje supervisado no hay, por lo tanto, un modelo lingüístico del documento de texto sobre el que se pude extraer un conocimiento estructurado, sino que se construye un modelo matemático que permite establecer la similitud entre los distintos documentos según las características que se han seleccionado de éstos para poder representarlos.

Estos tipos de métodos ofrecen buenos resultados en la tarea del análisis de sentimientos, ya que hacen un baipás sobre los problemas relacionados con el contexto, la polisemia o la ironía en el lenguaje al seguir un modelo de aprendizaje por “imitación”, sin tener que realizar un análisis estructurado de estos fenómenos. Sin embargo, presentan una gran desventaja, y es que necesitan un conjunto de datos de entrenamiento inicial cuyos elementos hayan sido clasificados previamente de manera manual. Además, este conjunto de datos debe ser, por lo general, de un gran tamaño, por lo que no siempre existe la posibilidad de encontrar un conjunto de estas características. Por otro lado, estos métodos son muy dependientes del dominio que se quiera tratar, ya que un mismo conjunto de datos de entrenamiento no puede ser utilizado para predecir la clase de unos objetos que no pertenezcan a su misma área de conocimiento, pues, de esta manera, los resultados obtenidos empeorarían considerablemente.

Por su parte, los métodos basados en características lingüísticas permiten extraer la opinión de un texto construyendo un modelo de algunas de esas características y realizando una serie de operaciones o de deducciones sobre ellas. Entre estos métodos destacan los que hacen uso de un diccionario de palabras etiquetadas con su polaridad. En ellos las palabras de un texto se etiquetan mediante una búsqueda en el diccionario para después ejecutar algún tipo de función que da como resultado la polaridad global del texto. Estos métodos no tienen en cuenta el contexto del documento, que puede hacer que

la polaridad de una misma palabra varíe, hecho que puede resolverse utilizando diccionarios específicos de dominio, aunque no siempre existen y su elaboración debe ser manual. Sin embargo, en general, los métodos basados en diccionario son menos dependientes del dominio que los algoritmos de aprendizaje supervisado.

Otros métodos destacables son los que buscan patrones sintácticos en las oraciones del texto obteniendo la categoría gramatical de las palabras que lo forman. En estos métodos, al igual que en los que utilizan diccionarios, debe existir una base de conocimiento que etiquete con una polaridad determinada a los patrones y, por lo tanto, tienen las mismas ventajas y desventajas que ellos respecto a los métodos de aprendizaje supervisado.

En la herramienta desarrollada en este trabajo, se utilizará un conjunto de algoritmos de aprendizaje supervisado para la clasificación de la opinión de los tweets. Sin embargo, en la selección de los atributos de los elementos que formarán parte del conjunto de datos de entrenamiento y predicción para esos algoritmos, se han añadido algunos que provienen de las características lingüísticas de los textos utilizando técnicas basadas en diccionario. En concreto, se etiquetará cada palabra y cada emoticono de cada tweet con una polaridad positiva o negativa, como se puede leer en el capítulo 5, dedicado al preprocesamiento de los datos.

4. Arquitectura de la aplicación

El sistema de clasificación de polaridad para clivajes políticos de mensajes obtenidos en tiempo real de la API de Twitter se compone de cinco elementos a alto nivel:

- Un conjunto de clases que realizan toda la funcionalidad de procesamiento del lenguaje natural (normalización de tweets, reducción de características y selección de atributos) y de utilización de los algoritmos de aprendizaje supervisado para el análisis de sentimientos.
- Una aplicación web con una arquitectura de tres capas que sigue el patrón modelo-vista-controlador y que utiliza el conjunto de clases anteriores para mostrar los resultados de la clasificación de polaridad política de los tweets obtenidos en tiempo real.
- Una base de datos relacional que almacena el modelo de datos utilizado para el entrenamiento de los algoritmos de aprendizaje supervisado que clasifican los tweets.
- Un servicio de consumo de la API REST de Twitter encargado de obtener los tweets que se utilizarán en el entrenamiento de los algoritmos de clasificación y de almacenarlos en la base de datos.
- Un servicio de consumo de la API streaming de Twitter, encargado de conectarse a ella, recibir los tweets en tiempo real de manera asíncrona y almacenarlos serializados en un fichero.

Esta separación de los cinco elementos a alto nivel es lógica o analítica, ya que, realmente, el paquete de clases para PLN y análisis de sentimientos, la aplicación web y los servicios de consumo de las API forman parte de un mismo proyecto de software en lenguaje PHP que se ejecuta en un mismo servidor web Apache. El único proceso independiente es el de la base de datos.

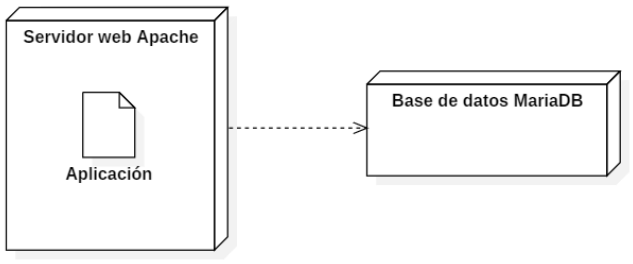


Figura 1. Diagrama de despliegue de alto nivel del sistema

Por otro lado, para el desarrollo del sistema se optó por utilizar un entorno de desarrollo WAMP, es decir, una combinación de sistema operativo Windows, servidor web Apache, base de datos MariaDB y el lenguaje de programación PHP. Respecto a las herramientas, se seleccionó Netbeans (versión 10) como entorno de desarrollo integrado, para la gestión de las dependencias externas se utilizó el software Composer y para el control de versiones se utilizó Git.

Pese a que los recursos en PHP para el desarrollo de aplicaciones del campo del procesamiento del lenguaje natural y del aprendizaje computacional son escasos y poco conocidos, se optó por este lenguaje por razones de competencias, habilidades y experiencia del autor frente a la insuficiencia en estos aspectos respecto a otras opciones como Python y Java.

La decisión final de seleccionar PHP como lenguaje para la aplicación vino motivada por la existencia de la biblioteca de *machine learning* para PHP PHP-ML, que cuenta con todas las funcionalidades necesarias para el cumplimiento de los objetivos del trabajo.

4.1. Arquitectura de la aplicación web

La aplicación web sigue una arquitectura en tres capas, implementando el patrón modelo-vista-controlador. La capa de presentación está compuesta por el fichero *index.html* y el fichero *manual_classification.html*, ubicados en directorio raíz del proyecto. El fichero de estilos CSS del *index* es *main.css* y el de *manual_classification* es styles.css, ambos ubicados en el directorio *styles* del proyecto. Por último, en la carpeta scripts se encuentran los ficheros con las funciones JavaScript. En conjunto, estos ficheros son los que el servidor web envía al ordenador del usuario y actúan como cliente consumiendo datos en formato JSON de una API mediante el uso de AJAX.

La API es la aplicación escrita en lenguaje PHP. Se puede dividir en tres módulos diferenciados:

- Un primer módulo de clasificación manual de tweets que se usarán en el conjunto de datos de entrenamiento para cada uno de los clivajes políticos.
- Un segundo módulo para obtener una evaluación de los algoritmos de clasificación utilizando el conjunto de datos de entrenamiento.
- Un tercer y principal módulo de clasificación de nuevos tweets en tiempo real.

Cada uno de estos módulos se compone de una o más clases PHP que actúan como controladores y de un fichero PHP que actúa como *endpoint* de las llamadas a la API, creando una instancia del controlador adecuado y llamando a sus métodos.

El módulo de clasificación manual utiliza el *endpoint polling.php* y está formado por los controladores *PollingController.php*, *PollingMainController.php* y *PollingPaginatorController.php*. *PollingMainController* actúa como controlador frontal, llamando al resto de controladores y sus métodos en función de los parámetros de la petición HTTP que llega al servidor web. *PollingPaginatorController* actúa como paginador, devolviendo el número de páginas en que se debe dividir la presentación en función del número de tweets totales y de la cantidad de tweets por página. El método *getTweets* de *PollingController* devuelve los tweets de una página concreta y el método *classify* se encarga de persistir la clasificación de un tweet en la base de datos.

El módulo de evaluación de los algoritmos de clasificación está formado por el *endpoint evaluate.php* y el controlador *EvaluationController.php*. El único método de este controlador, getEvaluation, se encarga de recopilar los tweets del conjunto de entrenamiento de la base de datos y pasárselos como parámetro a una instancia de la clase *Evaluation*, cuyo método *getEvaluation* devuelve una evaluación de los algoritmos de clasificación. Por último, el controlador codifica el resultado en formato JSON y lo devuelve en la respuesta HTTP al cliente. La clase *Evaluation* forma parte del conjunto de clases de PLN y análisis de sentimientos agrupadas en el espacio de nombres /api/utils/nlp y ubicadas en el directorio *utils* del proyecto.

Por su parte, el módulo principal está formado por el *endpoint stream.php* y el controlador *PredictionsController.php*. Este controlador recupera los tweets obtenidos en tiempo real de un fichero, donde el servicio de streaming los almacena serializados, y se los pasa como parámetros a una instancia de la clase *Predictions* (perteneciente al espacio de nombres api/utils/nlp), de la que se obtienen las clasificaciones finales que se devuelven en formato JSON.

El conjunto de controladores y de *endpoint* forman la capa de la lógica de negocio de la aplicación. Por otro lado, la capa de integración está formada por las clases de entidad de los tweets y de las clasificaciones de tweets del conjunto de datos de entrenamiento, así como de sus respectivos modelos. La entidad de un tweet de entrenamiento es la clase *Tweet.php* del directorio *entities* del proyecto, la entidad de una clasificación es la clase *Classification*. El modelo de un tweet lo implementa la clase *TweetModel.php* del directorio *models* del proyecto y el modelo de una clasificación lo implementa la clase *ClassificationModel.php*.

