Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Análisis y Visualización de Datos Masivos

Análisis del Sentimiento: Exploración de *League of Legends* y su comunidad en Twitter

Trabajo Fin de Máster

Tipo de trabajo:

Presentado por: Dávalos Miranda José Ricardo

Director/a: Ricardo Serafín Alonso Rincón

Resumen

En el presente trabajo se realiza una exploración acerca la toxicidad de la comunidad de *League of Legends* en la red social Twitter. En el estado del arte se observa cómo los autores de distintos estudios implementan técnicas de preprocesamiento de información que incluyen: eliminar instancias duplicadas, nombres de usuarios e hipervínculos. No obstante, los modelos de clasificación que los autores proponen no contextualizan el escenario de estudio. Es decir, no son capaces de identificar elementos clave del contexto para una clasificación precisa del sentimiento. El principal objetivo se centra en la implementación de un modelo de recolección y limpieza de datos con el fin de alimentar dos clasificadores del sentimiento. El primero es un modelo prediseñado e introducido en el 2014 mientras que el segundo es construido manualmente mediante el algoritmo de clasificación supervisada SVM. Para lograrlo, se ejecutan diversas tareas constituidas en tres fases principales: Adquisición, Almacenamiento y Análisis de datos utilizando herramientas de uso libre como el API de Twitter y el lenguaje de programación Python. Finalmente, se logró implementar y comparar los modelos de clasificación y los resultados obtenidos mostraron que el modelo prediseñado no revelaba una clara diferencia de polaridad del texto entre las categorías. Por otra parte, el modelo construido manualmente consiguió alcanzar una precisión del 65.08%. En conclusión, se determinó que el clasificador prediseñado no tiene las capacidades de discernir contenido del videojuego. No obstante, el nuevo modelo fue capaz de distinguir los elementos clave tras proporcionar nuevas reglas de clasificación y permitió identificar los rasgos de un comportamiento tóxico en la comunidad.

Palabras Clave: Twitter, Toxicidad, Análisis del Sentimiento, Aprendizaje de Maquina, League of Legends, SVM, VADER.

Abstract

This paper explores the toxicity of the League of Legends community in the social network Twitter. In the state of the art, it is observed how the authors of different studies implement information preprocessing techniques that include: eliminating duplicate instances, usernames and hyperlinks. However, the classification models proposed by the authors do not contextualize the study scenario. That is, they are not able to identify key elements of the context for accurate sentiment classification. The main objective focuses on the implementation of a data collection and cleaning model to feed two sentiment classifiers. The first one is a predesigned model introduced in 2014 while the second one is manually built using the SVM supervised classification algorithm. To achieve this, several tasks constituted in three main phases are executed: Acquisition, Storage and Analysis of data using free to use tools such as the Twitter API and the Python programming language. Finally, it was possible to implement and compare the classification models and the results obtained showed that the predesigned model did not reveal a clear difference in text polarity between the categories. On the other hand, the manually constructed model was able to achieve an accuracy of 65.08%. In conclusion, it was determined that the predesigned classifier does not have the capabilities to discern video game content. However, the new model was able to distinguish the key elements after providing new classification rules and was able to identify the traits of toxic behavior in the community.

Keywords: Twitter, Toxicity, Sentiment Analysis, Machine Learning, League of Legends, SVM, VADER.

Índice de contenidos

Índice de tablas

Índice de Figuras

Índice de Listados

1. Introducción

Los videojuegos se remontan hacia el año 1962, cuando un estudiante en el instituto de tecnología de Massachussets crea un juego llamado *Space Wars*, el cual llevaba al límite las capacidades computacionales de un ordenador del tamaño del laboratorio del instituto (Glancey, 1996, p.6). Estos han evolucionado notablemente demostrando una alta presencia como parte del mundo tecnológico en los últimos años. Lo que empezó siendo dos barritas a los costados de una pantalla, prosperó y se convirtió en una industria atractiva para grandes compañías como Microsoft y Sony. Para el final de los 90, la industria alcanzaba una suma de $5 millardos en los Estados Unidos ocasionando que Hollywood se interese en producir películas basadas en los videojuegos (Kent, 2021, p.9).

Dentro del amplio espectro de géneros de videojuegos, los pertenecientes a MOBA se han llevado la atención de muchos jugadores debido a sus complejas mecánicas y competitividad. En este complejo sistema de entretenimiento, aparecen aquellos personajes que deciden tomar acciones por su propia cuenta, arruinando la experiencia para otros jugadores mediante acciones que resultan en una derrota e insultos. Este comportamiento de agresión hacia los demás se denomina toxicidad e impone un reto para los diseñadores de videojuegos y que ocasiona la pérdida de jugadores y evita la llegada de nuevos (Martens te al, 2016, p.1).

Tanto jugadores como compañías han invertido recursos para prevenir el comportamiento tóxico, sin embargo, no se han logrado mayores resultados. Kowert (2020) explica que hay varios factores, tanto externos como internos, que se deben tomar en cuenta al momento de discutir sobre toxicidad. Entre ellos se puntualiza que los individuos aprenden sobre comportamientos tóxicos mediante una cultura tóxica pre-existente. También menciona que los individuos adoptan tales comportamientos debido a que no existe una consecuencia aparente, pues, gracias a la sensación de anonimato presente en las interacciones en línea, se genera un espacio ideal para promover dicho comportamiento e inclusive extenderlo a otros medios digitales.

El presente estudio se enfoca en *League of Legends*, un juego desarrollado por *Riot Games*. Un juego que actualmente cuenta con un promedio de 115 millones de jugadores activos mensualmente (Liu, 2021). En el estudio realizado por Watson (2015), relata cómo apenas en los primeros minutos del juego existe evidencia de comportamientos tóxicos. Al mismo tiempo, la compañía ha creado una cuenta oficial del juego en la plataforma Twitter (@LeagueOfLegends), la cual tiene 5.1 millones de seguidores al momento de desarrollo del

presente documento (*League Of Legends,* s.f). Interesantemente, Laroia et al. (2019) relatan en su reporte sobre el estado de Twitter que existe comportamiento tóxico dentro de la plataforma y, lo que es peor aún, no existen mayores esfuerzos por parte de la empresa por reducir tales actividades.

Debido al alto volumen de datos, identificar y extraer información sobre el sentimiento es una tarea difícil. Los científicos de datos han realizado investigaciones y desarrollado técnicas que permiten detectar la polaridad del texto. Algunos de ellos se incluyen en la Sección 2.2 Estado del Arte, y se observa el alto potencial que tiene el análisis de sentimiento utilizando el procesamiento del lenguaje natural. Gupta et al. (2017) proporcionan una lista de técnicas utilizadas a modo de pre-procesamiento de datos, que facilitan la interpretación del texto a nivel de máquina. Entre estas técnicas se encuentra la normalización del texto a solo minúsculas, eliminación de palabras sin un aporte al significado relevante o que aportan a la polaridad del texto, eliminación de elementos ruidosos como por ejemplo hipervínculos y referencias a otras cuentas de usuarios, etc. Por otro lado, Kalaivani y Dinesh (2020) proponen un sistema compuesto de varias fases que les permite realizar transformaciones de texto y eliminación de palabras.

A partir de la investigación realizada se propone un modelo de recolección, limpieza y análisis de datos que haga uso de técnicas de procesamiento y normalización de datos con el fin de utilizar un modelo de clasificación prediseñado y compararlo con un nuevo modelo entrenado bajo nuevas reglas de clasificación para validar su efectividad. El trabajo consiste en tres fases principales en las que primero se adquieren los datos mediante la ayuda de Python y el Twitter API. Utilizando el mismo lenguaje de programación, luego, se almacenan los metadatos de *tweets* en una base de datos NoSQL llamada MongoDB y finalmente, se procede a realizar el análisis de datos utilizando técnicas de exploración y elaboración de visualizaciones.

En conclusión, se puede afirmar que los objetivos planteados fueron alcanzados exitosamente. La investigación realizada facilitó el entendimiento de la temática como también de las técnicas de tratamiento de los datos utilizadas por varios autores. Se logró implementar el modelo COSA-DLC para la adquisición, almacenamiento y análisis de la información de tal forma que es clara y evidente el flujo de los datos. En cuanto a la última fase, se puede concluir que se logró analizar la interactividad de la comunidad y obtuvo un claro entendimiento de la naturaleza de los datos recolectados. Luego, las técnicas de limpieza y pre-procesamiento de datos permitieron obtener un conjunto de datos sin impurezas ni ruidos innecesarios que facilitan el entendimiento de la polaridad del texto para los clasificadores. Finalmente, se evaluó la usabilidad de un modelo de clasificación prediseñado y se encontró que no era capaz

de realizar una clasificación precisa a causa de que no consideraba elementos característicos de contenido del juego *League Of Legends*. Con el primer modelo se encontraron palabras repetidas a lo largo de las tres clasificaciones, por lo tanto, se identificó que un nuevo modelo era necesario. El nuevo modelo fue capaz de clasificar como neutrales a las palabras que son referencia directa del juego, y resaltó el comportamiento tóxico en las interacciones de la comunidad.

1.1 Justificación

Por lo planteado anteriormente, identificar el comportamiento tóxico en los medios digitales es una tarea que requiere de un alto esfuerzo para ser controlado, y luego, mitigado. Las principales causas se puntualizan en: una cultura tóxica pre-existente, la normalización de comportamiento tóxico y el anonimato del internet. El resultado de los comportamientos tóxicos resulta en daños a la salud y bienestar de las personas, y adicionalmente, arruinan la experiencia de uso de la plataforma. A partir del análisis de texto, es posible identificar la razón por las cuales dicha comunidad se comporta de tal manera, y permite que los dirigentes puedan tomar decisiones sobre la dirección de cada plataforma.

Se realizó una búsqueda de las siguientes palabras claves: análisis del sentimiento, procesamiento de lenguaje natural, Twitter, toxicidad en *League Of Legends,* clasificación de texto con sus sinónimos en inglés. Esta búsqueda se realizó en los principales indexadores de trabajos académicos: *Google Scholar* e *IEEE* en los cuales no se encontró ningún estudio semejante al propuesto. A continuación, se procedió a extraer las referencias bibliográficas de los estudios encontrados generando un muestreo de bola de nieve. Este permite expandir el conocimiento hacia los predecesores de estos estudios para profundizar en la historia de estos temas.

Existen trabajos como los mencionados en el estado del arte que abarcan la recolección de datos desde Twitter, la limpieza y el uso de clasificadores. Pero no abarcan excepciones de contexto como temáticas de juegos que pueden generar un sesgo en el análisis. El clasificador generado en este estudio puede ser reutilizado por otros miembros de la comunidad que necesiten analizar cuentas y excluir lenguaje coloquial o frases comunes que se considera carecen de significado ya que solo se refieren a contenido especifico al entorno, en este caso el juego.

1.2 Planteamiento del trabajo

Una vez entendidos los conceptos tratados anteriormente, el presente trabajo plantea elaborar un modelo de recolección de información que permita el estudio del texto generado por la comunidad de *League of Legends* en la plataforma Twitter. Para lograrlo se utiliza el Twitter API, el cual permite realizar consultas y obtener una lista de *tweets* que son almacenados posteriormente en una base de datos de MongoDB con el fin de comparar y elaborar dos modelos de clasificación de sentimientos. El primer modelo no cuenta con una contextualización sobre la terminología utilizada en el videojuego, ya que no utiliza ningún algoritmo de *Machine Learning* para clasificar los tweets. VADER funciona mediante un diccionario de palabras para calcular la intensidad y polaridad del texto analizado a través de un análisis por palabra. Esto evita que pueda analizar una palabra en base a un contexto y que su uso no pueda ser generalizado (Calderón, 2017). Mientras que el segundo fue desarrollado para considerar estos elementos. Este modelo se lo construye mediante el algoritmo de clasificación supervisada SVM, el cual, en base a los estudios descritos en el Capítulo 2, otorga los mejores resultados para el análisis de texto. Con esta premisa, se espera encontrar un modelo que permita identificar, en la red social Twitter, el comportamiento tóxico que se encuentra presente en el videojuego hoy en día.

1.3 Estructura de la memoria

En el Capítulo 2 se realiza una breve exploración sobre la historia de videojuegos y la relevancia de analizar esta industria. Luego, se describe el ámbito de las redes sociales y la toxicidad presente en ellas y en los videojuegos. Seguido, se exploran las técnicas de procesamiento de datos y de análisis del procesamiento del lenguaje natural, todo bajo un esquema de contextualización de la problemática y exploración de investigaciones realizadas en dicho ámbito. En el Capítulo 3 se definen los objetivos que se pretenden alcanzar con el desarrollo de este trabajo y se plantea una metodología, dividiendo el trabajo en tareas simples y medibles que deben ser completadas en un orden secuencial. En el Capítulo 4 se manifiesta el desarrollo del trabajo como tal. Se comienza por la adquisición de datos y se describe los puntos clave para cumplir este proceso y cómo se encuentra relacionado con la siguiente fase de almacenamiento de la información. Posteriormente, se realiza un análisis exploratorio de los datos recolectados de forma textual y gráfica. Seguido, se ejecutan tareas de limpieza con el fin de alimentar los dos modelos de clasificación de sentimientos. Por último, en el Capítulo 5 se presentan las conclusiones obtenidas a partir de cada fase, se describen las limitaciones encontradas y se presentan los lineamientos para trabajo futuro y mejora.

2. Contexto y Estado del arte

El siguiente apartado presenta la contextualización de la problemática y el estado del arte. La Sección 2.1 incluye un breve recuento histórico del surgimiento de los videojuegos, las consecuencias del comportamiento tóxico y el rol de las redes sociales en el entorno. Posteriormente, en la Sección 2.2 se detallan las técnicas de análisis del sentimiento.

2.1 Problemática

A continuación, se presenta una breve revisión del origen de los videojuegos y los eventos más importantes. Posteriormente, se describe la toxicidad, se exhiben ejemplos de comportamientos tóxicos y finalmente se presenta la relación entre los videojuegos, el comportamiento tóxico y las redes sociales, enfocándose en Twitter.

2.1.1 Historia de los videojuegos

Las raíces de la industria de los videojuegos se remontan hacia el año 1962. Cuando Steven Russel, un estudiante en el instituto de tecnología de Masachusets, crea un juego llamado *Space War* que consiste en dos 'naves espaciales' enfrentándose en combate, pero en realidad eran dos líneas en un monitor que llevaba al límite las capacidades computacionales de un ordenador que ocupaba completamente el laboratorio del instituto (Glancey, 1996, p. 6). Unos años después en 1972, apareció el primer videojuego comercial creado por Atari llamado *Pong,* dos simples barras en los costados y una pelota en medio, cuyo objetivo era obtener puntos cuando el contrincante perdía la pelota en un lado de la pantalla. Para 1979, los estudiantes de muchas escuelas, que ya conocían acerca los videojuegos comerciales y los famosos cartuchos de Atari, soñaban con tener el Atari 2600 VCS. Una máquina que se enchufaba en la televisión del hogar y considerada problemática por muchos padres (Glancey, 1996, p. 7). Poco después en 1981, la sociedad entre Microsoft e IBM trajo introducción de máquinas personales producidas por IBM con sistema operativo MS-DOS de Microsoft y microprocesadores de Intel (Computer History Museum, s.f.). Esta sociedad permitió producir computadoras a bajo costo y con las capacidades de generar graficas a color ideales para las oficinas, pero también para jugar. Sin embargo, una de estas máquinas, llamada ZX81, inventada por *Clive Sinclair* presentaba problemas de sobrecalentamiento y fue descontinuada al poco tiempo por la llegada de Nintendo y Sega en 1983 (Glancey, 1996, p. 10).