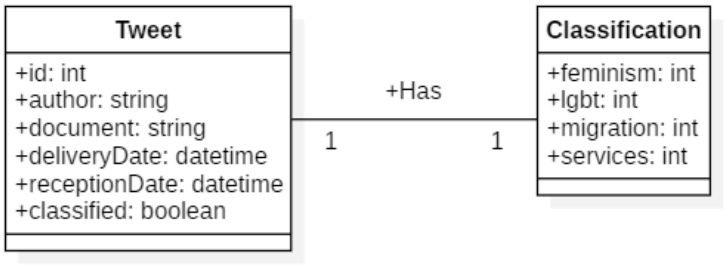


Figura 2. Modelo de datos

La figura 2 muestra el diseño del modelo conceptual de datos con los atributos de cada una de las entidades y su tipo de relación con conectividad 1:1. Tanto la clase del modelo de una entidad *Tweet* como la del modelo de una entidad *Classification* presentan métodos para obtener un solo elemento por su identificador y para persistir (insertar o actualizar) un objeto en la base de datos. El modelo de la entidad *Tweet*, además, presenta métodos para obtener todos los elementos almacenados, un subconjunto de elementos sucesivos, todos los que han sido clasificados o el número total de elementos que han sido clasificados.

4.2. Flujo general para la clasificación de tweets en tiempo real

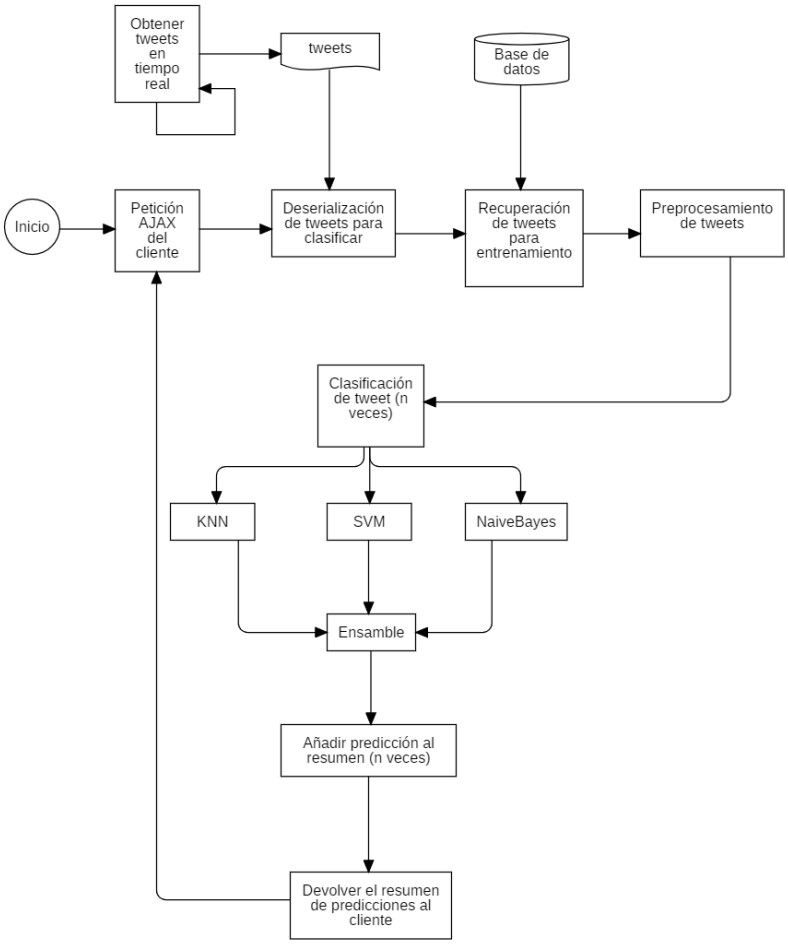


Figura 3. Proceso de clasificación

En primer lugar, se debe inicializar el script *consume.php*, ubicado en el directorio raíz de la API. En este script se establecen las constantes del sistema de autenticación OAuth que utilizará la conexión a la API streaming de Twitter y que se le pasan como parámetro al constructor de la clase *FilterTrackConsumer*, que hereda de la clase *OAuthPirehose* del paquete externo *Phirehose*[2].

Esta clase posee el método *enqueStatus*, que se ejecuta de manera asíncrona cada vez que se recibe un Tweet en tiempo real. En dicho método, únicamente se encola el tweet recibido y se comprueba si se ha llegado al número máximo de tweets establecido en la constante de clase *MAX_TWEETS*. Si se alcanza esta cifra, los tweets encolados se almacenan serializados en el fichero *tweets*, en el directorio *persistent*.

Es conveniente inicializar este script con el servidor web incluido en PHP, ya que deberá estar escuchando todo el tiempo posible. Esto se puede realizar mediante las órdenes en consola siguientes:

[2] https://github.com/fennb/phirehose

```
php -S [dirección]:[puerto]
```

Y, situado en el directorio donde se encuentra el fichero *consume.php*:

```
php consume.php
```

En paralelo al consumo asíncrono de tweets en tiempo real de la API streaming de Twitter, se debe iniciar la aplicación web de clasificación en otro servidor web. En nuestro caso, en el entorno de desarrollo se ha utilizado un servidor web Apache incluido en la solución de software WAMPP.

En este servidor, en el directorio raíz de la API, se encuentra el fichero *stream.php*, que actúa como *endpoint* de las peticiones HTTP realizadas con AJAX desde los documentos HTML y JavaScript que recibe el cliente cuando entra en la página principal de la aplicación. La responsabilidad de este *endpoint* está en crear una instancia del controlador *PredictionController* y llamar a su método *getPredictions*, que devuelve al cliente, en formato JSON, las clasificaciones de los clivajes políticos hechas sobre los últimos tweets almacenados en el fichero tweets creado por la herramienta de consumo de la API streaming de Twitter. Las peticiones AJAX desde el cliente se realizan de manera iterativa una vez que se ha obtenido una respuesta del servidor.

El método *getPredictions* de *PredictionController*, únicamente crea una instancia de la clase *Prediction*, llama a su método *getPrediction*, que devuelve la clasificación de los tweets, establece las cabeceras de la respuesta HTTP y como cuerpo convierte la dicha clasificación a formato JSON.

La clase *Prediction* es el núcleo de la operación de clasificación de los tweets. Forma parte del espacio de nombres api/utils/nlp y está ubicada en el directorio *utils* de la API. Esta clase, en su método constructor, se encarga de recuperar los tweets serializados del fichero tweet creado por el consumidor de la API streaming de Twitter. Tiene dos métodos privados que son llamados desde el método público *getPrediction*:

- preprocess: encargado de preprocesar tweets para transformarlos en un conjunto de entrenamiento y en un conjunto de prueba preparado para ser utilizado por los algoritmos de clasificación.
- classify: recibe el conjunto de muestras de entrenamiento, sus clases y el conjunto de muestras de prueba y devuelve un resumen de la predicción realizada.

El método *preprocess*, primero crea una instancia de *TweetModel* (modelo de la entidad Tweet) y llama a su método *getClassified* para obtener de la base de datos todos los tweets que han sido clasificados manualmente. Estos tweets, después, serán transformados en el conjunto de datos de entrenamiento de los algoritmos de clasificación mediante la clase *TrainningSet*. Por otro lado, mediante el uso de esa misma clase, los tweets obtenidos del consumo de la API streaming de Twitter serán transformados en el conjunto de datos del que se quiere obtener su predicción de clases. El proceso de preprocesamiento de los datos se puede leer en detalle en el capítulo 5 del trabajo.

Una vez se han obtenido el conjunto de entrenamiento, las clases de éste y el conjunto de datos a clasificar, se pasan como parámetros al método *classify*. En este método se crea una instancia de la clase *Classifier* y llama a su método *train* para entrenar a los algoritmos de clasificación y a su método *predict* por cada uno de los tweets a clasificar. Cada una de las predicciones se añade a una instancia de la clase *PredictionSummary*, que elabora un resumen de éstas y es el objeto que finalmente se devuelve en el método *getPrediction*.

La clase *PredictionSummary* tiene un método llamado *getSummary* que devuelve el atributo privado *$summary*, que es un array asociativo de tres niveles de claves: el primero es de cada clivaje político, el segundo es el de la polaridad por clivaje y el tercer nivel almacena un entero con el número de tweets que han sido clasificados para una polaridad en un clivaje determinado.


```
{
  "feminism" : {
    "positive" : 35,
    "neutral" : 55,
    "negative" : 10
  },
  "lgbt" : {
    "positive" : 20,
    "neutral" : 75,
    "negative" : 5
  },
  "migration" : {
    "positive" : 15,
    "neutral" : 80,
    "negative" : 5
  },
  "services" : {
    "positive" : 45,
    "neutral" : 50,
    "negative" : 5
  }
}
```


Figura 4. Objeto PredictionSummary en formato JSON

La clase *Classifier* tiene asignada la responsabilidad de la predicción individual de la clase de un tweet para cada uno de los algoritmos de clasificación seleccionados y el ensamble final. En su método constructor se asigna al atributo de clase correspondiente una instancia de los tres clasificadores incluidos en la biblioteca PHP ML (*KNearestNeighbors*, *SVC* y *NaiveBayes*) por cada uno de los clivajes políticos que van a ser analizados, dando como resultado un total de 12 clasificadores. El método *train* recibe como parámetros las muestras de entrenamiento y sus clases y llama al método *train* de cada uno de estos clasificadores con ellos. El método *classify* llama al método *predict* de cada uno de los clasificadores y obtiene la clase mayoritaria para cada clivaje como clasificación final utilizando el método *getMayority*.

5. Preprocesamiento de los datos

Las expresiones informales del lenguaje natural como las que se dan en los tweets que componen el conjunto de datos del sistema no suelen ser correctas ortográficamente y es habitual que hagan uso de abreviaturas propias de los microtextos y palabras pertenecientes a una jerga o argot (variante del lenguaje estándar utilizada por un grupo social determinado). Es por ello por lo que deben ser tratadas en un proceso de normalización cuyo objetivo final es reducir el número de atributos con los que los algoritmos de clasificación tienen que tratar para conseguir aumentar su rendimiento y su precisión.

En este caso, la normalización consiste en sustituir cada uno de los elementos léxicos abreviados, de jerga u ortográficamente incorrectos de un tweet por la palabra, el lema o la raíz del diccionario de la lengua estándar a la que se corresponde o por el nombre propio al que hace referencia. En este proceso también se debe considerar la eliminación o la inclusión en el conjunto de atributos de otros elementos que pueden aparecer en un tweet, como son: las URL, los emoticonos, los hashtags o las menciones.