En un inicio los videojuegos eran vistos como un nicho de mercado pequeño, solo para niños, que no merecía la atención de las compañías. No obstante, para el final de los 90, la industria

alcanzaba una suma de $5 millardos en los Estados Unidos ocasionando que Hollywood se interese en producir películas basadas en videojuegos como, por ejemplo: *Street Fighter y Tomb Raider (*Kent, 2021, p. 9*).* Definitivamente, en este periodo se llevó a cabo uno de los mayores avances tecnológicos en la industria con la salida de procesadores para computadora capaces de proyectar diseños en 3D. Al mismo tiempo, con la salida de Windows 95, Microsoft empezó a dirigir su atención hacia la industria ocasionando una avalancha de nuevos videojuegos por diferentes autores (Kent, 2021, p. 21). Avanzando al 2005, las grandes compañías del mercado como Microsoft, Sony y Nintendo definieron una nueva era de videojuegos en alta definición, y poco a poco han implementado mecanismos innovadores para capturar movimiento mediante sensores y proporcionar una forma distinta de jugar videojuegos. Actualmente, tienen su atención en los videojuegos de realidad virtual (History Editors, 2017).

2.1.2 Comportamiento tóxico en los videojuegos

La industria de los videojuegos tiene un amplio espectro de géneros para todos los apasionados. En particular, aquellos de arena de combate multijugador en línea o MOBA por sus siglas en inglés, han recibido una creciente ola de jugadores debido a las complejas mecánicas y naturaleza competitiva que este género ofrece. Los MOBA consisten en partidas independientes con dos equipos generalmente conformados por cinco personas, en los que cada equipo requiere cooperar para vencer las defensas del enemigo y obtener la victoria. No obstante, existen jugadores que deciden tomar su propio camino y actuar según su propio criterio. Esto resulta muchas veces en que dichos jugadores se conviertan en blancos fáciles, lo cual, a su vez, reduce las posibilidades de cumplir el objetivo principal. Al encontrarse este tipo de jugadores, los canales de comunicación; originalmente pensados para coordinar con el equipo, se tornan con el propósito de agredir verbalmente a los demás jugadores. Martens et al. (2016) explica que «La hostilidad percibida en una comunidad de jugadores suele denominarse toxicidad» (p. 1) e impone un reto para los diseñadores de videojuegos ya que puede ocasionar la pérdida de jugadores y evitar la llegada de nuevos.

Jugadores y compañías han invertido distintos recursos para prevenir este tipo de comportamientos. Las compañías agregan diferentes tipos de controles como reportes entre jugadores para detectar y sancionar a aquellos que incurran en estos comportamientos. A pesar de ello, la toxicidad se mantiene presente. Desde una perspectiva más amplia, todo el comportamiento hostil que se efectúa en línea, tanto en los videojuegos como fuera de ellos, puede ser ubicado bajo el concepto de participación oscura o *dark participation*. La Figura 1 muestra cómo todas las acciones verbales y de comportamiento entran bajo el concepto descrito. El resultado de los comportamientos que causan daño a la salud o al bienestar de

otras personas se consideran tóxicos. Es importante mencionar una distinción clave en cuanto a estos conceptos. La participación oscura se refiere a cualquier acción discrepante que tenga lugar en espacios en línea, pero un comportamiento tóxico suele estar definido culturalmente (Kowert, 2020, p. 4). Es decir, en ciertos círculos sociales de la comunidad, jugar de forma que no fue la intención primaria por el desarrollador puede ser considerada tóxica. Por ejemplo, utilizar los canales de comunicación encontrados en el juego para agredir a los demás es considerado un comportamiento tóxico, mientras que terminar el juego lo antes posible, no. Dentro de la misma categoría se encuentran los jugadores que utilizan herramientas externas o errores de programación para obtener una ventaja injusta sobre el resto de los jugadores.

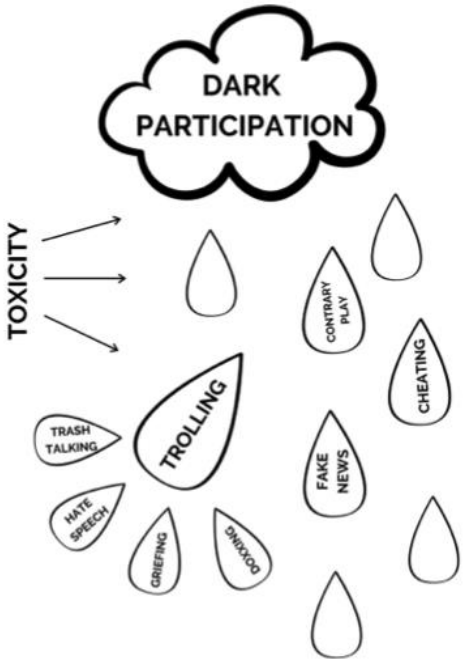

Figura 1: Conceptualización de la participación oscura.

Fuente: Kowert, 2020, p.4

Kowert (2020) indica que «hay varios factores del entorno y de la comunidad para tener en cuenta a la hora de hablar del comportamiento tóxico en los espacios en línea…la teoría cognitiva social, la teoría del comportamiento planificado y el efecto de desinhibición en línea» (p. 2). Puesto en términos sencillos, la primera se refiere a que los individuos aprenden el comportamiento tóxico en los juegos a través de una cultura tóxica pre-existente. La segunda explica que las intenciones de un individuo por adoptar un comportamiento tóxico se basan en el contexto de la situación en específico siempre y cuando sea aceptado como una norma del grupo y no existan consecuencias. Finalmente, el efecto de desinhibición en línea se relaciona con la anterior debido a que las interacciones a través de internet vuelven al individuo invisible. Otros miembros de la comunidad no pueden ver a la persona y por lo tanto

es anónima creando un espacio ideal para promover el comportamiento tóxico sin repercusiones. Lo que es peor, Kowert (2020) menciona que este tipo de comportamiento también se lo encuentra en otros espacios de los medios digitales como por ejemplo en los grupos de Facebook. (p. 2)

Beres et al. (2021) refuerzan lo previamente mencionado por Kowert. Además, lo complementan con su estudio acerca de la normalización de la toxicidad de los juegos en línea, en donde afirman que «normalizar los comportamientos tóxicos en los juegos es problemático porque crea un ciclo de perpetuación recíproca» (p. 2). Esto se refiere a que aquellos individuos que aprueban un tipo de comportamiento en específico serán más propensos a repetirlo. Sobre todo, si la comunidad lo acepta como normal, provocando que este comportamiento se extienda a todos los miembros del grupo con el paso del tiempo.

Este análisis se enfoca en *League of Legends*, un videojuego para computadora catalogado bajo el género MOBA y lanzado al mercado en el año 2009 por la empresa *Riot Games.* Actualmente cuenta con un promedio de 115 millones de jugadores activos mensualmente (Liu, 2021). En poco tiempo, se ha convertido en uno de los juegos más populares del género, sobre todo, debido a que fusiona aspectos de los juegos de rol (RPG) como: personajes con historias de fondo, adquisición de nuevas habilidades, experiencia y objetos a medida que el juego progresa. Existen diferentes maneras de determinar si un juego es una actividad importante para las personas, entre ellos se encuentran el juego, la música y la guerra. Los cuales son aspectos que le añaden un grado de seriedad volviéndolos significativos para quienes lo juegan. En su estudio, Watson (2015) relata su propia experiencia dentro del juego y como en apenas los primeros minutos existen comentarios inadecuados y con un tono agresivo provocando una mala experiencia para todos los jugadores (p. 226). Esto se debe a la seriedad que toma el juego por la importancia que los jugadores le entregan.

Parte de esta comunidad sigue a la cuenta oficial de *League of Legends (@LeagueOfLegends)* en la plataforma de Twitter la cual tiene 5.1 millones de seguidores al momento de escritura de este documento. Kasfia (2018) describe a Twitter como una red social creada en 2006 que se hizo muy popular por la falta de restricciones y la frecuencia de las publicaciones. Las publicaciones se llaman *tweets* las cuales tienen una longitud máxima de 140 caracteres (p. 5). Laroia et al. (2019) en su reporte sobre el estado de Twitter menciona que a pesar de las acciones tomadas por su grupo interno *Change The Terms*, la cual tiene como objetivo animar a individuos a que se posicionen en contra de actividades que provocan que la plataforma se vuelva en un lugar peligroso y tóxico para todos, no existe un compromiso por reducir o mitigar el comportamiento tóxico en la plataforma. A diferencia de otras redes

sociales, Twitter explica claramente las reglas y políticas de uso de la plataforma en su sitio web. Sin embargo, permite actividades, personas e instituciones que incitan al odio (p. 5).

El presente estudio tiene como intención examinar la medida en la cual los jugadores de *League of Legends* extienden su comportamiento tóxico en el ámbito social de Twitter. Dado que ambos entornos digitales conllevan la categoría de comunidad tóxica, se plantea la hipótesis de que los jugadores que son tóxicos dentro del juego también lo serán en Twitter. Para ello, se plantea un modelo de recolección de datos y técnicas de análisis de sentimiento que se describen en la siguiente sección.

2.2 Estado del Arte

La era del internet ha permitido que las personas tengan la capacidad de expresar sus opiniones de forma rápida y sencilla. Las redes sociales generan un alto volumen de datos con contenido sentimental plasmado en *tweets*, actualizaciones de estado, publicaciones en blogs, comentarios, reseñas, etc. El contenido generado por los usuarios es una fuente de opiniones valiosa en contextos que requieran el entendimiento de la opinión pública sobre un concepto. Giachanou et al. (2016) explican que uno de los casos más comunes se enfoca en cómo las empresas utilizan esa información para capturar la opinión de sus clientes sobre sus productos o la de la competencia. Luego, la misma información obtenida puede ser utilizada para mejorar la calidad del producto o servicio (p.1). Otro ejemplo se lo describe desde el punto de vista del cliente, donde el usuario utiliza la información que existe sobre un producto para decidir si comprarlo o no (p. 2).

Debido al alto volumen de datos generados por los usuarios y compartidos por el internet, la extracción de opiniones y sentimiento se vuelve una tarea difícil. La información sobre la opinión se encuentra oculta en los datos, tornando casi imposible que una persona busque y extraiga información útil entre las distintas fuentes. Por tal motivo, los científicos han realizado investigaciones y desarrollado técnicas que permitan detectar automáticamente la polaridad del texto y extraer información eficazmente. A raíz de estas investigaciones aparecen dos áreas de estudio: minería de opinión (MO) y el análisis del sentimiento (AS). La primera tiene como objetivo determinar si el texto en cuestión tiene opinión, mientras que la segunda detecta la polaridad del sentimiento expresado a través del texto y le asigna un sentimiento positivo, negativo o neutral (Giachanou, p. 2). Formalizando, Kharde et al. (2016) definen al análisis del sentimiento como «el proceso de automatizar la minería de actitudes, opiniones, puntos de vista y emociones de texto, discursos, *tweets* y fuentes de bases de datos a través de procesamiento del lenguaje natural» (p. 5).

El análisis del sentimiento en Twitter (AST) pretende analizar el texto de los *tweets* en términos del sentimiento que estos expresan (p. 6). Giachanou et al (2016) explican que en la mayoría de los métodos de análisis del sentimiento en Twitter se utiliza un clasificador de *Machine Learning* (ML) y que el primer paso involucra la recolección de *tweets* (p. 3). Para lograrlo, se considera que Twitter cuenta con una versión libre de su API (Aplicación Programming Interface) que facilita el proceso de recolección de *tweets*. Sin embargo, tiene algunas limitaciones. Por ejemplo, la API libre de Twitter limita a una extracción de 500,000 *tweets* por mes. Adicionalmente, limita el acceso a algunas funciones de recuperación de *tweets,* tal como, la búsqueda de *tweets* en todo el repositorio. Con la versión gratuita de la API, es posible realizar una búsqueda de hasta 7 días previos (Twitter Developer Platform, 2022). Una descripción comparativa de las diferencias entre versiones puede ser encontrada en el siguiente enlace: Pricing – Twitter Developers[1]. Para este estudio en particular, las limitaciones mencionadas anteriormente no representan un impedimento para el desarrollo exitoso. Se toma en cuenta las condiciones de uso de la versión gratuita de la API y se propone un modelo de recolección que permite la persistencia de los datos. A continuación, se relata las técnicas utilizadas por diferentes autores para abordar este tema y los resultados obtenidos.

Gupta et al. (2017) detallan en su estudio sobre el 'Análisis del Sentimiento en Twitter utilizando algoritmos de *Machine Learning'* que el paso más importante es el pre-procesamiento de los *tweets*. Los pasos involucrados en este proceso tienen como objetivo que los datos se encuentren estructurados, simplificados y sin duplicidad lo que facilita la interpretación a nivel de máquina y reduce la ambigüedad en la extracción de características. Entre las tareas utilizadas para el pre-procesamiento de *tweets* se encuentra:

- Eliminar *tweets* retransmitidos ya que son copias de un *tweet* escrito previamente por otro autor.
- Conversión de mayúsculas a minúsculas con el fin de evitar que dos ocurrencias de una misma palabra se consideren diferentes a causa de cómo se encuentra escrita.
- Eliminar *stopwords* o palabras que no aportan a la polaridad del texto como por ejemplo los artículos (a, el, la).
- Eliminar características del *tweet*. Los nombres de usuarios e hipervínculos insertados en el texto no son relevantes desde el punto de vista de análisis ya que no aportan con sentimiento o información importante para el estudio.

[1] Tabla Comparativa Versiones Twitter API: https://developer.twitter.com/en/pricing/search-fullarchive

- *Stemming:* Sustituir las palabras por sus raíces. Esto ayuda a reducir la dimensionalidad del conjunto de características.
- Eliminar caracteres especiales y dígitos ya que no transmiten ningún sentimiento y a veces se mezclan con las palabras.

Posteriormente, los autores exploraron diferentes clasificadores de sentimientos proporcionados por la herramienta scikit-learn, una librería usualmente utilizada en el lenguaje de programación Python. El conjunto de datos obtenido desde la API de Twitter fue dividido en un subconjunto de pruebas (70%) y otro de validación (30%) con el fin de entrenar los modelos de clasificación. Los resultados demuestran un alto valor de precisión, superior al 87%, para los modelos de redes neuronales, arboles de clasificación y máxima entropía. Estos resultados demuestran que a medida que la cantidad de clases a clasificar incrementa, menor será el desempeño del modelo de clasificación (p. 30 - 33). Es decir, un modelo de clasificación binario (Ej.: positivo y negativos) tendrá una mejor eficiencia al compararlo con uno no binario (Ej.: positivo, negativo y neutral).