Los trabajos sobre normalización léxica del español son escasos (Ruíz et al., 2014). La mayoría son los que se presentan a las ediciones anuales del TASS (Taller de análisis semántico), de la Sociedad Española de Procesamiento del Lenguaje Natural. En muchos de estos trabajos, el proceso de normalización se produce en dos fases:

- Una primera fase en la que se corrige el texto mediante reglas de transformación basadas en expresiones regulares sin distinción entre mayúsculas y minúsculas. En esta fase se detectan abreviaturas, onomatopeyas, emoticonos y otros elementos típicos de los tweets y se sustituyen las palabras en argot general.
- Una segunda fase en la que las palabras calificadas como OOV (fuera de diccionario o Out Of Dicctionary en inglés), es decir, que no pertenecen al léxico estándar una vez se ha procesado el texto en la primera fase, se comparan con las palabras del diccionario y con un corpus de nombres propios utilizando medidas como la distancia de Levenshtein, heurísticas o incluso algoritmos de clasificación. Después se utiliza un algoritmo de selección de la mejor candidata para su sustitución.

En nuestro sistema, únicamente se utilizarán algunas técnicas del preprocesamiento basado en reglas (que se explicarán en el siguiente apartado), ya que la implementación de los algoritmos para la selección de candidatos de diccionario y de nombres propios excede el alcance del trabajo.

5.1. Reglas de preprocesamiento

La funcionalidad del preprocesamiento basado en reglas se ha desarrollado a través de la interfaz *Normalizer*, que obliga a la implementación del método

getText, y de la clase *TweetNormalizer*, que extiende a la interfaz anterior y que implementa los métodos específicos para la corrección de tweets basada en reglas. De esta manera, se mantiene el principio de alta cohesión y se permite aumentar la funcionalidad a otros tipos de textos en un futuro mediante los mecanismos de herencia. A continuación, se muestra una lista exhaustiva de los métodos miembros de *TweetNormalizer*:

toLower: devuelve la cadena de entrada en minúsculas. Esto es necesario ya que los algoritmos de clasificación distinguirían entre una misma palabra con cualquier combinación de mayúsculas y minúsculas, aumentando de manera considerable el número de atributos de las muestras de entrenamiento y predicción.

reduceWhiteSpaces: devuelve la cadena de entrada con cada una de las series de espacios en blanco reducidas a uno sólo. Este método es necesario para poder utilizar un *tokenizador* de espacios en blanco (más adelante se explicará la fase de *tokenización* de los datos).

removeCarriageReturns: devuelve la cadena de entrada sin retornos de carro.

removeNumbers: devuelve la cadena de entrada eliminando todas las cifras numéricas. Será útil en el caso de que no se necesite tener en cuenta expresiones con cifras tales como fechas o cantidades.

replaceAccentMarks: devuelve la cadena de entrada con todas las apariciones de vocales acentuadas sustituidas por las mismas vocales sin tilde. Este método es necesario para la reducción de los atributos de las muestras de los algoritmos de clasificación, ya que distinguen entre una misma palabra con tilde y la misma sin tilde o acentuada incorrectamente debido a un error ortográfico, algo muy habitual en las expresiones informales del lenguaje natural.

removeHashtags: devuelve la cadena de entrada eliminando todas las apariciones de palabras que empiecen con el símbolo #. Con este método se consigue eliminar todos los hashtags de un tweet, ya que no se tendrán en consideración durante el proceso de selección de atributos.

removeMentions: devuelve la cadena de entrada eliminando todas las apariciones de palabras que empiecen por el símbolo @. Con este método se consigue eliminar todas las menciones a usuarios del tweet, ya que no se tendrán en consideración durante el proceso de selección de atributos.

removeURLs: devuelve la cadena de entrada eliminando las URL, que no se tendrán en consideración durante el proceso de selección de atributos.

removeRTs: devuelve la cadena de entrada eliminando las apariciones de la abreviatura de la palabra *retweet*, que suelen aparecer cuando un usuario hace un retweet manual.

normalizeLaugh: devuelve la cadena de entrada con todas las expresiones que indiquen risa normalizadas, ya que la onomatopeya de la risa suele ser escrita de muy diversas formas o mediante acrónimos.

Tras aplicar las reglas de preprocesamiento al siguiente tweet del conjunto de datos:

> Nace nuestro portal de participación en @MasMadrid__. Necesitamos toda la creatividad, el talento y la solidaridad de los madrileños y madrileñas para construir una candidatura amplia que reúna toda la excelencia de nuestra ciudad. #ÚneteMásMadrid en https://t.co/g9PiothbIp. https://t.co/c0tfMmUlQT

Se obtendría la siguiente cadena de texto:

> nace nuestro portal de participacion en . necesitamos toda la creatividad, el talento y la solidaridad de los madrileños y madrileñas para construir una candidatura amplia que reuna toda la excelencia de nuestra ciudad. en .

5.2. Tokenización

La tokenización o segmentación consiste en la división del texto en partes para formar un vector de *tokens* que, normalmente, coinciden con cada uno de los elementos léxicos de dicho texto, aunque, dependiendo del tipo de segmentación, pueden aparecer otros elementos como símbolos, signos de puntuación, de exclamación e interrogación y otros muchos.

La tokenización es un proceso fundamental en el análisis de sentimientos, ya que, a partir del vector resultante de aplicarla a un texto se extraen los atributos que finalmente serán considerados en los algoritmos de clasificación.

En el caso de nuestro sistema, este proceso se llevará a cabo mediante el método *tokenize* de la clase *WordTokenizer* de la librería PHP-ML. La tokenización con esta clase selecciona tokens del texto de entrada de 2 o más caracteres alfanuméricos ignorando los signos de puntuación y tratándolos como elementos separadores de los tokens.

Por ejemplo, el siguiente tweet del conjunto de datos:

> Los de los autobuses medievales sacan hoy uno pidiendo a Casado, Abascal y Rivera que deroguen las leyes contra la violencia machista. Del contenido ni hablamos, para qué. Pero ¡ay, los tiempos del "liberalismo progresista"! Ciudadanos en la caverna de los homófobos y machistas.

Se transformaría en el siguiente vector (tras haber pasado por el preprocesamiento basado en reglas y la eliminación de stop words):

Tabla 2. Vector de tokens de un tweet

0	autobuses	11	machista
1	medievales	12	contenido
2	sacan	13	hablamos
3	hoy	14	ay
4	pidiendo	15	tiempos
5	casado	16	liberalismo
6	abascal	17	progresista
7	rivera	18	ciudadanos
8	deroguen	19	caverna
9	leyes	20	homofobos
10	violencia	21	machistas

5.3. Reducción de atributos

Una vez se ha conseguido un texto normalizado y se ha obtenido su vector de segmentos o tokens, se debe realizar un proceso de reducción de los atributos que después se tendrán en cuenta en la fase de clasificación. En este caso se utilizarán dos métodos de reducción: la eliminación de palabras vacías (stop words en inglés) y el *stemming*.

5.3.1. Eliminación de stop words

Las palabras vacías o stop words en inglés son palabras que se utilizan para construir la estructura gramatical de una oración pero que no aportan significado a ésta, son los artículos, pronombres, preposiciones y otras. Son palabras que aparecen de manera muy frecuente y que deben ser filtradas del texto que se va a clasificar para reducir la cantidad de atributos a tener en cuenta. La lista de stop words que se ha utilizado en el modelo se ha obtenido del repositorio de Github Stopwords ISO[3].

Para hacer más sencillo el modelo, se ha decidido que el proceso de eliminación de stop words se realice durante la fase de preprocesamiento

[3] https://github.com/stopwords-iso/stopwords-es

basado en reglas, antes de la eliminación de las tildes del texto original y tomando como entrada del método la cadena completa y no su vector de tokens, ya que las palabras vacías son ortográficamente correctas e incluyen las marcas de acentuación y su eliminación es más simple cuando se utilizan funciones del lenguaje de manipulación de cadenas sobre el texto completo como entrada.

La eliminación de stop words se lleva a cabo mediante el método *removeStopWords* de la clase *FeaturesReductor*. Este método recorre un vector constante de palabras vacías que se obtiene por el método *getStopWords* de la clase *SpanishStopWords* (que sobre escribe el método de la interfaz *StopWords*) y utiliza la función del lenguaje *str_replace* para eliminar las coincidencias de cada una de las palabras en el texto de entrada.

5.3.2. Stemming

Stemming es un proceso cuyo objetivo es reducir una palabra a su raíz, eliminando los afijos que se le añaden para modificar su significado. Mediante esta técnica se consigue reducir el conjunto de atributos a tener en cuenta en los algoritmos de clasificación, ya que varias palabras se agrupan por su raíz compartida.

Para llevar a cabo el proceso de stemming en el modelo, se ha utilizado la librería *php-stemmer*[4], que implementa en PHP el sistema Snowball, basado en el algoritmo de Porter, uno de los más utilizados en el campo.

El proceso de stemming se lleva cabo tras el procesamiento basado en reglas del texto y su tokenización, aplicando el método *stem* de la interfaz *Stemmer* sobre cada uno de los tokens del vector obtenido en la fase previa de preprocesamiento.

5.4. Selección de atributos

Una vez se han normalizado los tweets del conjunto de datos aplicándoles las reglas de preprocesamiento, se ha reducido el número de palabras diferentes que contienen y se ha obtenido su vector de tokens, se deben seleccionar cuáles serán los atributos que se tendrán en cuanta en los algoritmos de clasificación.

El método más habitual de representar estos atributos en el PLN es mediante el uso de la bolsa de palabras o *Bag of Words* en inglés. Se trata de representar los documentos de texto como un vector de distintas palabras sin orden, en el que a cada una de ellas se le asigna una medida de frecuencia, esta medida puede representarse como una característica binaria si la palabra aparece en el texto o no aparece, o como una medida de su frecuencia de aparición, tanto absoluta como relativa, respecto al documento o a todo el conjunto de datos.

[4] https://github.com/wamania/php-stemmer

También es posible añadir a la lista de características además de las medidas de frecuencia de la bolsa de palabras otro tipo de atributos muy comunes en el ámbito del PLN y el análisis de sentimientos, como son el léxico de polaridad, la categoría gramatical de las palabras (Parts of Speech en inglés) y la polaridad de los emoticonos.

5.4.1. Frecuencia de término – frecuencia inversa de documento

Como medida de frecuencia de los tokens se ha seleccionado la frecuencia de término – frecuencia inversa de documento (TF-IDF, por sus siglas en inglés). Puede ser intuitivo pensar que las palabras más frecuentes en un documento de texto son las que pueden indicar mejor su significado, sin embargo, los trabajos en PLN indican que las palabras que orientan la polaridad y el significado de un texto suelen ser bastante infrecuentes en el conjunto de datos. Por ello, se ha seleccionado esta medida de frecuencia, que indica el nivel de concentración de un token en pocos documentos del conjunto de datos.

Se define frecuencia de un término en un documento como el número de ocurrencias de ese término en el documento (f) dividido entre la frecuencia máxima de una palabra de ese mismo documento (max). Por su parte, la frecuencia inversa de documento es el logaritmo en base dos del número total de documentos en el conjunto de datos (N) dividido entre el número de documentos donde aparece el término (n). La medida de frecuencia final es el producto de estos dos resultados, tal y como se muestra a continuación:

$$TFIDF = \frac{f}{max} \cdot \log_2 \frac{N}{n}$$

La formación del vector de tokens con sus medidas TF-IDF se realiza en dos pasos:

- Se instancia la clase *TokenCountVectorizer* de la librería PHP-ML pasando a su método constructor una instancia de la clase *WordTokenizer*. Después, se llama al método *fit* de esta clase pasando como parámetro el conjunto de tweets normalizados, con esto se construye el diccionario de tokens de todo el conjunto de datos. Tras esto, se llama al método *transform*, pasando como parámetro los tweets normalizados, con lo que obtenemos una matriz de vectores de tokens de cada documento con la frecuencia absoluta de cada palabra.
- El segundo paso consiste en obtener una instancia de la clase *TfIdfTransformer* de la librería PHP-ML y llamar a su método *transform* pasándole como parámetro la matriz de vectores de tokens con frecuencias absolutas que se ha obtenido en el paso anterior. Este método devuelve la misma matriz con medidas TF-IDF. Esta matriz ya puede pasarse como parámetro de los métodos de entrenamiento y predicción de los distintos algoritmos de clasificación.

5.4.2.Léxico de polaridad

El léxico de polaridad es una lista de palabras de un idioma etiquetadas por su polaridad positiva o negativa respecto al significado emocional que aportan al texto. También existen otros léxicos que etiquetan las palabras por el tipo de emoción que aportan, ampliando las clases de dos a varias (alegría, sorpresa, ira, miedo, tristeza, etc.). El léxico de polaridad se utiliza para añadir como atributos del conjunto de datos la frecuencia de aparición en cada uno de los documentos de texto de palabras positivas y negativas, lo que puede resultar en una mayor precisión de los algoritmos de clasificación, al tenerse en cuenta estos aspectos.

En el modelo se extraerá un vector de dos posiciones, la primera con la cantidad de palabras positivas y la segunda con la cantidad de palabras negativas de cada uno de los tweets del conjunto de datos. Está operación se llevará a cabo antes del preprocesamiento basado en reglas por medio del método *countLexicon* de la clase *AdditionalFeatures*. El léxico se extrae de un documento CSV que contiene los términos generales y las interjecciones del diccionario ElhPolar del modelo presentado por la fundación Elhuyar a la edición del año 2013 del TASS[5].

5.4.3.Emoticonos

Los emoticonos, según la Real Academia de la Lengua Española, son una combinación de símbolos de teclado que se asemejan gráficamente al concepto que quieren describir, normalmente un estado de ánimo o una emoción. Su uso es muy popular en las aplicaciones de chat y en las aplicaciones de redes sociales, especialmente en Twitter. Por ello, pese a que no hay conclusiones definitivas sobre su utilidad para determinar la polaridad de un texto y sólo unos pocos emoticonos se muestran como signos precisos del sentimiento (Wang y Castanon, 2015), se ha decidido tenerlos en cuenta como atributos de los algoritmos de clasificación.

En el modelo se seguirá la misma lógica que para el tratamiento del léxico de polaridad, es decir, se obtendrá la frecuencia con la que aparecen emoticonos en cada tweet según su polaridad positiva o negativa. Se ha obtenido una lista de 118 de los emoticonos Unicode más utilizados en formato JSON[6], en la que se ha clasificado manualmente la polaridad en una escala desde el valor -5 (más negativo) hasta el 5 (más positivo). Sin embargo, en el modelo, se considerarán como negativos los emoticonos de polaridad menor que 1 y positivos los de polaridad mayor o igual a 1.

La operación se lleva a cabo mediante el método *countEmojis* de la clase *AdditionalFeatures*, que devuelve un vector de dos posiciones: la primera contiene el número de veces que aparecen emoticonos positivos y la segunda el número de veces que aparecen emoticonos negativos.

[5]Saralegi, X., San Vicente, I. (2013). "Elhuyar at TASS 2013", en *XXIX Congreso de la Sociedad Española de Procesamiento de lenguaje natural,* Madrid, SEPLN.

5.4.4.Parts of speech

Parts of speech es un método para ampliar el conjunto de atributos de los datos asociando a cada palabra de un documento la categoría gramatical que le corresponde y obteniendo una medida de frecuencia de la aparición de cada una de estas categorías en el texto.

En el modelo se ha implementado mediante el método *POSTagger* de la clase *AdditionalFeatures*. Este método emplea un envoltorio en PHP[7] que utiliza el software *Log-linear Part of Speech Tagger* de la Universidad de Stanford. El método devuelve un array asociativo en el que las claves son el tipo de categoría gramatical y los valores representan las cantidades de palabras en el texto que pertenecen a cada una de estas categorías.

5.4.5. Resultado final

Los atributos finales de cada tweet que se van a tener en cuenta a la hora de realizar la clasificación con los algoritmos de aprendizaje supervisado son, por tanto:

- La frecuencia inversa de documento de cada una de las raíces de las palabras del texto del tweet tras aplicársele las reglas de preprocesamiento y la eliminación de stop words.
- La frecuencia de absoluta de léxico de polaridad positiva y de léxico de polaridad negativa.
- La frecuencia absoluta de emoticonos de polaridad positiva y de emoticonos de polaridad negativa.

Los atributos se representarán en PHP mediante una lista que seguirá el orden de aparición en este texto.

---

[6] https://github.com/words/emoji-emotion

[7] Hays, C. (2019): "PHP Class Wrapper for Stanford Parto f Speech Tagger". *Charles Hays,* 3 de febrero. Disponible en: http://charleshays.com/php-class-wrapper-for-stanford-part-of-speech-tagger/ [Consulta: 15-04-2019]

6. Clasificación y evaluación de resultados

La clasificación es la tarea fundamental del aprendizaje supervisado. Consiste en predecir la clase a la que pertenece un elemento del conjunto de datos dado su vector de atributos. Se diferencia del aprendizaje no supervisado en que los algoritmos ejecutan una fase previa de entrenamiento en la que se alimentan de elementos cuya clase es conocida. Tras esta fase, la predicción puede realizarse mediante diversos métodos: por regresión estadística, haciendo uso del Teorema de Bayes, separando los elementos con funciones en un espacio n-dimensional, disminuyendo el error de aproximación a la función objetivo modificando los pesos de una red neuronal artificial, comparando las distancias del vector de atributos del elemento a clasificar con la de los elementos del conjunto de entrenamiento, utilizando una estructura arborescente de clasificación o muchas otras más.

En nuestro caso, se utilizará un conjunto de clasificadores (*ensamble*) en el que la clase de salida será la que aparezca más a menudo en cada una de las predicciones de los miembros que lo forman. Los algoritmos que se utilizarán serán tres: k vecinos más próximos, una máquina de vectores de soporte y un clasificador bayesiano, todos ellos incluidos en la biblioteca PHP-ML. Los atributos del vector de entrada serán los descritos en el capítulo 5, dedicado al preprocesamiento de los datos. Las posibles clases de salida son tres y se corresponden con cada una de las polaridades establecidas en el análisis de sentimientos: favorable, neutra y desfavorable. Se hará una predicción de cada elemento para cada uno de los clivajes políticos nombrados en la introducción del trabajo.

6.1. K vecinos más próximos

En este método, el entrenamiento consiste en almacenar una serie de ejemplos de los que se conoce su clase de antemano. La clase del objeto que se quiere predecir será, entonces, la clase del elemento del conjunto de entrenamiento más cercano a él. La generalización de este algoritmo, que es la que se utilizará en la aplicación, escoge la clase mayoritaria entre los k elementos más cercanos al objeto de entrada. En el modelo que se ha desarrollado se asigna el valor 3 a k.

Para determinar qué elementos son los más cercanos se utiliza una medida de distancia matemática, en nuestro caso, la distancia euclidiana, que, para el par de elementos x_j y x_k se define de la siguiente manera:

$$d(x_j, x_k) = (\sum_{i=1}^{M} (A_i(x_j) - A_i(x_k))^2)^{1/2}$$

Donde A_i es el atributo i-ésimo de cada uno de los elementos y M es el número de atributos.

La clase que implementa este algoritmo en PHP-ML es *KNearestNeighbors*. Todas las clases que definen clasificadores en esta biblioteca implementan una

interfaz con los métodos *train* y *predict,* para entrenar el modelo y para predecir la clase de un elemento del conjunto de datos, respectivamente.

6.2. Máquina de vectores de soporte

Las máquinas de vectores de soporte (SVM por sus siglas en inglés, *Support Vector Machines*) separan los objetos del conjunto de datos por medio de una función de frontera (llamada Kernel) en el espacio M-dimensional que define dichos objetos, donde M es el número de atributos de cada uno de ellos. Las SVM se suelen utilizar con una función lineal que separa los objetos en dos clases, sin embargo, en nuestro modelo, se utilizará la generalización que se acaba de definir, puesto que se usará una función polinomial y las posibles clases de salida son tres: polaridad positiva, neutra o negativa.

El objetivo del algoritmo de la SVM es encontrar los parámetros de la función de frontera utilizada que maximice su distancia a los conjuntos de elementos pertenecientes a cada una de las clases, es decir, encontrar los parámetros que permitan una mejor separación entre los elementos de las clases. Las SVM, además, aceptan otros parámetros de entrada como: K, que indica hasta qué punto un elemento puede violar la clasificación resultante o C, que indica el grado de penalización por una mala clasificación de un elemento. La clase que implementa una SVM en PHP-ML es *SVC*.

6.3. Clasificador bayesiano

Los clasificadores bayesianos, establecen la clase de un objeto del conjunto de datos mediante el estudio de su probabilidad condicionada respecto a la probabilidad de sus atributos utilizando el Teorema de Bayes, que se expresa de la siguiente manera:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Donde $P(A \mid B)$ es la probabilidad condicionada de A, es decir, la probabilidad de que A ocurra cuando B es cierto. $P(B \mid A)$ es la probabilidad condicionada de B a A, y $P(A)$ y $P(B)$ son las probabilidades de A y B.