Ranganathan y Tzacheva (2019) en su trabajo proponen un enfoque para detectar automáticamente las emociones en los mensajes de Twitter y explican cómo la minería de emociones ha sido un tópico de interés en los últimos años. La información que se puede extraer es lo suficientemente amplia para realizar sugerencias sobre acciones que ayudan a que las emociones de los usuarios sean más positivas. Este estudio se dividió en cinco fases: recolección de datos, pre-procesamiento y anotación de emociones, extracción de características, clasificación y descubrimiento de patrones procesables. En la primera, se recolectó un total de 520 000 *tweets* desde la API de Twitter que fueron filtrados por el lenguaje inglés. En la segunda fase, se aplicaron tareas de limpieza de datos en las que se convirtieron todos los tweets de mayúsculas a minúsculas, eliminaron palabras sin aporte a la polaridad del texto y se reemplazaron palabras de la jerga coloquial por un texto formalizado. En cuanto a la anotación de las emociones, los autores recurrieron a utilizar el National Research Council - NRC Emotion lexicón, un diccionario de palabras y su asociación con ocho emociones básicas mediante una ponderación calculada en base al sentido de la palabras, *hashtags* y emoticones. Para la tercera fase, se exploró un análisis del sentimiento basado en uni-gramas, bi-gramas, n-gramas y partes de oraciones. Es decir, se analizaron las palabras individualmente y poco a poco se expandió el análisis a pares de palabras y posteriormente a conjuntos de palabras u oraciones que permiten extraer el sentimiento del texto. Para la cuarta fase utilizaron diferentes modelos de clasificación supervisada en la que dividen al conjunto de datos en pruebas y validación, con el fin de posteriormente utilizar métricas de rendimiento sobre los modelos. En este estudio se enfocaron en emplear un modelo de clasificación SVM. En su última fase, realizaron un descubrimiento por patrones de acción que permiten describir

la transición de un estado a otro más deseable. Por ejemplo, uno de los patrones encontrados sugiere que para alcanzar un estado de 'felicidad' se requiere del uso de palabras positivas y minimizar las negativas. Finalmente, los autores presentan sus resultados de un modelo de clasificación supervisado con una precisión promedio de 98% (p. 61 - 65).

Anupama et al. (2020) en su estudio sobre el 'Análisis del sentimiento en tiempo real utilizando el Procesamiento de Lenguaje Natural' describen la existencia de un alto potencial para descubrir y analizar patrones interesantes a partir de las redes sociales, con aplicaciones orientadas a los negocios. Por ejemplo, los individuos publican sus reseñas acerca de una película que generan una discusión a gran escala y constituyen una base para que otros individuos evalúen y reflexionen no solo a nivel de películas si no de productos, si será un éxito o no. El enfoque principal de este estudio fue predecir la opinión de las masas expresados en *tweets* utilizando un modelo de clasificación de Naive Bayes y el procesamiento del lenguaje natural. Durante la ejecución de este estudio, se efectuaron tareas similares a las expuestas por los anteriores autores entre las que se incluyen la recolección de datos, la limpieza, pre-procesamiento, el entrenamiento y validación del modelo de clasificación. Los resultados del estudio demuestran que lograron una buena precisión del modelo utilizando Naive Bayes, sin embargo, se limitaron a obtener buenos resultados basándose en el análisis de las palabras individuales y proponen mejoras para su modelo a modo de trabajo futuro como analizar pares o conjuntos de palabras (p. 1107 - 1110).

Kalaivani y Dinesh (2020) afirman que el análisis automático de las redes sociales ayuda a analizar los sentimientos y proporcionan un informe imparcial sobre la emoción real de la gente. Este estudio tiene un enfoque político en el que su objetivo es analizar la respuesta emocional que existe en los individuos al recibir comunicados por parte del gobierno. El sistema propuesto por los autores (Figura 2) consiste en una fase de limpieza en la que se realiza varias transformaciones de texto y eliminación de palabras. Luego, se realiza la división del conjunto de datos en un sub-conjunto de pruebas y otro de validación. Posteriormente se elaboraron dos modelos de clasificación un SVM y otro ANN. La arquitectura de la red neuronal utilizada consiste en una capa de entrada y salida, y tres capas ocultas. A diferencia de los autores mencionados en este estudio, Kalaivani y Dinesh, tomaron un conjunto de *tweets* de la plataforma *Kaggle* el cual filtraron para obtener 11 719 instancias válidas para su análisis. Finalmente, los resultados de su estudio demostraron que el SVM superó a la red neuronal con una precisión del 69.24%. Los autores argumentan que el conjunto de datos no se encontraba balanceado y, sugieren que se debe realizar un mayor esfuerzo por ampliar el conjunto de datos y validar el resultado del estudio.

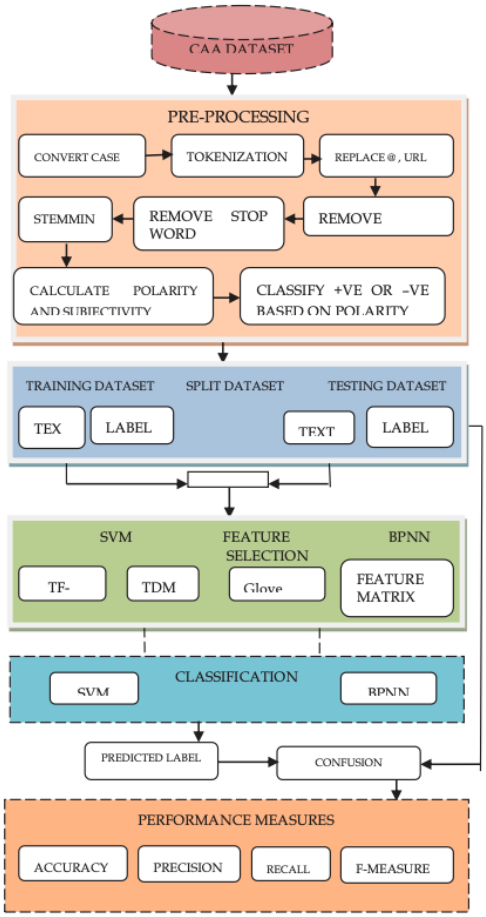

Figura 2: Metodología de Limpieza y clasificación de tweets.

Fuente: Kalaivani, 2020, p. 109

Neha et al. (2021) coinciden con los autores, Ranganathan y Tzacheva, en que el análisis de sentimiento es un método automático de leer y organizar texto en tres categorías (positivo, negativo y neutral). Sin embargo, amplían esta definición señalando la importancia que hoy en día tiene el monitoreo de la actividad de los usuarios en Twitter. Desde la perspectiva empresarial, permite poner en consideración los clientes de la organización, mantenerse al día con los productos, la competencia y a identificar posibles mercados emergentes. Para cumplir con la definición expuesta, los autores de este estudio decidieron utilizar una red neuronal convolucional ya que una de las características de las redes neuronales es que aprenden de sus propios errores. Su metodología se divide en cuatro fases consistentes con las de estudios anteriores. Inician con la fase de recolección de datos, en la que se registraron con una cuenta de desarrollador en el portal de Twitter para tener acceso a la versión gratuita de la API. Luego, decidieron preparar los datos eliminando cualquier tipo de material considerado innecesario como los emoticones, caracteres especiales y espacios en blanco. Luego, procedieron a crear un modelo de análisis de sentimiento utilizando una red neuronal de siete capas. Entre estas siete capas se incluye una para la ingesta de datos, otra de

vectorización y reducción de dimensionalidad, y la última capa de salida con una sola neurona indicando la clasificación positiva o negativa. Finalmente, se demostró que el modelo construido alcanza un 94% de precisión y, por lo tanto, se puede utilizar el modelo para la predicción de sentimientos y emociones al analizar los *tweets* de usuarios.

En resumen, se ha observado cómo distintos estudios han empleado técnicas similares de pre-procesamiento de información relacionadas a un conjunto de datos extraídos a partir de la API de Twitter. Los autores han ejecutado tareas de limpieza y filtrado de texto en la que se eliminan instancias duplicadas, nombres de usuarios y enlaces a sitios web externos. También incluyen tareas de *Stemming* y eliminación de artículos que no contribuyen a la polaridad del texto. No obstante, los modelos de clasificación no abarcan los excepciones o contextualizaciones enfocadas a la temática analizada. Es decir, no son capaces de identificar elementos clave del contexto de los videojuegos y por lo tanto podrían generar una clasificación errónea al momento de analizar.

Este trabajo propone un modelo de recolección de datos extraídos desde la plataforma Twitter. Utiliza como referencia las técnicas y tareas de pre-procesamiento utilizadas por los autores mencionados en este apartado para la evaluación de cada *tweet.* Procesa la información mediante un clasificador de sentimiento prediseñado de nombre VADER (Valence Aware Dictionary and Sentiment Reasoner) y otro modelo elaborado manualmente. VADER fue introducido en 2014 como un analizador de sentimiento de texto que combina «un análisis cualitativo y validación empírica utilizando calificadores humanos y la sabiduría de la multitud» (Calderón, 2017). De otra forma, utiliza un diccionario de palabras para calcular la intensidad y polaridad del texto. En cuanto al segundo modelo, se propone un algoritmo de clasificación supervisada SVM entrenada bajo un conjunto de reglas que le permiten contextualizar la clasificación del texto. Finalmente, se analizan los sentimientos expresados en cada *tweet* para determinar si la comunidad de *League of Legends* extiende su toxicidad al ámbito social.

3. Objetivos concretos y metodología de trabajo

El conocimiento previo de la comunidad de *League of Legends* y la red social Twitter se utilizan como base para el análisis del sentimiento cuyo objetivo general se presenta en la Sección 3.1. Los objetivos de este estudio se detallan en la Sección 3.2 y metodología se presenta en la Sección 3.3.

3.1. Objetivo general

Como se ha mencionado en el apartado anterior, el objetivo del presente trabajo consiste en elaborar un modelo de recolección y limpieza de datos utilizando técnicas de pre-procesamiento y normalización de los datos con el fin de elaborar un modelo de *Machine Learning* capaz de analizar y clasificar el sentimiento de tweets generados por individuos de la comunidad *gamer* y dirigidos a *League of Legends.*

Para llevar a cabo el modelo y ampliar el conocimiento sobre las técnicas de pre-procesamiento, limpieza y clasificación adquirido en investigaciones previas, se parte desde la adquisición de datos mediante el uso de la API de Twitter, descrita en la Sección 3.3 metodología. Del mismo modo, se hará uso de técnicas como: eliminación de características de *tweets* y *tweets* duplicados para entrenar y validar el modelo de *Machine Learning*.

3.2. Objetivos específicos

Para lograr el objetivo propuesto se plantean los siguientes objetivos específicos correspondientes a cada etapa del proyecto:

- Analizar la problemática del comportamiento tóxico en videojuegos y Twitter.
- Explorar y analizar los aportes académicos sobre el preprocesamiento de datos y análisis del sentimiento hasta la fecha.
- Diseñar un modelo de base de datos estandarizado que permita el almacenamiento de metadatos de *tweets.*
- Extraer y almacenar los metadatos de *tweets* de usuarios que mencionan a *League of Legends* en sus publicaciones por un periodo de 30 días.
- Explorar y analizar el conjunto de datos obtenido, mostrando de manera explicativa, clara y visual las características del conjunto de datos.
- Identificar y aplicar técnicas de preprocesamiento de datos sobre las instancias del conjunto de datos.

- Analizar el sentimiento de los datos obtenidos utilizando técnicas de aprendizaje supervisado.
- Diseñar y evaluar un modelo de *Machine Learning* capaz de identificar el sentimiento de *tweets* en el contexto de *League of Legends.*

3.3. Metodología del trabajo

En el siguiente apartado se describen las herramientas, tecnologías a utilizar, y se incluyen las ventajas y desventajas de cada tecnología. Posteriormente, se muestra una metodología para la recolección, almacenamiento y análisis de los datos que serán tratados en los siguientes apartados. Finalmente, se presenta una tabla con las distintas tareas a desarrollarse para la ejecución satisfactoria de este trabajo.

3.3.1 Herramientas y Tecnologías

A continuación, se describen las tecnologías y herramientas que serán utilizadas durante el desarrollo del trabajo partiendo desde el lenguaje de programación utilizado, la base de datos escogida para el tipo de datos a almacenar y la herramienta que permite la obtención de datos. Adicionalmente se describe el proceso de captura y análisis junto a las tareas que se deben cumplir en cada paso.

Lenguaje de Programación

Tomando en cuenta que en este proyecto se realiza un análisis exploratorio de los datos es necesario comprender y conocer cuáles son las herramientas disponibles. Existen cientos de lenguajes de programación, algunos más adecuados para el cumplimiento de esta tarea. La Figura 3, muestra una gráfica de los lenguajes más populares en el ámbito de la ciencia de datos y se observa cómo Python, R y Java se encuentran liderando esta lista con cifras semejantes para los años 2018 y 2019. En este análisis se omite el lenguaje de SQL ya que este es un lenguaje para consultas de base de datos.

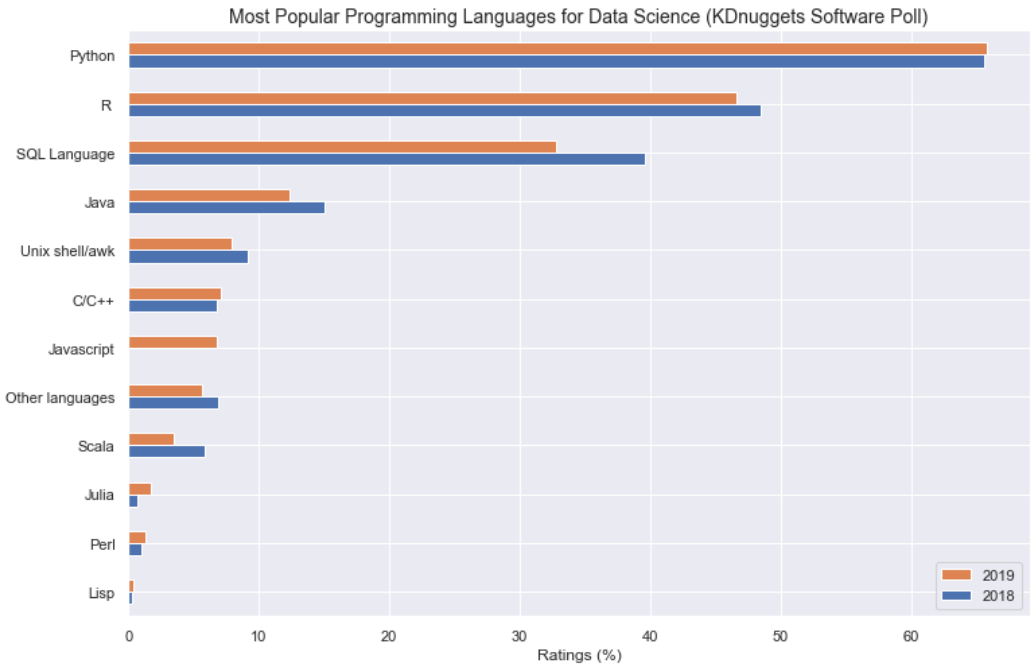

Figura 3: Lenguajes de programación más utilizados para Ciencia de Datos.