Este tipo de clasificadores se consideran “ingenuos” debido a que asumen la independencia de los atributos del objeto de entrada, lo que, en el caso de características provenientes del PLN, indica que las palabras no tienen relación sintáctica entre sí.

La clase que implementa un clasificador bayesiano en PHP-ML es *NaiveBayes*.

6.4. Métricas de evaluación del clasificador

Para la evaluación de los resultados de predicción del clasificador se va a utilizar una tabla comparativa con todas las métricas que ofrece la clase *ClassificationReport* de la biblioteca PHP-ML para cada uno de los métodos presentados anteriormente y para el método de conjunto. Estas métricas utilizan los cuatro estados posibles de una predicción para una clase para construir una medida del rendimiento del algoritmo. Para una clase determinada, los estados son:

- Verdaderos positivos (VP): cantidad de elementos clasificados correctamente pertenecientes a la clase.
- Falsos positivos (FP): cantidad de elementos clasificados con esa clase de manera incorrecta.
- Verdaderos negativos (VN): cantidad de elementos clasificados correctamente que no pertenecen a esa clase.
- Falsos negativos (FN): cantidad de elementos clasificados incorrectamente como no pertenecientes a esa clase pero que, en realidad, sí que pertenecen a ella.

Estos cuatro estados se obtienen operando sobre la matriz de confusión de los resultados de la clasificación. Esta matriz de $N \; x \; N$, donde N es el número de clases, representa, en sus filas, las clases reales a las que pertenecen los elementos del conjunto de datos y, en sus columnas, las clases predichas para esos mismos elementos.

Una vez explicados los estados de la predicción y la matriz de confusión, se puede continuar presentando las métricas que serán utilizadas en la evaluación de los clasificadores:

- Accuracy (exactitud): es la proporción de las predicciones correctas (verdaderos positivos y verdaderos negativos) sobre el total. Esta métrica se obtiene con el método *getAccuracy* de la clase *ClassificationReport*.

$$Accuracy = \frac{VP + VN}{VP + FP + VN + FN}$$

- Precision (precisión): es la razón de proporción entre la cantidad de elementos clasificados correctamente pertenecientes a una clase determinada (verdaderos positivos) y la cantidad de elementos que han sido clasificados como pertenecientes a esa clase (verdaderos positivos y falsos positivos). Esta métrica se obtiene mediante el método *getPrecision* de la clase *ClassificationReport*.

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

- Recall (exhaustividad): es la razón de proporción entre los elementos clasificados correctamente como pertenecientes a una clase (verdaderos positivos) y el total de elementos que pertenecen a dicha clase (verdaderos positivos y falsos negativos). Esta métrica se obtiene mediante el método *getRecall* de la clase *ClassificationReport*.

$$Recall = \frac{VP}{VP + FN}$$

- F-score (valor-F): esta métrica combina la cobertura y la exhaustividad en una media armónica de manera ponderada a través de un parámetro β, que, frecuentemente, es igual a 1, otorgando el mismo peso a las dos medidas.

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

Debido a que estas métricas de evaluación calculan el rendimiento del clasificador para cada una de las clases, es necesario utilizar una medida que resulte en una media ponderada de cada una de estas métricas que tenga en cuenta a todas las clases del modelo. En nuestro caso se utilizará una media ponderada (weighted average), ya que otras técnicas como macro-averagin no tienen en cuenta la desigualdad entre la cantidad de elementos pertenecientes a cada clase, fenómeno que se da con frecuencia en el modelo. En PHP-ML, para obtener la media ponderada se utiliza el método *getAverage* de la clase *ClassificactionReport*, en cuyo constructor hay que haber incluido previamente la constante *WEIGHTED_AVERAGE* como parámetro.

6.5. Resultados de la evaluación

La evaluación del rendimiento de los clasificadores se realizará utilizando la técnica de la validación cruzada con 10 iteraciones de entrenamiento, predicción y evaluación, cada una de ellas con un conjunto de prueba seleccionado aleatoriamente sobre el conjunto de datos original. El objetivo de esta técnica es garantizar la independencia entre la partición de datos entrenamiento y la de datos de prueba. La división aleatoria se obtiene mediante la clase *RandomSplit* de PHP-ML.

Se ha establecido el tamaño del conjunto de prueba en un 30% de los elementos del conjunto original. Cada una de las iteraciones realiza la evaluación sobre los cuatro clasificadores: KNN, máquina de vectores de soporte, clasificador bayesiano y *ensemble* de los tres anteriores.

La clase que implementa una prueba de validación cruzada es *EvaluationTest.* El controlador *EvaluationController* obtiene el conjunto original de tweets de la base de datos, lo prepara mediante una instancia de la clase *TrainingSet* y ejecuta una prueba de validación cruzada para cada uno de los cuatro clivajes

propuestos. Por último, devuelve los resultados en formato JSON. El script de la API para ejecutar una prueba completa se encuentra en la dirección relativa /api/evaluate.php.

Para evitar que se supere el tamaño máximo de uso de la memoria principal que se establece en la configuración de PHP, se ha decidido que el conjunto de datos sólo esté compuesto de los primeros 425 tweets clasificados. Con este número de muestras, el diccionario de tokens tiene un tamaño de 2697 elementos.

Respecto a las características del conjunto de datos, nos encontramos con que la distribución de clases es muy desigual, clasificándose los elementos mayoritariamente como neutros para todos los clivajes. Por ejemplo, para el clivaje feminismo, hay 348 tweets neutros, un 82% del total. Después, pese a que los elementos favorables y desfavorables son minoritarios, los del primer grupo son más que los del segundo para todos los clivajes. Siguiendo el ejemplo anterior sobre feminismo, los tweets clasificados como favorables son 55 respecto a los 22 desfavorables.

Esta característica puede ser favorable a la hora de aumentar la fiabilidad de los clasificadores cuando se utiliza una medida de frecuencia de las características del tipo TDIDF, ya que unos pocos tokens que aparecen en las muestras clasificadas como favorables o desfavorables son determinantes a la hora de decantar la clasificación si se encuentran en la muestra de la que se quiere predecir su clase.

A continuación, se muestran los resultados de la evaluación para cada uno de los clivajes. Las columnas de las tablas muestran cada uno de los indicadores de media ponderada utilizados y las filas cada uno de los clasificadores:

Tabla 3. Resultados de la evaluación para feminismo.

	Precision	Recall	F1 Score
KNN	85,44%	69,92%	71,41%
SVM	65,75%	81,02%	72,56%
Naive-Bayes	90,25%	90,08%	89,02%
Ensamble	81,80%	86,25%	83,01%

Tabla 4. Resultados de la evaluación para derechos LGTB.

	Precision	Recall	F1 Score
KNN	89,28%	91,56%	88,49%
SVM	80,47%	89,69%	84,82%
Naive-Bayes	93,75%	94,14%	93,33%
Ensamble	89,74%	93,05%	90,86%

Tabla 5. Resultados de la evaluación para migraciones.

	Precision	Recall	F1 Score
KNN	89,37%	87,34%	86,54%
SVM	80,93%	89,92%	85,17%
Naive-Bayes	94,18%	94,06%	93,65%
Ensamble	85,00%	89,45%	86,96%

Tabla 6. Resultados de la evaluación para servicios públicos

	Precision	Recall	F1 Score
KNN	77,31%	66,64%	60,53%
SVM	44,28%	66,40%	53,08%
Naive-Bayes	81,26%	81,33%	80,86%
Ensamble	77,27%	80,08%	78,37%

Como se puede observar resaltado en las tablas, el clasificador bayesiano obtiene los mejores resultados en todos los indicadores y para todos los clivajes de entre los cuatro algoritmos. Son unos resultados excelentes, con una fiabilidad superior al 90% en el caso de feminismo, derechos LGTB y migraciones, y superior al 80% en el caso de servicios públicos.

Contrariamente a lo que cabría esperar, el *ensamble* se establece en segunda posición, con unos resultados peores que Naive-Bayes, debido a que selecciona la clase mayoritaria de entre los tres clasificadores y,

presumiblemente, KNN y la máquina de vectores de soporte suelen clasificar de manera errónea las mismas muestras.

Por otro lado, es evidente que, cuando se atenúa la característica de la distribución irregular de las clases para el clivaje de servicios públicos (los tweets neutros suponen un 67,06% del total frente al 82% de feminismo), el *ensamble* se acerca más a los resultados del clasificador bayesiano y los resultados generales de los cuatro algoritmos son bastante peores, con caídas de hasta 35 puntos porcentuales en el indicador de precisión de la máquina de vectores de soporte.

7. Conclusiones

En este trabajo se ha realizado un recorrido histórico por los fundamentos teóricos de las disciplinas en las que se inscribe la herramienta desarrollada: el procesamiento del lenguaje natural y el análisis de sentimientos, y se han expuesto sus limitaciones y sus metas para el futuro próximo. Asimismo, se ha mostrado la arquitectura básica de una aplicación práctica de dichas disciplinas que puede ser reproducida a día de hoy en un entorno gratuito de software libre y ejecutada en un computador personal de rendimiento medio. Se ha explicado todo el proceso para implementar el sistema: la obtención del conjunto de datos, su preprocesamiento, su clasificación, y su exposición a través de la web, contribuyendo a la ampliación de los trabajos relacionados con la minería de opinión política, mostrando unos resultados aceptables durante la fase de evaluación.

Respecto al PLN y al análisis de sentimientos, tras la profundización en el estado del arte, se puede afirmar que son dos disciplinas que, del mismo modo que el área que las contiene, la inteligencia artificial, han demostrado enormes avances a lo largo de su historia en la consecución de sus objetivos últimos. Sin embargo, estos avances no se han producido de una manera lineal o progresiva, sino que han obedecido a grandes cambios de paradigma que han supuesto saltos discontinuos en su desarrollo. El optimismo de las primeras décadas arropado en la hipótesis de los símbolos físicos pronto se demostró insuficiente para la generación artificial de expresiones en lenguaje natural tal y como las producimos los seres humanos en nuestro día a día.