Fuente: Canales, Javier (2022)

En la Tabla 1 se presenta una descripción de los tres lenguajes más utilizados, sus ventajas y desventajas, con el fin de posteriormente analizar y escoger el lenguaje de programación que mejor se ajusta a la elaboración de este trabajo.

Tabla 1: Descripción de los lenguajes de programación.

Lenguaje	Descripción	Ventajas	Desventajas
Python	Python es un lenguaje de programación de propósito general con amplias aplicaciones en la industria. Desde las ciencias de datos, hasta el desarrollo web e inclusive desarrollo de video juegos.	• Herramienta de uso libre. • Sintaxis simple y similar al inglés. • Amplia gama de librerías y módulos. • Soporte de la comunidad.	• La tipificación dinámica puede resultar complicada para nuevos desarrolladores. • Lento en cuanto a tiempo de ejecución del código fuente.
R	R es un lenguaje y un entorno para la computación estadística y los gráficos.	• Es una herramienta de uso libre. • Enfocado fuertemente para el análisis estadístico y visualización de datos. • Soporte de la comunidad.	• Es lento • Almacena los objetos directamente en memoria. Provocando que no sea una herramienta viable para altos volúmenes de datos. • No es fácil de aprender ya que su sintaxis puede resultar confusa.
Java	Java es un lenguaje de propósito general, estrictamente tipado y orientado a objetos. A pesar de ser preferido para el desarrollo de sitios web o el desarrollo de aplicaciones, en los últimos años ha ganado su puesto dentro de la comunidad de la ciencia de datos.	• Mucha atención a la seguridad. • Multiplataforma. • Significativamente más rápido al compararlo con otros lenguajes debido que es un lenguaje compilado. Es decir, se conoce el tipo de las variables previo a la ejecución del programa.	• Difícil de comprender ya que puede volverse extenso al utilizar sintaxis de código compleja. • Alto uso de memoria. • Dado que debe ser interpretado por la máquina virtual de Java en tiempo de ejecución, siempre existirá una caída en rendimiento. • Requiere ser re-compilado y re-ejecutado cada vez.

Fuentes: Vitaly Naumenko (2022); Javier Canales (2022); Albert Christopher (2022).

Al observar Java, se puede decir que es un lenguaje escalable y con bases fuertes para el desarrollo de aplicaciones. Si bien ha ganado bastante popularidad en los últimos años para el ámbito de la ciencia de datos, este lenguaje carece de las cualidades y herramientas

necesarias para un análisis exploratorio de datos, sobre todo porque se debe compilar y ejecutar el script para observar resultados, por lo tanto, se descarta Java como lenguaje de programación para el desarrollo de este trabajo. Por otro lado, Python y R, sí cuentan con las bases necesarias para cumplir este objetivo. Como se pudo observar en la Figura 3, ambos lenguajes son altamente usados por los científicos de datos por sus capacidades para el análisis estadístico. Para determinar el lenguaje que mejor encaja para este trabajo se evalúan 4 áreas clave: Recolección, Exploración, Modelación y Visualización de datos. En la Figura 4, se presenta una gráfica comparativa de las puntuaciones asignadas para cada lenguaje, seguido de una descripción de las características que se consideraron para la puntuación.

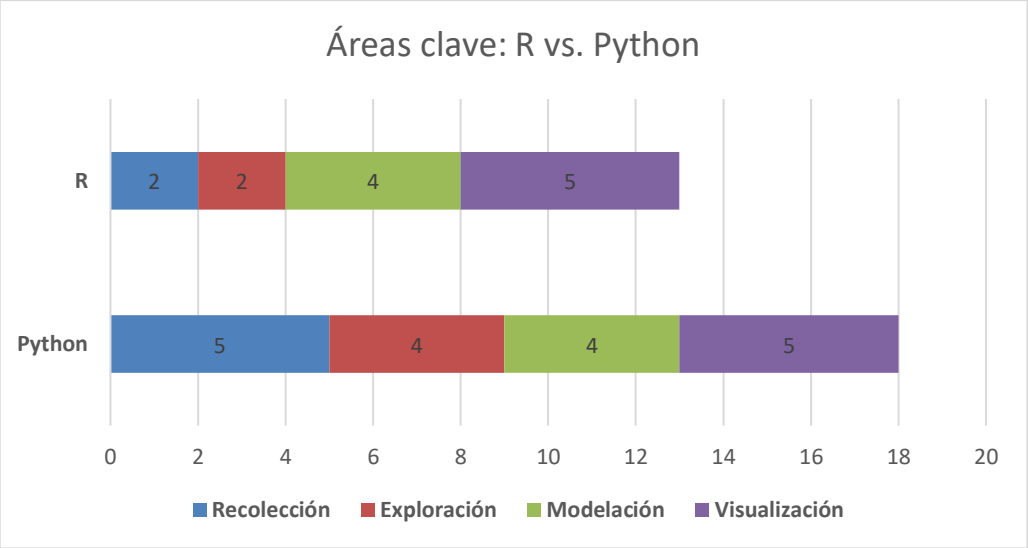

Figura 4: Evaluación de áreas clave para: R y Python.

Fuente: Elaboración propia.

1. **Recolección:** Python soporta todo tipo de formatos diferentes desde CSV hasta JSON e incluso datos obtenidos desde la web gracias a su librería de peticiones que facilita la creación de conjuntos de datos. Por el contrario, R está diseñado para que los científicos de datos importen el conjunto de datos al entorno de desarrollo desde un archivo Excel. Los paquetes modernos de R se encuentran diseñados para el *Web scraping* básico (IBM Cloud Team, 2021).

2. **Exploración:** Python cuenta con la librería Pandas, una herramienta de análisis de datos en la que se puede filtrar, ordenar y visualizar grandes cantidades de datos en cuestión de segundos. Por otro lado, R se encuentra optimizado para el análisis estadístico en donde se pueden construir distribuciones de probabilidad y ejecutar pruebas de estadística diferencial (Christopher, 2021).

3. **Modelación:** Ambos lenguajes de programación tienen capacidades semejantes para el modelado de los datos. En Python existen las librerías como Numpy, para el análisis de modelos numéricos y SciPy, para la computación y cálculos científicos. Mientras que, en R, existen paquetes como Tidyverse, que tiene características similares a las anteriores (IBM Cloud Team, 2021).

4. **Visualización:** Python cuenta con la librería Matplotlib, la cual se puede utilizar para generar gráficos. Sin embargo, R cuenta con las funcionalidades incorporadas de generar visualizaciones fácilmente (IBM Cloud Team, 2021).

Como se puede observar en la Figura 4, R recibió una puntuación de 13 mientras que Python recibió 18 de los 20 puntos disponibles en la sumatoria de las 4 áreas de evaluación. Con esta premisa, se decidió utilizar Python como el lenguaje de programación para el desarrollo de este trabajo. Es importante recalcar que uno de los puntos fuertes de este lenguaje es su simplicidad de la sintaxis y fácil entendimiento al parecerse bastante al idioma inglés. Además, la librería Pandas permite el análisis y manipulación de grandes cantidades de datos en un corto periodo de tiempo. Por último, cuenta con una amplia gama de librerías que facilitan la recolección, exploración, análisis y visualización del conjunto de datos.

Base de Datos MongoDB

El almacenamiento es uno de los aspectos importantes para este trabajo y una base datos es una herramienta indispensable para alcanzar uno de los objetivos planteados. Sin una base de datos que almacene la información, el trabajo se enfrentaría a diversos problemas de análisis y tiempo ya que, con cada sesión de trabajo, se tendría que solicitar nuevamente la información y analizar desde cero. Oracle (2022) define a una base de datos como «una colección organizada de información estructurada, o datos, normalmente almacenada electrónicamente en un sistema informático» y pueden ser de varios tipos: Relacionales y NoSQL. A continuación, se discuten las principales diferencias entre las dos y se analiza la base de datos que mejor se ajusta a las necesidades del proyecto dentro de uno de los tipos de bases de datos.

Las bases de datos relacionales almacenan los datos en forma de tablas que se forman de filas y columnas. Cada columna de la tabla corresponde a una categoría de datos, por ejemplo, un nombre o una dirección, mientras que cada fila contiene un valor de datos para la columna correspondiente. Se dice que este tipo de bases de datos es relacional por que se componen de diversas tablas que se relacionan unas con otras. Por ejemplo, se podría tener una tabla con información de un cliente mientras que en otra un historial de compras (Loshin, 2022). Por

otro lado, las bases de datos NoSQL son todas aquellas que almacenan información en un formato distinto a las tablas relacionales (MongoDB, Inc., 2022).

Tabla 2: Principales Diferencias de Bases de datos Relacionales y no Relacionales

Características	SQL	NoSQL
Modelo de Almacenamiento	Tablas con filas y columnas fijas.	Documentos (JSON), Pares clave-valor, Grafos.
Historia	Creada en 1970 enfocado en reducir datos duplicados.	Creado en el 2000 enfocado en el escalamiento y permitiendo el cambio rápido de la aplicación impulsado por las prácticas ágiles y DevOps.
Ejemplos	Oracle, MySQL, Microsoft SQL Server, PostgreSQL.	MongoDB, Redis, Cassandra, Neo4j.
Esquemas	Rígido.	Flexible.
Escalamiento	Escalamiento vertical (Se aumenta recursos a un servidor).	Escalamiento Horizontal (Se agrega nuevos servidores).
Relaciones	Requerido.	No es requerido.
Asignación de datos a Objetos	Requiere de un ORM.	No requiere de un ORM.

Fuente: MongoDB (2022)

Partiendo por las bases de datos relacionales, estas almacenan los datos en forma de filas y columnas de forma rígida y difícilmente modificable. Es decir, tienen un esquema estandarizado por cada relación creada. Al observar la Tabla 2, se puede determinar que la arquitectura que tienen las bases de datos relacionales no son las adecuadas. En particular, se necesita de la flexibilidad que ofrece una base de datos no relacional al momento de definir el modelo de almacenamiento. Otro motivo para escoger una base de datos NoSQL se fundamenta en el hecho de que, para el análisis de este trabajo, se almacenan metadatos de *tweets* en donde no existe una relación de una instancia con otra. Finalmente, una base de datos no relacional permite la consulta y manipulación de los datos directamente desde el lenguaje de programación sin la necesidad de utilizar un *Object Relational Mapper* (ORM).

En la Tabla 2 se pudieron observar ejemplos de bases de datos NoSQL. A continuación, se ostenta una tabla con dos ejemplos de bases de datos no relacionales en la que se evidencia las principales diferencias y posteriormente se escoge una de ellas.

Tabla 3: MongoDB vs. Cassandra

Características	Cassandra	MongoDB
Desarrollado	Java.	C++.
Licencia	Libre.	Libre.
Arquitectura	*Peer-to-Peer.*	*Master-slave.*
Teorema CAP	Alta Disponibilidad y Tolerancia a Particiones.	Consistencia y Tolerancia a Particiones.
Formato	Tabular o Columnar.	Documentos JSON (BSON).
Modelo	Filas y Columnas.	Orientado a objetos.
Lenguaje de Consulta	Cassandra Query Language (CQL).	MongoDB Query Language (MQL).
Replicación	Sí.	Sí.
Agregación	No.	Sí.
Transacciones	Sí.	No.
Velocidad de lectura/escritura	• Escritura rápida. • Lectura lenta.	• Escritura lenta. • Lecturas rápidas.
Lenguajes soportados	C++, C#, Java, Python, NodeJS, Ruby, Go, Scala.	C/C++, C#, Java, Python, NodeJS, Ruby, Go, Scala, Matlab, R.

Fuente: Devashree Madhugiri (2022)

A partir de la Tabla 3 se puede observar similitudes entre las bases de datos. Por ejemplo, ambas son bases de datos NoSQL que funcionan bajo la licencia de uso libre. Sin embargo, para determinar cuál de estas resulta adecuada para este trabajo, se enfoca en las diferencias clave de sus características. Primero, en MongoDB se utiliza JSON o BSON, un modelo expresivo y flexible de almacenamiento de datos que permite adaptarse a los posibles cambios en los datos. Por otro lado, Cassandra recurre al estilo tradicional de almacenar los datos en forma de tabla, utilizando filas y restringiendo el tipo de dato por cada columna. Segundo, MongoDB tiene una velocidad de escritura lenta, esto se debe a su arquitectura *master-slave* en la cual un nodo del sistema se encarga de replicar la información a los nodos correspondientes. Por otro lado, Cassandra tiene una arquitectura *Peer-to-Peer* en la que todos los nodos se encuentran a cargo de la distribución de los datos. Gracias a su arquitectura, MongoDB logra una velocidad de lectura rápida que permite realizar consultas frecuentemente y obtener una respuesta de forma inmediata.

La velocidad y arquitectura de cada base de datos se encuentra estrechamente relacionada con el teorema CAP. Este indica que un sistema distribuido puede entregar dos de las tres características que se describen a continuación:

- **Consistencia:** Se basa en replicar los datos escritos a un nodo, hacia los demás. De esta forma, todos los usuarios conectados observarán la misma información al mismo tiempo (IBM cloud Education, 2019).
- **Disponibilidad:** Todas las peticiones de datos deben dar una respuesta. Esto se debe cumplir incluso ante la caída de un nodo (IBM cloud Education, 2019).
- **Tolerancia a las particiones:** Quiere decir que el sistema distribuido debe permanecer en operación a pesar de que se produzca una falla de comunicación entre los nodos del sistema (IBM cloud Education, 2019).

La Figura 5 muestra las características que ofrece MongoDB son consistencia y tolerancia a las particiones, mientras que Casandra brinda disponibilidad y tolerancia a las particiones. Para el caso en estudio, no se requiere que los datos se encuentren disponibles todo el tiempo, por lo que el componente que ofrece Cassandra no es el indicado. Sin embargo, la consistencia de la información es de importancia para un análisis correcto del texto extraído, motivo por el cual MongoDB es la base de datos escogida.

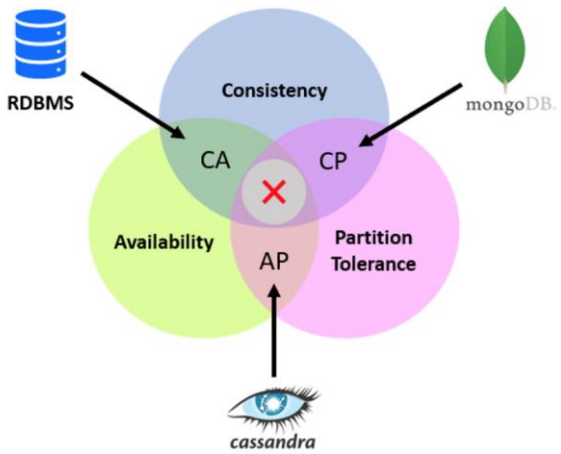

Figura 5: Teorema CAP.