De este modo, la irrupción de las primeras aplicaciones prácticas que siguen el enfoque subsimbólico (estadístico o conexionista) durante la década de los años noventa, ha supuesto una verdadera revolución en el campo, al proveer de un modelo universal de aprendizaje similar al de los humanos y avalado por las recientes investigaciones en neurociencia. La calidad y la eficiencia de algunas aplicaciones específicas ha experimentado un salto cualitativo evidente, como se puede observar, por ejemplo, con la herramienta de traducción de Google. Sin embargo, aunque en término generales, estos nuevos enfoques establezcan un marco correcto en el que seguir trabajando, también poseen algunas limitaciones que todavía no somos capaces de superar y que relegan por un tiempo el destierro completo de los métodos simbólicos, surgiendo la necesidad de plantear aplicaciones híbridas.

Por otro lado, se hacen evidentes las ventajas y oportunidades que trae el análisis de sentimientos al terreno de la política. Es posible construir herramientas muy refinadas, que sigan las bases establecidas en este trabajo, capaces de presentar el “estado de ánimo” de una sociedad en tiempo real respecto a los conflictos políticos que la atraviesan, o a la valoración sobre las ideas y acciones de partidos, sindicatos y organizaciones de la sociedad civil. La disponibilidad de una fuente de datos continua y directa, como es Twitter, junto con las capacidades de computación actuales y la extensión de internet y los sistemas distribuidos, posibilita que los actores políticos puedan conocer tendencias, puntos de inflexión y nuevos consensos en el momento en que se

están produciendo, obteniendo información más dinámica y complementaria a la que se puede extraer de los clásicos estudios sociológicos de opinión.

7.1. Propuestas de mejora

En cuanto a las características técnicas de la herramienta desarrollada, se han detectado varios puntos susceptibles de mejora o ampliación:

7.1.2.Obtención de datos de entrenamiento

El primero de todos es la mejora del conjunto de datos de entrenamiento, ya que se han incluido tweets en otros idiomas diferentes al castellano, como el inglés, el euskera o el catalán, debido a que algunos de los perfiles escogidos publican mensajes en varios idiomas, error que puede ser corregido refinando los parámetros de consulta a la API REST de Twitter. Además, se debería tener en cuenta el hecho de que muchos tweets no tratan sobre alguno o todos los clivajes políticos tenidos en cuenta, con lo que sería necesario agregar una categoría de clasificación nula fuera del rango de polaridad.

7.1.3.Polaridad y clivajes políticos

Respecto a los valores de polaridad, una versión más refinada de la aplicación podría construirse con una escala numérica continua, aportando mayor información sobre el grado de acuerdo o desacuerdo con un tema determinado. También se podría ampliar el proceso de clasificación a los tipos de emociones que integran el mensaje y no sólo limitarlo a su valoración polarizada.

En relación a esta ampliación del rango de valores de polaridad posibles, se puede proponer la misma idea para los clivajes políticos tratados. El estado actual, con cuatro grandes ideas, es un tanto reduccionista y no ofrece información detallada sobre los conflictos o disputas internas entorno a las posiciones dentro de esos clivajes.

Se simplifican cuatro conceptos muy complejos que podrían subdividirse en componentes más pequeños hasta el nivel de profundidad deseado. Por ejemplo, respecto al concepto de servicios públicos, éste se podría dividir en áreas económicas como sanidad, educación, transporte, energía y otras, o también en función del modelo de propiedad: propiedad pública completa, propiedad pública con gestión privada, propiedad de los trabajadores, modelos de propiedad comunal, etc.

7.1.4.Preprocesamiento de los datos y clasificación

En cuanto a la fase de preprocesamiento de los datos, la complejidad del algoritmo es lineal en función de la cantidad de elementos del conjunto de entrenamiento y del tamaño del diccionario de tokens, lo que hace el tiempo de procesamiento y la memoria consumida sean elevados. Estos costes podrían reducirse imponiendo un tamaño de diccionario fijo con las palabras más significativas o comunes respecto a cada tema, lo que también permitiría

almacenar serializados los algoritmos de clasificación una vez han sido entrenados y no tener que volver a ejecutar el proceso cada vez que se quiere predecir la clase de un conjunto de nuevos tweets. Esto sucede porque, en la actualidad, el diccionario varía de tamaño en función de los nuevos tweets que se quieren clasificar, ya que pueden incluir palabras que no aparecen en el conjunto de entrenamiento.

Otro de los factores que agrandan el diccionario es que el algoritmo de stemming es poco eficaz (algunas palabras con la misma raíz devuelven distintos resultados) y debería revisarse. También deberían mejorarse las reglas de normalización, ya que las expresiones regulares utilizadas en algunos casos no tienen en cuenta todas las posibilidades que deben cubrir. Además, podría ampliarse el proceso de normalización utilizando algunas de las técnicas de sustitución de palabras OOD (out of dicctionary o fuera de diccionario), reduciendo así las diferentes abreviaturas, expresiones de jerga y palabras mal escritas a su correspondiente léxico de diccionario.

En cuanto al proceso de clasificación, se podría añadir a los atributos del conjunto de datos las etiquetas de Parts of Speech y otros derivados de algunas de las técnicas basadas den características lingüísticas del texto.

7.1.5. Arquitectura de la aplicación

Para el desarrollo de la herramienta se escogió el lenguaje PHP por las razones expuestas en el capítulo dedicado a la arquitectura de la aplicación, sin embargo, la biblioteca de clases PHP-ML es el trabajo personal de un desarrollador que todavía no cuenta con una versión estable y que tiene muchas limitaciones respecto a las bibliotecas de aprendizaje automático de otros lenguajes de programación. Por ello, una versión completamente funcional de la aplicación debería ser reescrita con la biblioteca Scikit Learn del lenguaje Python, que es la utilizada mayoritariamente en problemas de machine learning, cuenta con una comunidad de desarrolladores consolidada, y está ampliamente probada y documentada.

Por último, respecto a la parte web para consultar y publicar los resultados de las clasificaciones, podría mantenerse el lenguaje PHP, diseñado desde su inicio con este propósito y utilizado mayoritariamente para ello. Sin embargo, a medida que crezca la funcionalidad podría ser conveniente utilizar algún framework de desarrollo que agilice las tareas de desarrollo que no pertenecen al núcleo de la aplicación, como Symfony, Laravel o Zend.

8. Glosario

A

AJAX. Siglas en inglés de Asynchronous JavaScript And XML. Es una técnica de desarrollo web para mantener una comunicación asíncrona con el servidor desde el cliente. Pág. 18.

Algoritmos de clasificación. Algoritmos de aprendizaje computacional utilizados para predecir la clase a la que pertenece un elemento del conjunto de datos. Pág. 5.

Análisis de sentimientos. Disciplina de la inteligencia artificial encargada del estudio computacional de las opiniones, emociones o sentimientos expresados en texto. Pág. 1.

Apache. Popular software libre de servidor web de la fundación Apache. Pág. 2.

API. Interfaz de programación de aplicaciones o *Aplication Programming Interface* en inglés. Conjunto de clases, funciones o procedimientos que ofrece una biblioteca para ser usados por otro software. Pág. 2.

Aprendizaje automático (véase también aprendizaje computacional o *machine learning*). Disciplina de la inteligencia artificial dedicada al desarrollo de algoritmos de aprendizaje para las computadoras. Pág. 1.

Aprendizaje supervisado. Uno de los tipos de aprendizaje computacional que se caracteriza por tener como objetivo la aproximación de una función a partir de unos datos de entrenamiento de los que se conoce su resultado de antemano. Pág. 1.

B

Bolsa de palabras (bag of words). Método del procesamiento del lenguaje natural apra representar documentos de texto como un conjunto no ordenado de palabras. Pág. 27.

C

Clasificador bayesiano. Algoritmo de aprendizaje supervisado probabilístico basado en el Teorema de Bayes. Pág. 32.

Clivaje. Son las líneas de división o polarización de una sociedad respecto a determinados temas políticos. Pág. 1.

Composer. Software gestor de paquetes y dependencias para el lenguaje PHP. Pág 3.

Controlador. Componente funcional del patrón de arquitectura de aplicaciones modelo-vista-controlador encargado de la lógica de negocio. Pág. 18.

CSS. Siglas de Cascading Style Sheets, hojas de estilos en cascada en español, es un lenguaje de diseño para definir la presentación de un documento estructurado con un lenguaje de marcas. Pág. 18.

E

Endpoint. Punto de conexión de un canal de comunicación. Pág. 18.

Enfoque conexionista. Enfoque de la inteligencia artificial que representa los problemas a alto nivel como una red de unidades de cómputo interconectadas. Pág. 9.

Enfoque estadístico. Enfoque de la inteligencia artificial que utiliza métodos estadísticos para la resolución de problemas. Pág. 9.

Enfoque simbólico. Enfoque de la inteligencia artificial basado en la representación de los problemas a alto nivel como un espacio de estados definido por un conjunto de símbolos físicos sobre los que se ejecutan operaciones de transformación. Pág. 9.

Ensamble. Función compuesta de un conjunto de algoritmos de clasificación. Pág. 31.

Entidad. Clase asociada a una tabla de una base de datos. Pág. 18.

Entorno de desarrollo integrado. Software con utilidades para facilitar el trabajo de desarrollo de aplicaciones informáticas. Pág. 3.

Entrenamiento. Fase del proceso de aprendizaje supervisado en la que un algoritmo es ajustado según un conjunto de datos de ejemplo. Pág. 1.

F

F1Score. Métrica de evaluación de algoritmos de aprendizaje supervisado que combina precisión y recall en una media armónica de manera ponderada. Pág. 34.

H

HTML. Lenguaje de marcas para el desarrollo de páginas de internet. Pág. 4.

J

Java. Lenguaje de programación de propósito general muy popular perteneciente a Oracle. Pág. 3.

JavaScript. Lenguaje de programación usado principalmente en el entorno del cliente para el desarrollo de las interfaces web. Pág. 4.

JSON. Acrónimo de JavaScript Object Notation, es un formato de texto sencillo para el intercambio de datos. Pág. 18.

K

K-vecinos más próximos. Método de aprendizaje supervisado en el que el resultado de la clasificación es la clase del elemento del conjunto de entrenamiento más cercano al objeto del que se quiere predecir la clase. Pág. 31.

L

Lenguaje natural. Lenguaje de propósito general que surge espontáneamente de la actividad comunicativa de los seres humanos. Pág. 1.

Léxico de polaridad. Lista de palabras de un idioma etiquetadas por su polaridad positiva o negativa según el significado emocional que aportan al texto. Pág. 29.

M

Máquina de vectores de soporte. Método de aprendizaje supervisado que utiliza una función llamada kernel para separar los objetos de las diferentes clases en un espacio n-dimensional. Pág. 31.

MariaDB. Sistema gestor de bases de datos relacionales. Pág. 2.

Modelo vista controlador. Patrón de arquitectura de software de tres niveles que se para la integración con datos, la lógica de negocio y la presentación. Pág. 18.

N

Netbeans. Software de entorno de desarrollo integrado de código abierto. Pág. 3.

Normalización. Adaptación léxica del texto de un Tweet al diccionario formal de la lengua utilizada. Pág. 23.

O

OAuth. Estándar abierto para la emisión segura de autorizaciones de comunicación con una API. Pág. 19.

Opinión. Valoración polarizada sobre una entidad referenciada en un documento de texto. Pág. 13.

P

Parts of Speech. Método de selección de atributos mediante el cual se asigna a una palabra su categoría sintáctica en la oración. Pág. 28.

PHP. Lenguaje de programación muy extendido para la creación de sitios web. Pág. 2.

PHP-ML. Biblioteca de clases en lenguaje PHP para realizar tareas relacionadas con el machine learning. Pág. 3.

Precision. Métrica de evaluación de algoritmos de aprendizaje supervisado que es la razón de proporción entre la cantidad de elementos clasificados correctamente pertenecientes a una clase determinada la cantidad de elementos que han sido clasificados como pertenecientes a esa clase. Pág. 33.

Python. Lenguaje de programación muy utilizado para proyectos de minería de datos y de machine learning. Pág. 3.

R

Recall. Métrica de evaluación de algoritmos de aprendizaje supervisado que es la razón de proporción entre los elementos clasificados correctamente como pertenecientes a una clase y el total de elementos que pertenecen a dicha clase. Pág. 34.

Reducción de atributos. Reducción del número de variables del vector de tokens que se utiliza como entrada de los procesos de entrenamiento y predicción de los algoritmos de clasificación. Pág. 26.

REST. Acrónimo de Representational State Transfer, una arquitectura de servicios web sin estado, más ligera que SOAP y en la que se utilizan los verbos del protocolo HTTP para determinar las operaciones a realizar. Pág. 17.

S

Selección de atributos. Proceso de selección del conjunto de características menor que aporten mayor valor de predicción a los algoritmos de clasificación. Pág. 27.

Serialización. Proceso de codificación de un objeto en memoria a un medio de almacenamiento persistente. Pág. 19.

SQL. Lenguaje declarativo estándar para la consultar y creación de bases de datos relacionales. Pág. 4.

Stemming. Proceso por el cual una palabra es sustituida por su raíz eliminando sus afijos. Pág. 27.

Stop Word. Palabra vacía en castellano, es una palabra que no aparto significado a la oración. Pág. 26.

Streaming. Transmisión de mensajes de Twitter en tiempo real y de manera continua. Pág. 17.

Subsimbólico. Paradigma de la inteligencia artificial que aboga por la utilización de métodos en los que se crea una representación simbólica a alto nivel del problema. Pág. 10.

T

TFIDF. Siglas de term frecuency - inverse document frecuency, frecuencia de término – frecuencia inversa de documento en castellano. Es una medida de frecuencia de las palabras de un tweet en el conjunto de datos de entrenamiento. Pág. 28.

Tokenización. Proceso de división del texto de un Tweet en el conjunto de sus palabras y la sustitución de cada una de éstas por un valor no identificable. Pág. 25.

V

Validación cruzada. Técnica de evaluación de los algoritmos de clasificación que divide el conjunto de datos de entrenamiento en k particiones de manera aleatoria y realiza una evaluación k veces, tomando en cada una de ellas un conjunto de prueba diferente para evitar la dependencia entre éste y el conjunto de entrenamiento. Pág. 34.

W

WAMP. Acrónimo de Windoes, Apache, MariaDB y PHP. Es un software que ofrece un entorno de desarrollo basado en estas tecnologías. Pág. 2.

Weigthed average. Media ponderada de las métricas de evaluación de algoritmos de clasificación para el conjunto de clases tratadas. Pág. 34.

9. Bibliografía

1. AKLIWAL, Akshat; FOSTER, Jennifer; VAN-DER-PUIL, Jennifer; O'BRIEN, Ron; TOUNSI, Lamia; HUGHES, Mark. Sentiment analysis of political tweets: Towards an accurate classifier. En: *Proceedings of the Workshop on Language in Social Media*. Atlanta: Association for Computational Linguistics, 2013, pp. 49-58.
2. ANKIT, Nabizath Saleena. An Ensemble Classification System for Twitter Sentiment Analysis. *Procedia Computer Science,* 2018, vol. 132, pp. 937-946.
3. ARCILA CALDERÓN, Carlos; BARREDO IBÁÑEZ, Daniel y CASTRO, Cosette. *Análisis y visualización de datos en Twitter.* Barcelona: Universitat Oberta de Catalunya, 2017.
4. ARCILA-CALDERÓN, Carlos; ORTEGA-MOHEDANO, Félix; JIMÉNEZ-AMORES, Javier y TRULLENQUE, Sofía. Análisis supervisado de sentimientos políticos en español: clasificación en tiempo real de tweets basada en aprendizaje automático. *El profesional de la información*. Septiembre-octubre, 2017, vol. 26, n. 5, pp. 973-982.
5. CARBONELL, Jaime, 1992. El procesamiento del lenguaje natural, tecnología en transición. En: *La lengua española y las nuevas tecnologías. Inteligencia artificial y lengua española. Congreso de la lengua española* [en línea]. Sevilla: Instituto Cervantes [consulta: 03/04/2019]. Disponible en: https://cvc.cervantes.es/obref/congresos/sevilla/tecnologias/ponenc_carbonell.htm
6. GONAWELA, A'Ndre and PAL, Joyojeet. Studying political communication on Twitter: the case for small data. En: *Current Opinion in Behavioral Sciences*. Michal Kosinski y Tara Behrend, octubre de 2017, 18, pp. 97-102.
7. HUTCHINS, Jhon, 2014. The History of Machine Translation in a Nutshell. En: hutchinsweb [en línea]. Disponible en: http://www.hutchinsweb.me.uk/Nutshell-2014.pdf [consulta: 15/03/2019]
8. LAGE GARCÍA, Lola. *Herramienta para el análisis de la opinión en tweets periodísticos*. Barcelona: Universidad Pompeu Fabra, 2014.
9. LEE, Lillian; PANG, Bo y VAITHYANATHAN, Shivakumar. Thumbs Up? Sentiment Classification Using Machine Learning Techniques. *Proceedings of the Conference on Empirical Methods in Natural Languaje Processing*. Philadelphia: Association for Computational Linguistics, julio de 2002, pp. 79-86.
10. LIDDY, Elizabeth D. Natural Language Processing. *Encyclopedia of Library and Information Science*, Segunda edición, Nueva York, Ed. Marcel Decker Inc.
11. MASIP I RODÓ, David; MORENO I RIBAS, Antonio y TORRA I REVENTÓS, Vicenç. *Aprendizaje computacional.* Barcelona: Universitat Oberta de Catalunya, 2013.
12. PANG, Bo y LEE, Lillian, 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*. Boston: Now, vol. 2, no. 1-2, pp. 1-135.
13. RUDER, Sebastian, 2019. The 4 biggest problems in NLP. En: ruder.io [en línea]. Disponible en: http://ruder.io/4-biggest-open-problems-in-nlp/ [consulta: 1 abril 2019].

14. SOBRINO SANDE, José Carlos. *Análisis de sentimientos en Twitter* [en linea]. Universitat Oberta de Catalunya, 2018. Actualizado: 27 de junio de 2018 [consulta: 28 febrero 2019]. Formato en pdf, 2,31 MB. Disponible en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/81435/6/jsobrinosTFM0618memoria.pdf

10. ANEXOS

10.1. Manual de instalación de la aplicación

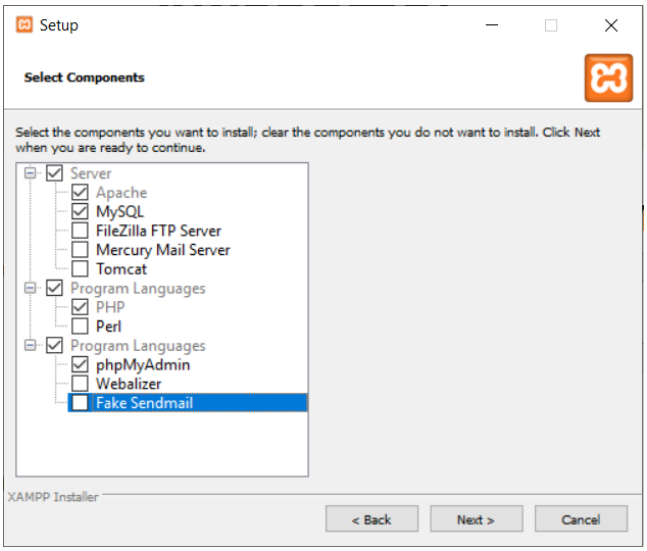
10.1.1. Descargar e instalar XAMPP

XAMPP es un paquete de software que ofrece en una única instalación todas las herramientas necesarias para el funcionamiento de una aplicación web. Con XAMPP quedarán instalados en el equipo un servidor web Apache, una base de datos MariaDB con el gestor web phpMyAdmin y el lenguaje PHP. XAMPP se puede instalar tanto para sistemas Windows como para sistemas Linux. Para descargarlo se debe acceder al sitio oficial de descarga:

https://www.apachefriends.org/download.html

Seleccionamos la opción que corresponda con nuestro sistema e instalamos el software siguiendo las indicaciones del instalador descargado.