Fuente: Devashree Madhugiri (2022)

Herramientas

Una de las herramientas clave para el desarrollo de este proyecto es el consumo de la interfaz de programación de aplicaciones o API por sus siglas en inglés. Según RedHat (2022), «Las API permiten que su producto o servicio se comunique con otros productos y servicios sin tener que saber cómo están implementados». Es decir, permite realizar consultas de datos a un servicio externo, el cual proporciona una respuesta en forma de JSON. Para el desarrollo de este trabajo, se consume la interfaz de Twitter a través de una librería de Python

denominada *tweepy*. El Twitter API es de uso libre y su documentación se encuentra en el siguiente enlace: Twitter API[2]. De la misma manera, se utiliza la librería de *pymongo*, este facilita la conexión y ejecución de consultas a la base de datos desde el entorno de desarrollo. También, se utiliza el entorno de desarrollo integrado Visual Studio Code o IDE por sus siglas en inglés. Este se caracteriza por tener instrumentos integrados que ayudan al desarrollo y a la depuración de problemas en el código. La Figura 6 muestra de forma general el entorno de trabajo de Visual Studio Code. En la región de la izquierda, se encuentra el explorador de archivos y componentes adicionales como el control de historia, búsqueda y configuración del entorno. En la región derecha, se ubica el archivo seleccionado para editar. Por último, la Figura 7 muestra la herramienta de MongoDB Compass, un gestor de base de datos para MongoDB que permitirá realizar consultas y observar la respuesta en una interfaz gráfica.

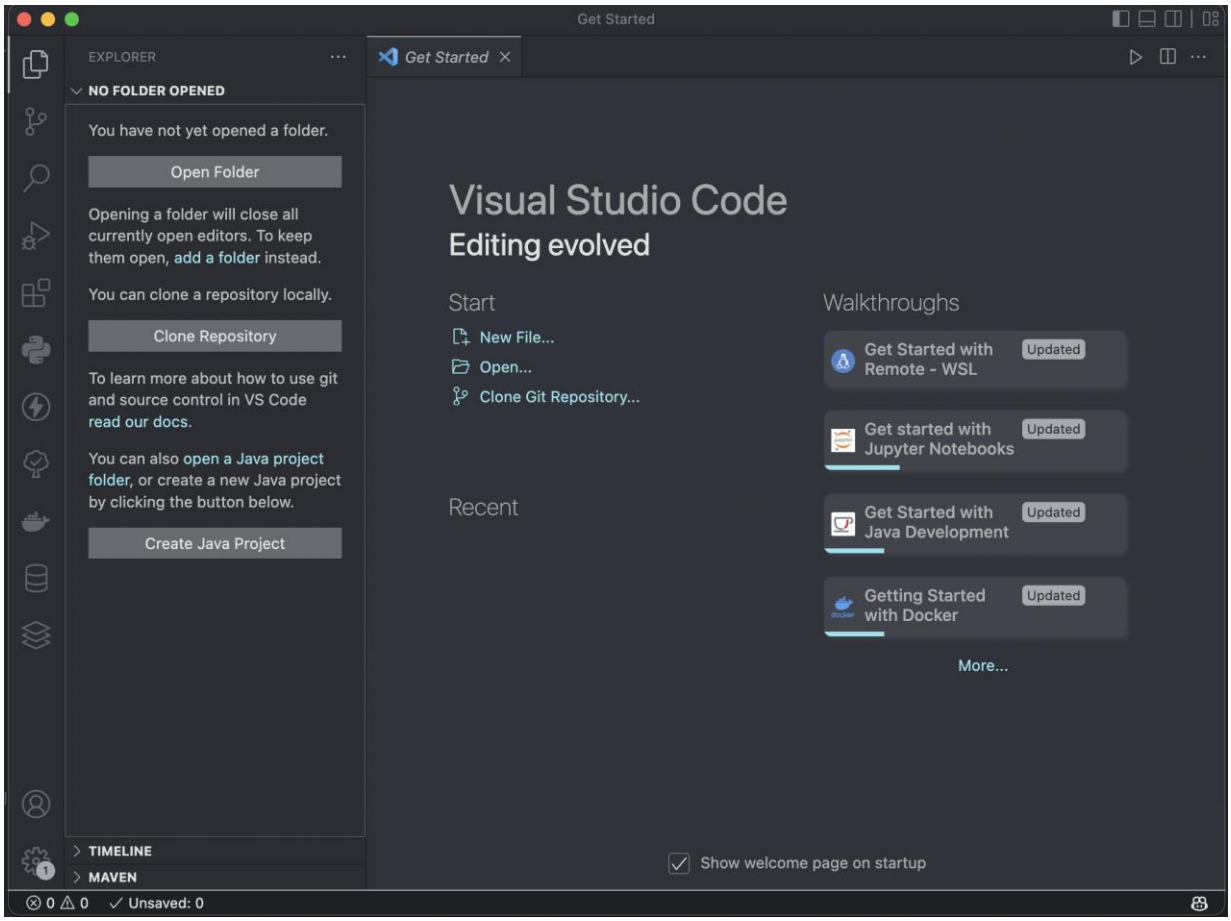

Figura 6: Entorno de desarrollo integrado.

Fuente: Elaboración propia (Captura de pantalla)

[2] Twitter API: https://developer.twitter.com/en/docs/twitter-api

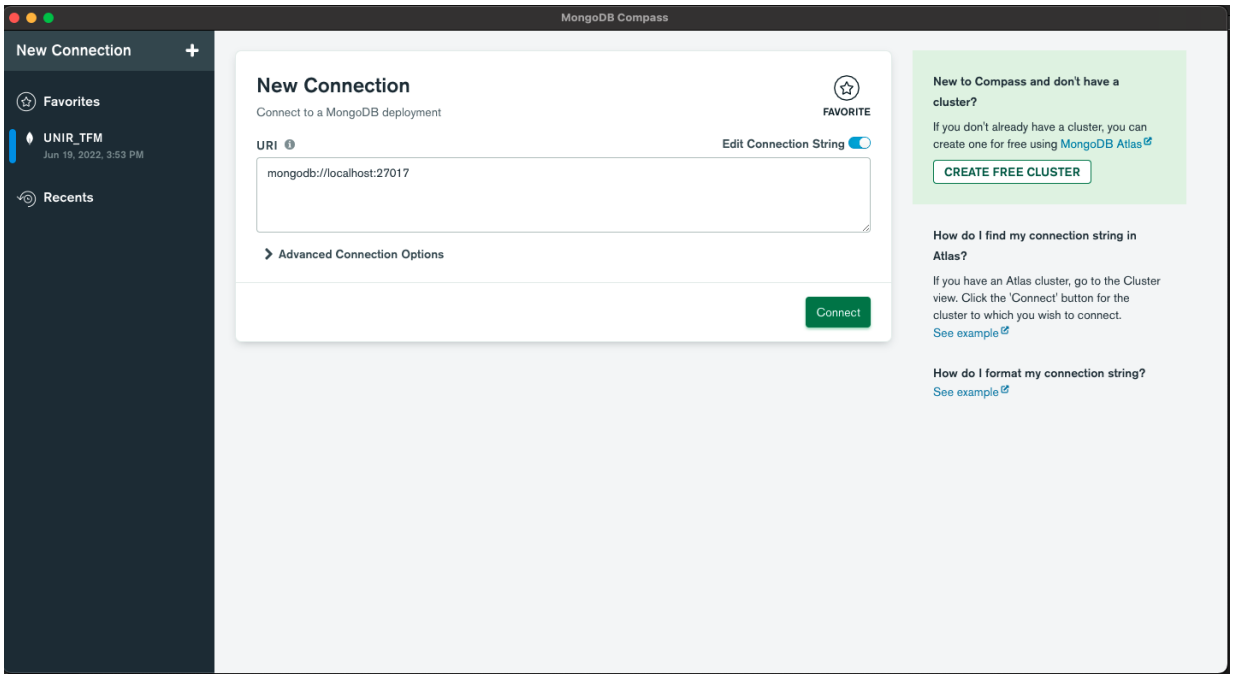

Figura 7: MongoDB Compass.

Fuente: Elaboración propia (Captura de pantalla)

3.3.2 Metodología

Este apartado describe el proceso utilizado para la recolección de datos a partir del API de Twitter. Para lograr dicho análisis, Amir Sinaeepourfard (2016) propone el modelo: *The comprehensive scenario agnostic data lifecycle model* o COSA-DLC por sus siglas (p.102). Este considera todas las fases del ciclo de vida del dato y se divide en tres bloques principales (Figura 8). El bloque *Data Acquisition* se encarga de recolectar los datos para el sistema y las envía hacia el bloque *Data Preservation* o hacia *Data Processing*. El bloque de procesamiento tiene la tarea de realizar todos los cálculos relacionados a la extracción del conocimiento utilizando las respectivas técnicas de análisis de datos. Finalmente, el bloque de almacenamiento guarda la información y la conserva para la futura presentación o tratamiento posterior.

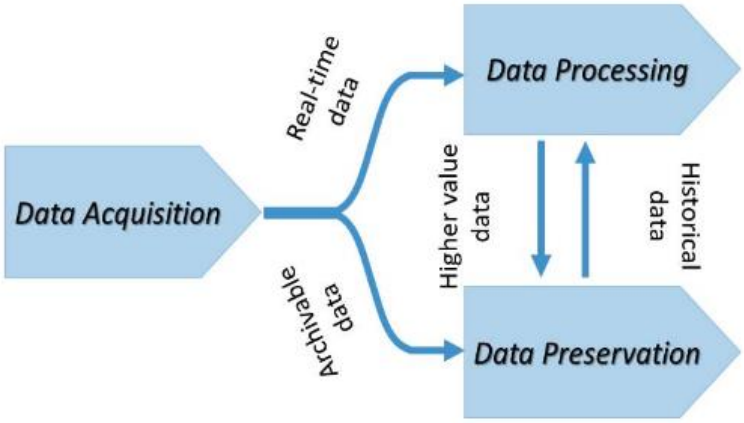

Figura 8: Metodología COSA-DLC.

Fuente: Sinaeepourfard (2016)

De la metodología planteada, la Tabla 4 presenta las tareas a ejecutarse y, después, una descripción de cara una:

Tabla 4: Tareas a realizar.

Bloque	Tareas	Requisitos
Adquisición	**Tarea 1.-** Crear cuenta de desarrollador.	Correo electrónico.
	Tarea 2.- Crear proyecto y obtener API *keys* y *token*.	Cuenta de desarrollador de Twitter.
	Tarea 3.- Diseñar el modelo y la estructura de los datos a capturar.	*Tweet* de ejemplo
	Tarea 4.- Captura de datos mediante la API de Twitter.	API *keys* y *token*
Almacenamiento	**Tarea 5. –** Creación de base de datos en MongoDB.	Cuenta de MongoDB.
	Tarea 6.- Almacenamiento de la información según el modelo diseñado en la Tarea 3.	Diseño del modelo de base de datos.
Procesamiento	**Tarea 7.-** Limpieza y preprocesamiento de *tweets*.	Conjuntos de datos obtenido en Tarea 3 y almacenados en Tarea 6.
	Tarea 8.- Análisis del sentimiento sobre conjunto de datos limpio.	Conjuntos de datos limpio.

Fuente: Elaboración propia

Adquisición de datos

En esta etapa se realiza la captura y diseño estructural del modelo de datos para su manipulación. Adicionalmente, se presenta un ejemplo de captura de *tweets*.

- **Tarea 1.- Crear cuenta de desarrollador**

  Previo a ejecutar esta tarea, es necesario contar con un correo electrónico ya que llegará un correo de verificación al crear la cuenta. Para registrarse y obtener una cuenta de desarrollador, basta con dirigirse al siguiente enlace: https://twitter.com/i/flow/signup. Se solicitará que se ingrese el nombre de la cuenta, ubicación, y detalles del caso de uso. Después, se deberá leer y aceptar el acuerdo de desarrollador y finalmente, hacer clic en enviar. En pocos minutos un correo electrónico llegará a la cuenta enlazada con Twitter, solicitando que se verifique la cuenta de desarrollador. Una vez realizada la verificación, se tendrá acceso al portal de desarrollador de Twitter. La Figura 9 muestra una captura de pantalla del inicio del proceso de registro para una cuenta de desarrollador en el portal de Twitter.

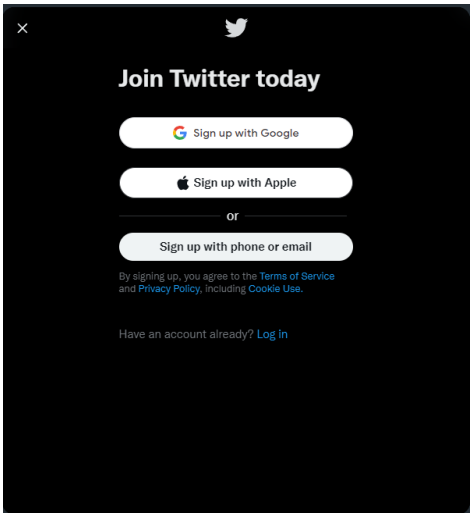

Figura 9: Registro de cuenta Twitter. (Captura de Pantalla).

Fuente: https://twitter.com/i/flow/signup

- **Tarea 2.- Crear proyecto y obtener las API *keys* y *token*.**

  En este paso, se procede a crear un proyecto en el portal de desarrollador de Twitter. Al hacer clic en 'Crear App' se revelará una pantalla similar a la Figura 10 en donde se debe especificar el nombre de la App. Después, se presentarán las siguientes credenciales (Figura 11): *API Key*, *API Key Secret*, *Bearer Token*. Adicionalmente, se requiere de un *Access Token* y *Access Token Secret*, que pueden ser obtenidos fácilmente desde los ajustes de la App recientemente creada. Todas las credenciales deben ser guardadas ya

que serán utilizadas en tareas posteriores con el fin de autenticación por cada petición al API de Twitter.

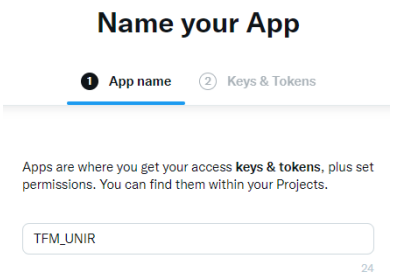

Figura 10: Creación de App (Captura de Pantalla).

Fuente: Twitter Developer Portal

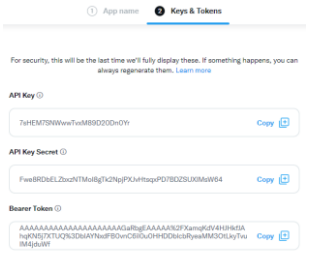

Figura 11: Keys & Tokens (Captura de pantalla).

Fuente: Twitter Developer Portal

- **Tarea 3.- Diseñar el modelo y la estructura de los de los datos a capturar.**

El API de Twitter tiene mucha flexibilidad en cuanto los parámetros que se pueden recuperar desde las respuestas a sus peticiones. Sin embargo, es importante plantear un modelo estándar de esta información. Por defecto, cada petición de un tweet retornará el *id,* un identificador único del *tweet y text*, refiriéndose al texto que el autor escribió. Para este análisis, se considera necesario recuperar atributos adicionales como, por ejemplo, el `author_id`, `created_at` y `lang` indicando el autor del *tweet*, la fecha en la que fue creado y el lenguaje, respectivamente. Estos campos permitirán explorar las preguntas ¿Cuántos *tweets* son emitidos diariamente? ¿Cuáles son los usuarios que más interactúan? y ¿En qué lenguaje se generaron? El Listado 1, presenta un ejemplo del modelo de *tweet* a capturar y como se lo almacenaría en una base de datos.