En la ventana de selección de componentes a instalar, deberemos desactivar aquellos que no vayamos a necesitar, dejando marcados únicamente los siguientes:

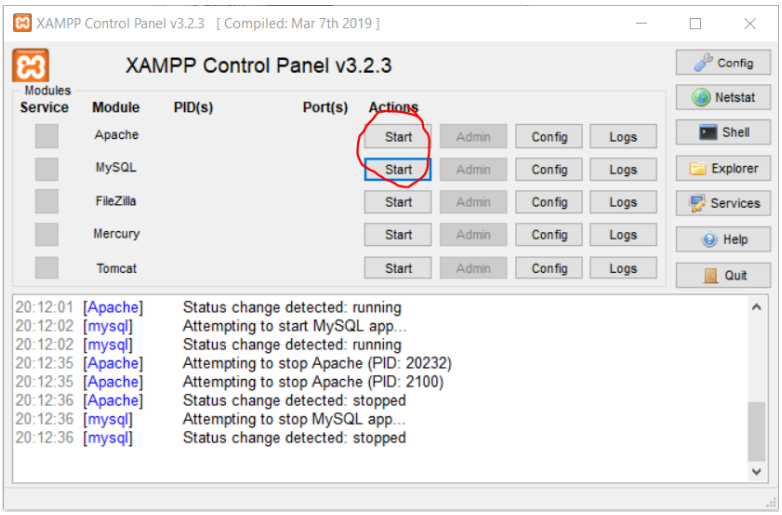


10.1.2. Instalar la aplicación

Para instalar la aplicación, debemos copiar la carpeta raíz del código fuente en la ubicación /htdocs de la carpeta de instalación de XAMPP.

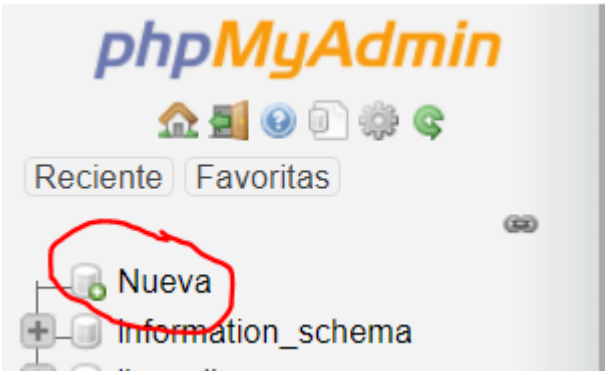
10.1.3. Instalar la base de datos

EL primer paso es iniciar el servicio de Apache y de MariaDB de la aplicación XAMPP. Para ello ejecutamos XAMPP Control Panel y hacemos click sobre los botones Start de dichos servicios:



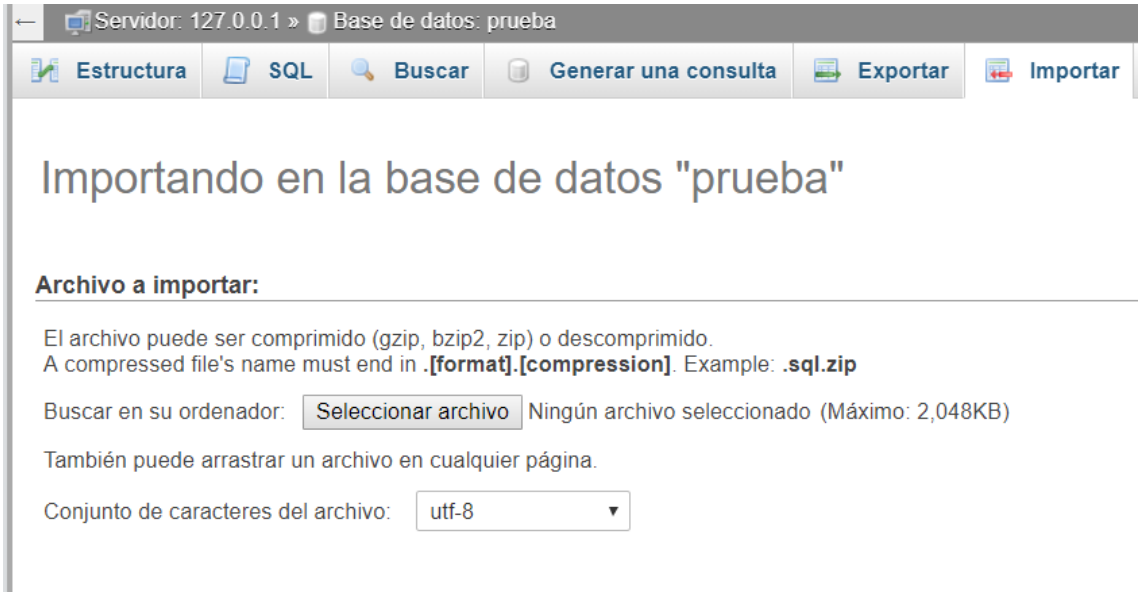
Una vez iniciados, entramos a la herramienta phpMyAdmin accediendo a la dirección “localhost/phpMyAdmin” en nuestro navegador de internet.

Una vez dentro de la herramienta, vamos a crear una nueva base de datos haciendo click en el enlace “Nueva” del menú de la barra lateral izquierda:



Una vez accedemos a la pantalla de creación, elegimos el nombre que queramos para la base de datos, seleccionamos el juego de caracteres utf8mb_unicode_ci y pulsamos en el botón “Crear”:

Una vez creada la base de datos, debemos importar las tablas y los datos del conjunto de datos de entrenamiento. Para ello, accedemos a la nueva base de datos creada haciendo click en el link con su nombre en el menú de la barra lateral y después seleccionamos la opción “Import” del menú de la barra superior. Seleccionamos el fichero database.sql del proyecto, el juego de caracteres utf8, hacemos click en el botón “Continuar” y esperamos a que finalice la importación. Nuestra base de datos ya está lista.



Ahora, para que la aplicación se pueda conectar a la base de datos hay que configurar algunas constantes en el fichero Config.php de la aplicación, que se encuentra dentro de la carpeta /api del código.

```php
<?php

define("DB_HOST", "localhost");
define("DB_NAME", "");
define("DB_USER", "root");
define("DB_PASS", "");
define("DS",DIRECTORY_SEPARATOR);
define("API_PATH",realpath(__DIR__).DS);

define("NUM_TWEETS_PER_PAGE", 50);
```

Se debe definir la constate DB_NAME con el nombre que hayáis dado a la base de datos y la constante DB_PASS con la contraseña que hayáis seleccionado para MariaDB, por defecto este campo debe quedar vacío, ya que XAMPP instala el servicio sin contraseña.

10.1.4. Instalar Composer y los paquetes externos

Para instalar las dependencias que utiliza el proyecto, primero se debe descargar e instalar el software Composer desde su sitio oficial:

https://getcomposer.org/Composer-Setup.exe

Se debe instalar siguiendo las instrucciones del instalador. Una vez instalado, debemos abrir la consola del sistema y situarnos en el directorio raíz de la aplicación dentro de la carpeta /htdocs del directorio de instalación de XAMPP. Ahí debemos ejecutar la orden que buscará e instalará las dependencias necesarias para el proyecto siguiendo el fichero composer.json:

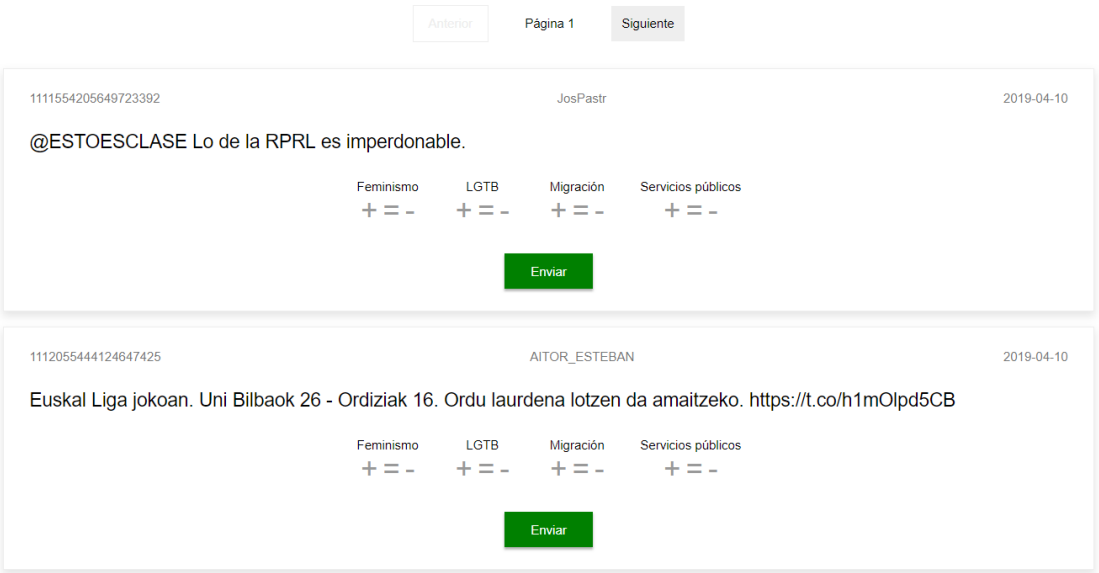
```
php composer.phar install
```

Una vez realizado este paso, la aplicación ya está lista para ser utilizada.

10.1.5. Uso de la aplicación

Para iniciar el módulo de clasificación manual de tweets de entrenamiento, debemos acceder a la siguiente dirección en nuestro navegador de internet:

`localhost/`*`nombre de la carpeta raíz de la aplicación`*`/manual_classification.html`



Cada tweet para clasificar aparece en un cuadro con el texto de este y otra información como el autor, la fecha de recolección o el identificador. Debajo del texto, aparecen las opciones para clasificar cada uno de los cuatro temas políticos y el botón para guardar la clasificación en la base de datos.

Antes de poder comenzar a visualizar la clasificación de los tweets, debemos poner en marcha el servicio de consumo del API Streaming de Twitter para obtener tweets en tiempo real. Para ello, desde la consola, primero debemos iniciar el servidor web interno de php sobre la carpeta en la que se encuentra instalada la aplicación con la siguiente orden:

```
php -S localhost:8000
```

Después, debemos iniciar el servicio de consumo de la API Streaming de Twitter accediendo a la ruta /api/stream de la carpeta de instalación de la aplicación y ejecutando la orden siguiente:

```
php FilterTrackConsumer.php
```

Con esto, el servicio quedará iniciado permanentemente, a no ser que paremos el servidor web interno de PHP.

Para iniciar el módulo principal de la aplicación debemos acceder a la siguiente dirección en el navegador de internet:

`localhost/`*`nombre de la carpeta raíz de la aplicación`*

Para comenzar a obtener clasificaciones de tweets en tiempo real debemos hacer click en el botón “Comenzar streaming”.

Los primeros resultados aparecerán tras unos minutos de carga:


cleavage.io
Favorable
Neutro
Desfavorable
Feminismo
21%
79%
0%
Derechos LGTB
15%
85%
0%
Inmigración
6%
94%
0%
Servicios públicos
44%
56.00000000000001%
0%