Listado 1: Ejemplo respuesta del API al solicitar un tweet.

```
{
    "created_at": "2022-05-18T21:34:00.000Z",
    "retweet_count": 2,
    "reply_count": 0,
    "like_count": 0,
    "quote_count": 0
    "author_id": "1261133296903151616",
    "lang": "en",
    "text": "RT @yuuneelli: @LeagueOfLegends happy birthday best girl 😢💔 so glad you
are back to being super popular and also very loved by the communit…",
    "tweet_id": "1527039828054917120",
    "date_collected": "2022-05-18T00:00:00.000Z"
}
```

Fuente: Elaboración Propia

- **Tarea 4.- Captura de datos mediante la API de Twitter.**

Con los *keys* y *tokens* obtenidos en los pasos anteriores, es posible realizar peticiones al API. Mediante la ayuda de un *script* de Python, se realizan múltiples solicitudes de

información. En el Listado 2, se muestra un ejemplo de petición en la que se solicita los *tweets* más recientes y dirigidos a *League of Legends*. La variable `tweet_fields` contiene la lista de parámetros a recuperar de un *tweet y* la variable *`since_id`* permite especificar un *`tweet_id`* a partir del cual se iniciará la búsqueda. La función *`tweepy.Paginator()`* recibe varios argumentos: la ruta a la que se realizará la consulta, el filtro, el número de resultados por página y los parámetros ya mencionados. Como resultado se obtiene una lista de *tweets* listos para ser manipulados.

Listado 2: Ejemplo de petición al API.

```
1.  # atributos de los tweets
2.  tweet_fields = ['author_id','created_at','lang','public_metrics','text']
3.  # id del tweet
4.  since_tweet_id = 1525962355502358530
5.  # Consulta al API
6.  for response in tweepy.Paginator(client.search_recent_tweets, query =
    f'to:{username}',max_results = 100, tweet_fields = tweet_fields, since_id =
    since_tweet_id):
7.          print(f'Response {response.meta}')
```

Fuente: Elaboración Propia

Almacenamiento de datos

Una de las principales limitaciones del API de Twitter es la cuota mensual de peticiones que se puede realizar. Con el fin tener los datos disponibles para su manipulación se realizan las siguientes tareas.

- **Tarea 5. - Creación de base de datos en MongoDB**

  Una vez creada la cuenta de MongoDB, se procede a crear un *clúster* sobre que se trabajará y guardarán los datos. Una guía detallada del paso a paso para la creación de la base de datos se encuentra en el siguiente enlace: MongoDB[3]. El procedimiento utilizado para este estudio se lo detalla en el Capítulo 4 Desarrollo. Al final, se obtiene el *clúster* de la Figura 12 donde se observa la versión, la región y otros ajustes especificados durante el proceso de configuración.

[3] Crear base de datos de MongoDB: https://www.mongodb.com/basics/create-database

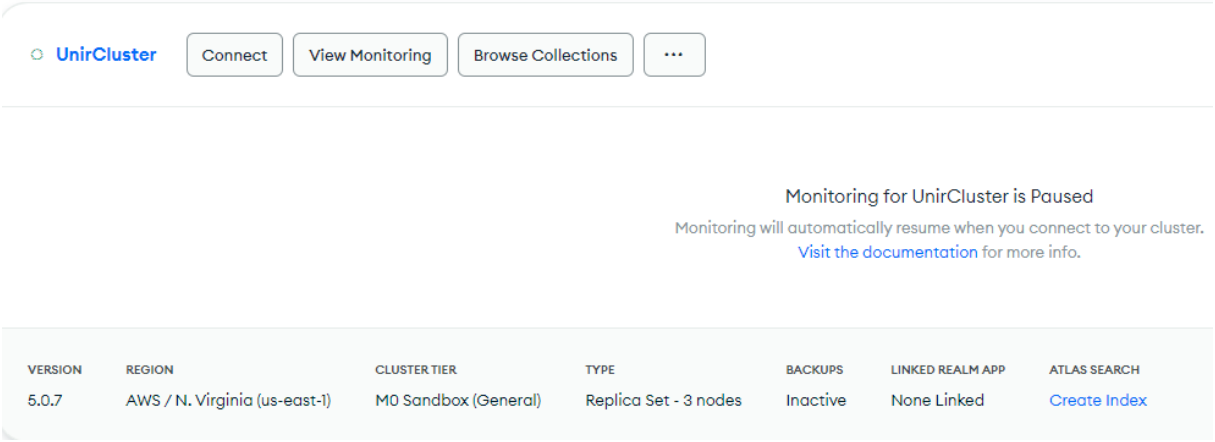

Figura 12: Clúster creado para la Base de datos.

Fuente: Elaboración Propia.

- **Tarea 6.- Almacenamiento de la información según el modelo diseñado en la Tarea 3**

Del resultado de la Tarea 5 se obtiene una base de datos lista para ser utilizada. Al ser una base de datos NoSQL, se tiene la flexibilidad de estructurar los datos de acuerdo con la estructura planteada en la Tarea 3. La Figura 13 evidencia el flujo de los datos que se divide en tres componentes: un script de Python que actúa como controlador, el API y la base de datos. El script de Python se encarga de realizar las peticiones hacia el API y a la vez las recepta para luego enviarlos hacia la base de datos en MongoDB.

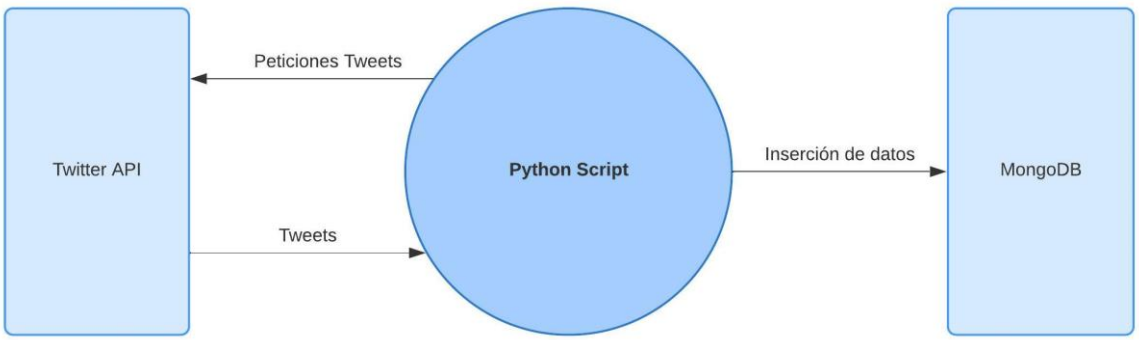

Figura 13: Flujo de solicitud de tweets y almacenamiento.

Fuente: Elaboración propia.

Procesamiento de datos

Para este punto, la recolección y almacenamiento de datos ha finalizado y se procede a realizar un análisis exploratorio de los datos ejecutando las siguientes tareas:

- **Tarea 7.-** Limpieza y preprocesamiento de *tweets*

Después de la recolección de un alto volumen de datos es común encontrar datos duplicados, incompletos y/o con errores de formato. Con la ayuda de Python y la librería Pandas se procede a corregir los datos de forma que todas las instancias sean consistentes y puedan aportar para la resolución del estudio. Las subtareas realizadas con Pandas son las siguientes:

1. Eliminación de URL y nombres de usuario. Son pedazos de texto considerados como ruidosos ya que no aportan a la polaridad del texto.

2. Eliminación de caracteres especiales, símbolos o números.

3. Eliminación de *stopwords.* Es decir, eliminar palabras que no contribuyen a la polaridad del texto y pueden generar conflictos con el modelo de clasificación.

4. Normalización del texto. Se refiere a transformar el texto para que se encuentre solamente minúsculas ya que “HOLA MUNDO” y “hola mundo” no modifica el sentido original del texto. Mediante esta transformación, se evita que exista una diferencia entre mayúsculas y minúsculas.

5. Tokenización. Se separa las palabras mediante espacios en el texto con el fin de que el modelo de clasificación pueda analizarlas de forma independiente.

Al efectuar todas estas subtareas, se espera que el modelo de clasificación otorgue una polaridad a cada *tweet* de forma acertada permitiendo así la ejecución de la siguiente tarea.

- **Tarea 8.-** Análisis del sentimiento sobre conjunto de datos limpio

Finalmente, con los resultados de la Tarea 7 se enviarán los *tweets* a un modelo de clasificación prediseñado, VADER. El cual devolverá como resultado una de las tres categorías: positivo, negativo o neutral. Posteriormente, se crea un nuevo clasificador entrenado bajo los parámetros del contexto de *League of Legends* y se comparan los resultados de ambos clasificadores mediante gráficas de tarta y una nube de palabras en la que se visualiza la frecuencia de las palabras más utilizadas por la comunidad de *League of Legends.* Por último, se recoge la información presentada por los clasificadores y se comenta sobre ellos.

4. Desarrollo específico de la contribución

En el siguiente apartado se procede a describir los resultados obtenidos a partir de las tareas en la Tabla 4 y se realiza un análisis utilizando el código desarrollado y visualizaciones sobre los datos. Las siguientes secciones se encuentran divididas de acuerdo con las fases de la metodología descrita anteriormente y presentada en la Figura 5. La Sección 4.1 presenta la adquisición de datos, la Sección 4.2 describe el almacenamiento y la Sección 4.3 muestra el análisis de los datos.

4.1 Adquisición de datos

Para llevar a cabo esta fase, se realizan peticiones de *tweets* a un punto de enlace (*endpoint*) especifico. La documentación de Twitter la denomina *Tweet Lookup*, y explica que esta realiza una búsqueda según los parámetros establecidos en el campo *query* y retorna una lista de *tweets* que cumplen con las condiciones especificadas en dicho parámetro. Sin embargo, dado que se utiliza la versión libre del API, los resultados obtenidos corresponden a los *tweets* realizados por usuarios en los últimos siete días. Por lo tanto, se planteó realizar una recolección de datos durante treinta días comenzando el 29 de abril 2022, en los cuales el código se ejecutaría diariamente, modificando el campo `since_id`. Para encontrar el campo, se filtró manualmente la base de datos por orden de fecha de creación de forma descendente y se extrajo el `tweet_id` del primer resultado. Este campo luego fue incluido en el código del Listado 2.

El primer paso para realizar peticiones a la librería `tweepy` es la configuración del cliente.

Listado 3: Configuración del cliente para consumo de Twitter API mediante tweepy.

```
1.  # Se configura la API de Twitter
2.  client = tweepy.Client(
3.          bearer_token=env_keys.BEARER_TOKEN,
4.          consumer_key=env_keys.CONSUMER_KEY,
5.          consumer_secret=env_keys.CONSUMER_SECRET,
6.          access_token=env_keys.ACCESS_TOKEN,
7.          access_token_secret=env_keys.ACCESS_TOKEN_SECRET,
8.          wait_on_rate_limit=True,
9.  )
```

Fuente: Elaboración Propia.

En el Listado 3 se evidencia la configuración del cliente que hará las peticiones hacia el Twitter API. La llamada al objecto *Client* en la línea 2 se encarga de establecer los permisos de autorización y de acceso. Por lo tanto, se le asigna a cada parámetro las credenciales obtenidas en la Tarea 2, descrita en el Capítulo 3. Al ejecutar la sentencia, se obtiene como

resultado el cliente configurado y listo para realizar consultas a un punto de enlace. El Listado 4 muestra un fragmento del código utilizado para la adquisición de los datos.

Listado 4: Fragmento de Código - Adquisición de datos.

```
1.  def get_recent_tweets(username):
2.      since_tweet_id = 1530277916575244288
3.      tweet_fields = ['author_id','created_at','lang','public_metrics','text']
4.      for response in tweepy.Paginator(client.search_recent_tweets, query =
    f'to:{username}', max_results = 100, tweet_fields = tweet_fields, since_id =
    since_tweet_id):
5.          print(f'Response {response}')
```

Fuente: Elaboración Propia

En general, el Listado 4 muestra una función de Python que recibe un nombre de usuario como parámetro y en la línea 4, se utiliza el cliente para realizar la consulta al punto de enlace. Ventajosamente, *tweepy* recibe un parámetro *since_id*, el cual permite especificar un *tweet* a partir del cual se realiza la búsqueda de *tweets.* Esto permite tener un control sobre los *tweets* solicitados y evita duplicidad. En la línea 2 se asigna un identificador de *tweet* a la variable *since_tweet_id,* la cual es utilizada más adelante en la línea 4. Es importante mencionar que en la primera ejecución del código este parámetro fue omitido ya que no se contaba con ningún *tweet* previo. Continuando, en la línea 3 se define una lista de los atributos que se desean obtener de cada *tweet*, siendo *text* el principal. En la línea 4 se ejecuta un bucle en el que se itera por cada página de resultados obtenidos al consumir el punto de enlace. En este se especifica que la condición de búsqueda sea todos aquellos *tweets* que se encuentren dirigidos a *LeagueOfLegends*, el nombre del usuario correspondiente al juego de interés. Adicionalmente, se define la cantidad máxima de resultados por página y los campos que se desean recuperar de cada *tweet*.

Listado 5: Ejemplo de Resultado al realizar una consulta.

```
1.  Response(data=[<Tweet id=1538623672960929792 text='@LeagueOfLegends project ✨✨✨'>,
    <Tweet id=1538609548461846530 text='@LeagueOfLegends Project by a long shot'>, <Tweet
    id=1538608660930301952 text='@LeagueOfLegends https://t.co/ROVEE3hIPF'>, <Tweet
    id=1538607253263208451 text='@LeagueOfLegends Project.'>, <Tweet id=1538605809264902144
    text="@LeagueOfLegends  @LoLUKN  \n\nI've been put dodge timer due to software issues on
    league is there anything we can do?\n\n#73853652">, <Tweet id=1538605478908944384
    text='@LeagueOfLegends definitely pool party'>, <Tweet id=1538603531921412099
    text='@LeagueOfLegends None because the gay chameleon isnt in it
    https://t.co/9Nd8qfQXlH'>, <Tweet id=1538582224856961024 text='@LeagueOfLegends
    Project'>, <Tweet id=1538581813861416961 text='@LeagueOfLegends pool party is so superior
    tbh https://t.co/5g1vmFAj8P'>, <Tweet id=1538578571245113351 text='@LeagueOfLegends for
    the love of god please bring back twisted tree line #LeagueOfLegends #twistedtreeline
    #lol #RiotGames @riotgames #League_of_Legends'>], includes={}, errors=[],
    meta={'newest_id': '1538623672960929792', 'oldest_id': '1538578571245113351',
    'result_count': 10, 'next_token': 'b26v89c19zqg8o3fpyznxtxg7t7fz2rsok2kbzhd5i4ql'})
```

Fuente: Elaboración Propia.

El Listado 5 muestra un ejemplo del resultado obtenido. Este es un objeto de tipo *Response* el cual tiene 4 atributos: *data, includes, errors, meta*. En *data* se encuentra una lista de los *tweets* en que se pueden visualizar los atributos de *id* y *text*. El atributo *includes* contiene también una lista de *tweets*, sin embargo, en este caso no recibió ningún resultado. La diferencia entre estos dos atributos se encuentra en que el parámetro *includes* es utilizado por el API cuando en la petición realizada por el cliente se especifica algún parámetro perteneciente a los atributos de expansión, que requieren de un mayor nivel de permisos y que se encuentran fuera de los parámetros de análisis de este trabajo. El atributo *error* retorna una lista de mensajes de error en caso de que hubiese. Por ejemplo, puede retornar mensajes relacionados a falta de permisos o cuando se intenta solicitar *tweets* que sobrepasan los siete días especificados por el punto de enlace.

4.2 Almacenamiento de datos

Para el desarrollo de esta fase la Tarea 5 debió ser completada, esta proporciona un enlace de conexión hacia la base de datos. De la misma forma que en la Sección 4.1, el primer paso es realizar la configuración del cliente de mongo utilizando la librería *pymongo*. El Listado 6 muestra la configuración del cliente de MongoDB a la cual se le pasa como parámetro el enlace de conexión. Posteriormente en la línea 4 se crea una base de datos de nombre *twitter_db* y en la línea 6 la colección *tweet_collection* que almacenará todos los *tweets*.

Listado 6: Configuración de cliente de MongoDB.

```
1.  # Se Configura MongoDB
2.  mongo_client =
    mongo.MongoClient(f"mongodb+srv://{env_keys.MONGO_USER}:{env_keys.MONGO_PASSWORD}@unirclu
    ster.92brj.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
3.  # Se llama a la db, si no existe se crea
4.  db = mongo_client['twitter_db']
5.  # Se llama a la colección, si no existe se crea
6.  tweet_collection = db['tweet_collection']
```

Fuente: Elaboración propia.

Una vez ejecutadas las sentencias del Listado 6, se tendría listo un objeto de Mongo para efectuar consultas e inserciones a la base de datos. Continuando con la función *get_recent_tweets* definida en el Listado 4 y ahora en el Listado 7, una vez obtenida una respuesta desde el API, se procede a verificar que existan *tweets* para ingresar en la base de datos mediante la validación de la línea 2 en la cual se accede al atributo *meta* y verifica que la cantidad de resultados sea distinta de 0. Dado que la información se encuentra en una lista, se define un nuevo bucle en el cual se itera por cada *tweet*. La línea 4 define un diccionario, una estructura de datos de Python, que utiliza el modelo definido en la Tarea 3 del Capítulo 3 y asigna en cada clave el valor correspondiente del *tweet* en cuestión. Posteriormente, en la

línea 16 se realiza una consulta a la base de datos para recuperar un *tweet* con el mismo código de identificación. Si esta consulta retorna un valor, quiere decir que el *tweet* que se encuentra en la iteración ya fue almacenado previamente, caso contrario se lo agrega a un *Dataframe* de Pandas definido en la línea 1. De este modo se tiene un conjunto de *tweets* únicos que son insertados en la base de datos al ejecutar la sentencia en la línea 19.

Listado 7: Inserción de Tweets a la base de datos.

```
1.  recent_tweets_df = pd.DataFrame()
2.      if response.meta['result_count'] != 0:
3.          for tweet in response.data:
4.              row = {
5.                  'tweet_id': tweet['id'],
6.                  'author_id': tweet['author_id'],
7.                  'text': tweet['text'],
8.                  'lang': tweet['lang'],
9.                  'created_at': tweet['created_at'],
10.                 'retweet_count': tweet['public_metrics']['retweet_count'],
11.                 'reply_count': tweet['public_metrics']['reply_count'],
12.                 'like_count': tweet['public_metrics']['like_count'],
13.                 'quote_count': tweet['public_metrics']['quote_count'],
14.                 'date_collected': datetime.now().strftime("%d/%m/%Y, %H:%M:%S")
15.             }
16.             tweet_in_coll = tweet_collection.find_one({'tweet_id': int(tweet['id'])})
17.             if tweet_in_coll is None:
18.                 recent_tweets_df = recent_tweets_df.append(row, ignore_index = True)
19.         tweet_collection.insert_many(recent_tweets_df.to_dict('records'))
```

Fuente: Elaboración propia.

En la Figura 14 se puede visualizar una captura de pantalla de MongoDB Compass. En este se pueden observar 2 ejemplos de *tweets* almacenados en la colección *tweet_collection*. Adicionalmente, se puede observar que hay un total de 19 697 instancias almacenadas al final del periodo de análisis.

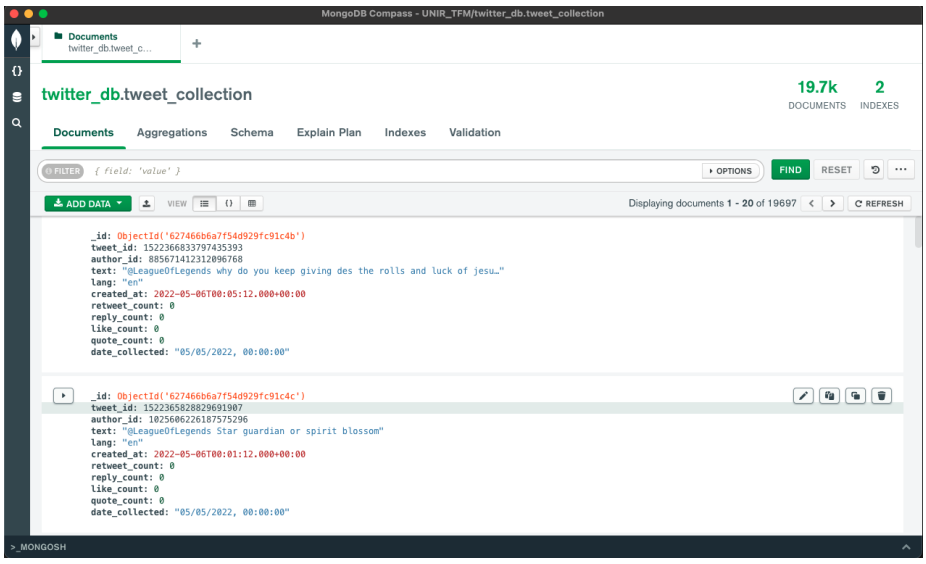

Figura 14: Tweets Almacenados en MongoDB.

Fuente: Elaboración propia

4.3 Análisis de datos

En esta sección se realiza un análisis sobre el conjunto de datos obtenido. Primero se exploran los datos en forma textual y luego de forma gráfica. Luego, se realiza un pre-procesamiento y limpieza del texto de cada *tweet*. Después, se analiza la polaridad que entrega un clasificador de sentimiento prediseñado y se evalúan los resultados. Finalmente, se implementa un modelo de clasificación definido bajo nuevas reglas que se ajustan al contexto de *League of Legends*.

4.3.1 Análisis exploratorio de los datos

Antes que todo, es importante mencionar que en este punto se ha concluido con dos de las fases del modelo COSA-DLC (Adquisición y Preservación) presentado en la Figura 8 y se inicia con el procesamiento y análisis de los datos. Como primer paso, se realiza la carga de la información al entorno de desarrollo con la ayuda de las librerías de *pymongo* y Pandas. El Listado 8 muestra en la línea 2 una consulta realizada a la colección definida en el Listado 6. En particular, se utiliza la condición de recuperar todos los *tweets* que se encuentren en el idioma inglés y ordenados de forma ascendente, es decir, el más antiguo se encuentra al principio de la lista de resultados. Se escoge el idioma inglés, ya que el clasificador prediseñado solo tiene soporte para este idioma. En la línea 3, se transforma la lista de resultados en un *Dataframe* de Pandas, esto permite que la manipulación, filtrado, análisis y entendimiento de los datos sea sencillo, como también habilita el análisis del conjunto de datos en forma de texto.

Listado 8: Carga de Datos.

```
1.  # Se carga los datos desde la DB filtrando por aquellos en inglés.
2.  raw_data = tweet_collection.find({'lang': 'en'}).sort("created_at", mongo.ASCENDING)
3.  df = pd.DataFrame(list(raw_data))
4.  df.head()
```

Fuente: Elaboración propia

En la Tabla 5 se puede observar el resultado de ejecutar la sentencia de la línea 4 en el Listado 8. La función *head()* permite observar las 5 primeras instancias y es posible determinar que el conjunto de datos cuenta con 11 columnas de distintos tipos. Puesto que este trabajo se enfoca en el análisis del sentimiento en el texto, se decidió conservar las columnas: *tweet_id, text* y *created_at* y eliminar el resto. Al hacerlo, se libera memoria del ordenador para aumentar la velocidad de procesamiento de los datos. La Tabla 6 muestra un ejemplo de cómo se encuentran los datos después de la eliminación de columnas:

Tabla 5: Ejemplo de Tweets recuperados desde la base de datos con su estructura original.

_id	tweet_id	author_id	text	lang	created_at	retweet_count	reply_count	like_count	quote_count	date_collected
6274698ea7f54d929fc939c6	1519837319964016640	1042309753551114240	@LeagueOfLegends Ugly and boring skins	en	2022-04-29 00:33:48	0	0	1	0	05/05/2022, 00:00:00
6274698ea7f54d929fc939c5	1519838071662993408	1429156502506545154	@LeagueOfLegends These are some amazing skins!...	en	2022-04-29 00:36:48	0	0	0	0	05/05/2022, 00:00:00
6274698ea7f54d929fc939c4	1519838585662459904	8925292260286504096	@LeagueOfLegends Literally no comment/announce...	en	2022-04-29 00:38:50	0	1	14	0	05/05/2022, 00:00:00
6274698ea7f54d929fc939c3	1519839873447960579	7566055775943065760	@LeagueOfLegends I go to sleep without my GPs...	en	2022-04-29 00:43:57	0	1	14	0	05/05/2022, 00:00:00
6274698ea7f54d929fc939c2	1519840363707523077	1207871786479689728	@LeagueOfLegends WHY DONT THEY STACK https://t...	en	2022-04-29 00:45:54	0	0	0	0	05/05/2022, 00:00:00

Fuente: Elaboración propia.

Tabla 6: Dataframe de tweets después de eliminar columnas.

tweet_id	author_id	text
1519837319964016640	1042309753551114240	@LeagueOfLegends Ugly and boring skins
1519838071662993408	1429156502506545154	@LeagueOfLegends These are some amazing skins!...
1519838585662459904	892529226028650496	@LeagueOfLegends Literally no comment/announce...
1519839873447960579	756605577594306560	@LeagueOfLegends I go to sleep without my GP s...
1519840363707523077	1207871786479689728	@LeagueOfLegends WHY DONT THEY STACK https://t...

Fuente: Elaboración propia

Una vez eliminadas las columnas, se utiliza `info()` una función que posibilita obtener más información acerca del tipo de cada columna. En detalle, esta función infiere que las columnas son de tres tipos distintos: `tweet_id` es de tipo `int64`, `text` es de tipo `Object` y `created_at` representa un `datetime`. Las cuales representan un número entero, una cadena de caracteres y una fecha respectivamente. El total de instancias en el conjunto de datos sin filtrar corresponden a 19 696. Sin embargo, como el conjunto de datos fue filtrado para dejar solo aquellos en el idioma inglés, la función indica que existen 15 385 instancias. Este número corresponde a la cantidad de *tweets* que será utilizados para el análisis en las siguientes fases.

Continuando con el análisis, y ahora de forma gráfica, en la Figura 15 se evidencia la distribución de *tweets* emitidos por usuarios hacia la cuenta de *League Of Legends.* En general, es evidente que los usuarios de Twitter interactúan con la cuenta de *League Of Legends* y existe un día en particular que sobresale del resto. Dado que la diferencia en la cantidad de *tweets* es bastante amplia, se considera importante realizar una búsqueda de los *tweets* emitidos por la cuenta en cuestión. Omitiendo el 3 de mayo, se observa que en promedio se escriben 408.62 *tweets* por día.

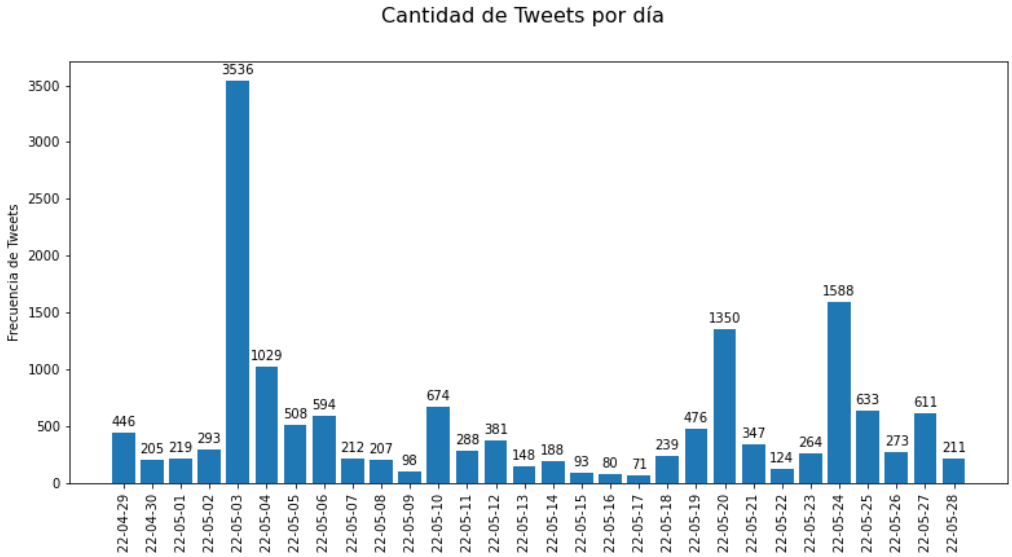

Figura 15: Tweets por día.

Fuente: Elaboración propia

Al hacer una búsqueda en la línea del tiempo por los *tweets* emitidos desde la cuenta *@LeagueOfLegends* se encontró con la Figura 16, un *tweet* que traducido desde el inglés se lee: «Bien, es hora de una pregunta importante. ¿Cuál es tú línea de *skins* favorita de todos los tiempos?» (*League Of Legends*, 2022). Este *tweet* incentiva a la comunidad a compartir sus gustos sobre alguna línea de *skins* dentro del juego y a participar en una conversación interactiva con el resto de la comunidad. Aquí, el termino *skin* se refiere a un artículo cosmético que se puede adquirir dentro del juego. De forma similar, la Figura 17 anuncia un nuevo personaje para formar parte del juego y la Figura 18 presenta una nueva línea de artículos cosméticos que podrán ser adquiridos por los jugadores. Se puede decir que los tres *tweets* tienen la intención de atraer a los jugadores e incentivarlos a interactuar. Ciertamente, estas interacciones abren la oportunidad de un análisis más provechoso en el que se puede analizar el texto insertado por cada usuario como se detalla en la siguiente sección.

Figura 16: Tweet de League of Legends - 03 de mayo 2022.

Fuente: League of Legends (2022)

Figura 17: Tweet de League of Legends - 20 de mayo 2022.

Fuente: League of Legends (2022)

Figura 18: Tweet de League of Legends - 24 de mayo 2022.

Fuente: League of Legends (2022)

4.3.2 Limpieza y Preprocesamiento de los datos

Existen aspectos dentro del texto que interfieren con la asignación de una categoría por parte del clasificador. Por lo tanto, es necesario realizar una limpieza del conjunto de datos. Primero, se utiliza la función *drop_duplicates()* de la librería Pandas sobre el conjunto de datos, este retorna un nuevo *DataFrame* sin filas duplicadas. Esto se lo realiza con el fin de evitar un sesgo en los datos debido a su existencia. Luego, se aplica un filtro para identificar *tweets* retransmitidos o *retweets*, esto se refiere que los usuarios reenviaron un mensaje publicado por otro, generando un nuevo *tweet_id,* no obstante, el contenido del *tweet* es el mismo y se lo considera como duplicado. El Listado 9 muestra un ejemplo de un mensaje reenviado y se lo identifica por los caracteres RT al principio. Se encontraron 2 234 instancias de mensajes reenviados que fueron eliminados del conjunto de datos cargado en la memoria, no de la base de datos.

Listado 9: Ejemplo de un mensaje reenviado.

```
1.  "text": "RT @riotgamesmusic: @LeagueOfLegends @KDA_MUSIC we couldn’t agree more with this
    reminder❤"
```

Fuente: Elaboración propia.

Continuando, se tomó la decisión de eliminar las referencias a cuentas de usuarios, *hashtags*, enlaces a direcciones externas y caracteres especiales. Todos estos elementos agregan ruido al mensaje de un usuario y no aportan con ningún valor al contenido del mensaje. Para lograr la limpieza del texto, en el Listado 10 se muestra la función *clean_tweet_text()* la cual recibe como argumentos el *Dataframe* de los *tweets,* un patrón por el cual se desea filtrar y un argumento opcional en el que se especifica el carácter de reemplazo. A partir de la línea 5, se realizan varios llamados a esta función con distintos patrones de expresiones regulares. Por último, en la línea 10 se transforma cada palabra a minúsculas. Esto se lo hace con la intención de que los modelos de clasificación no consideren dos palabras que evocan el mismo significado como distintas. Es decir, la palabra ‘Hola’ tiene el mismo significado que ‘hola’ o ‘HOLA’.

Listado 10: Función de limpieza de texto.

```
1.  def clean_tweet_text(tweets, reg_pattern, sub = ' '):
2.      tweets['text'] = tweets['text'].str.replace(reg_pattern, sub, flags=re.UNICODE)
3.      return tweets
4.  # clean tweets text
5.  df = clean_tweet_text(df, "@[\w]*")
6.  df = clean_tweet_text(df, "#([\w]+)")
7.  df = clean_tweet_text(df, "https?:\/\/[A-Za-z0-9.\/]*")
8.  df = clean_tweet_text(df, "[^A-Za-z]")
9.  # Transform the text case to lower case
10. df['text'] = [word.lower() for word in df['text']]
```

Fuente: Elaboración propia.

En la Tabla 7 se puede visualizar el efecto de aplicar una a una las expresiones regulares en un *tweet* de ejemplo:

Tabla 7: Patrones de expresiones regulares.

Expresión Regular	Texto	Eliminación de Ruido
"@[\w]*"	RT @riotgamesmusic: @LeagueOfLegends @KDA_MUSIC we couldn't agree more with this reminder Https://mylink.com #LOL ❤	RT we couldn't agree more with this reminder Https://mylink.com #LOL ❤
"#([\w]+)"	RT we couldn't agree more with this reminder Https://mylink.com #LOL ❤	RT we couldn't agree more with this reminder Https://mylink.com ❤
"https?:\/\/[A-Za-z0-9.\/]*"	RT we couldn't agree more with this reminder Https://mylink.com ❤	RT we couldn't agree more with this reminder ❤
"[^A-Za-z]"	RT we couldn't agree more with this reminder ❤	RT we couldnt agree more with this reminder

Fuente: Elaboración propia.

Adicionalmente, el Listado 11 muestra el proceso de eliminación de *stopwords* y de tokenización en la línea 1 y 2, respectivamente. Dado que las *stopwords* no aportan con significado o polaridad al texto, es posible eliminarlas para reducir la carga de procesamiento que se entrega a los modelos y acercarse más a una clasificación precisa. Por otro lado, analizar cada palabra individualmente ayuda a entender el contexto del conjunto de mensajes y a interpretar el significado analizando la secuencia de palabras.

Listado 11: Procesamiento del texto.

```
1.  df['text'] = df['text'].apply(lambda x: ' '.join([word for word in x.split() if word not
    in (stop_words)]))
2.  df['tokenized_text'] = [word_tokenize(word) for word in df['text']]
```

Fuente: Elaboración propia.

En la Tabla 8 se observa una comparativa de tres instancias de *tweets* en la que el texto ha sido limpiado, normalizado y tokenizado. Como se mencionó previamente, esto permite al clasificador enfocarse en las palabras clave del mensaje y no en ornamentos que no contribuyen con algo de valor al texto. En todos los casos se elimina la mención de la cuenta *@LeagueOfLegends* y a otros usuarios, como también enlaces externos y caracteres especiales.

Tabla 8: Comparación de tweets original y después del proceso de limpieza.

Texto Original	Texto Limpio	Texto Tokenizado
@LeagueOfLegends Ugly and boring skins	ugly and boring skins	[‘ugly’, ‘and’, ‘boring’, ‘skins’]
@LeagueOfLegends @KDA_MUSIC @riotgamesmusic Stan Kda stream more https://t.co/1M8IiHjYj7	Stan kda stream more	[‘Stan’, ‘kda’, ‘stream’, ‘more’]
@LeagueOfLegends @KDA_MUSIC @riotgamesmusic need comeback so bad :<<	Need comeback so bad	[‘need’, ‘comeback’, ‘so’, ‘bad’]

Fuente: Elaboración propia.

Ahora, el conjunto de datos se encuentra preparado para ser analizado en cuanto a su contenido. Por lo tanto, se procedió a realizar una nube de palabras (Figura 19) en la que se pueden observar cuáles son las palabras que se repiten con más frecuencia en el conjunto de datos. De un primer vistazo se observa que las palabras que más sobresalen son aquellas que tienen una referencia directa del contenido en el juego. Por ejemplo, existen muchas instancias que incluyen la palabra *skin*. También existen varias instancias que mencionan alguna temática de artículos cosméticos dentro del juego: *star guardian, spirit blossom, blood moon, etc.* Del mismo modo, mencionan nombres de los personajes seleccionables en el juego: s*eraphine, bel veth, kassadin*.

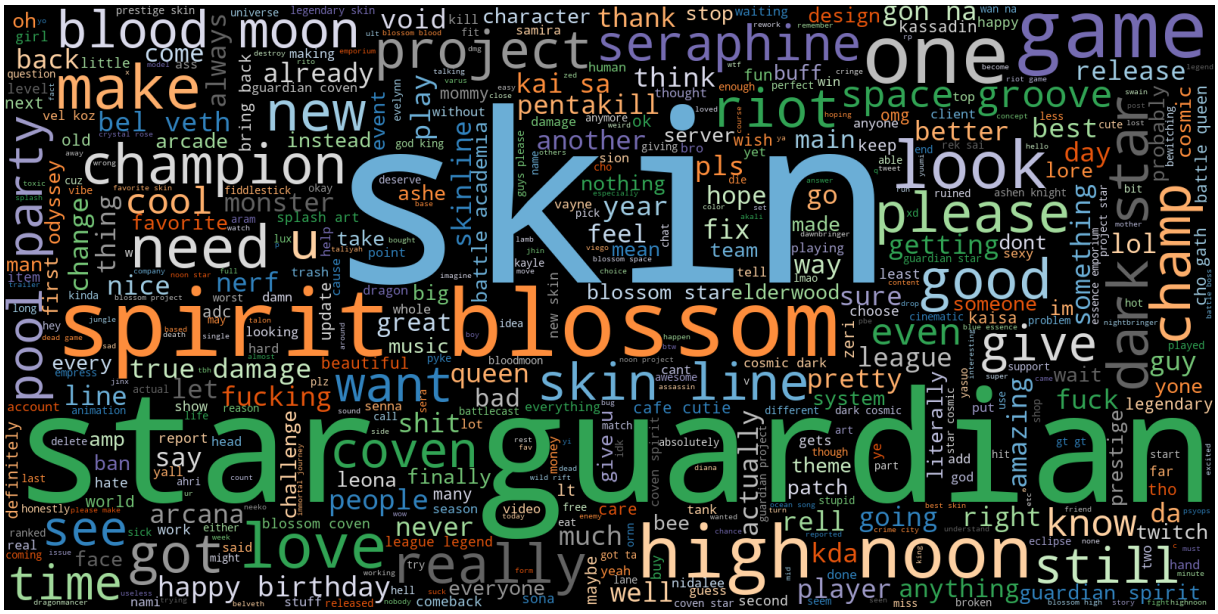

Figura 19: Nube de Palabras – Corpus Completo.

Fuente: Elaboración propia

Dada la alta frecuencia de menciones a contenido del juego, se removieron temporalmente algunas palabras clave. En el Anexo I se encuentran tres listas de palabras que se clasifican según las siguientes categorías: temáticas de *skins,* nombres de campeones y palabras adicionales. Al generar una nueva visualización, se obtiene como resultado la Figura 20.

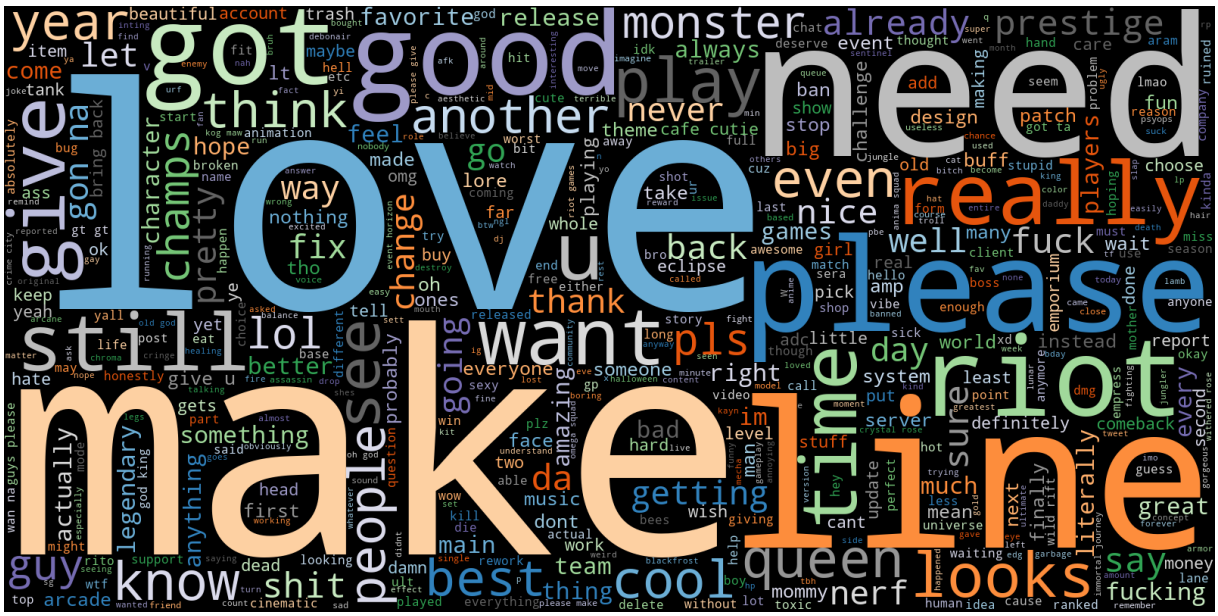

Figura 20: Nube de Palabras - Palabras de referencia al juego eliminadas.

Fuente: Elaboración Propia

Al eliminar las palabras que de alguna forma se refieren a contenido del juego, se tiene una visión más clara del léxico utilizado por los usuarios de Twitter al interactuar con la cuenta del

juego. De forma general, se puede observar que las palabras más utilizadas se encuentran entre: *love, make, good, need* las cuales evocan algo positivo o neutral. Sin embargo, se puede observar la presencia de palabras con una denotación negativa como, por ejemplo: *trash, toxic, fuck, etc.* Al observar detenidamente, poco a poco empiezan a surgir malas palabras y con ellas los primeros indicios de un comportamiento tóxico en el ámbito de las redes sociales. En el siguiente apartado se pone a prueba el clasificador de sentimiento prediseñado VADER, el cual permite observar la distribución general de los *tweets*.

4.3.3 Análisis de Clasificación del modelo Prediseñado (VADER)

Esta etapa se enfoca en analizar cómo se comporta un clasificador de sentimiento prediseñado en el ámbito de las redes sociales y videojuegos. Para esto, se importa el modelo desde la librería *vaderSentiment* y su implementación se describe en el siguiente fragmento de código.

Listado 12: Implementación de VADER.

```
1.  analyzer = SentimentIntensityAnalyzer()
2.  analyzer.polarity_scores(df['text'])
```

Fuente: Elaboración propia.

En la línea 1 del Listado 12, se instancia el clasificador lo cual permite tener acceso a sus métodos y variables desde la variable *analyzer*. En la línea 2 se accede y ejecuta la función *polarity_scores()* la cual recibe como argumento un texto plano. Al ejecutar esta sentencia, se obtiene como resultado un diccionario de Python que incluye cuatro llaves con un valor asignado a cada uno. Como muestra, se toma la primera instancia del conjunto de datos y se observa que sobre el texto: “ugly boring skins” se reciben los valores de la Figura 21:

Figura 21: Resultados de Polaridad.

Fuente: Elaboración propia

Analizando estos resultados se puede determinar que el clasificador encuentra con un 84.8% de precisión de que el mensaje es negativo y por lo tanto se determina que el mensaje tiene una clasificación negativa. El mismo análisis se lo realiza con el resto de las instancias y se procede a realizar una visualización de tarta (Figura 22).

Figura 22: Polaridad de los tweets - VADER.

Fuente: Elaboración propia.

En la Figura 22, se evidencia que existe una división equitativa entre las categorías de positivo y neutral con un valor del 41.9% y 40.4%, respectivamente. Por otro lado, los *tweets* clasificados como negativos alcanzan apenas el 18%. Esto nos lleva a la pregunta ¿Qué palabras ocasionan que la polaridad del texto sea positiva, negativa o neutral? Para analizar de mejor manera, se volvió a realizar una nube de palabras, pero en esta ocasión, una visualización para cada categoría que se discuten a continuación:

Figura 23: Nube de Palabras Positivas - VADER.

Fuente: Elaboración propia.

Al observar la Figura 23 en donde se encuentran las palabras de instancias de *tweets* consideradas como positivas. Si se analiza desde un punto de vista general, se observa que el clasificador realiza un buen trabajo en cuanto la determinación de la polaridad de los *tweets* ya que se pueden encontrar con bastante frecuencia las palabras: *love, spirit, blossom y guardian*. Palabras que traducidas al español significan: amor, espíritu, florecer y guardián. Estas evocan positivismo y se encuentran en su categoría correcta, sin embargo, bajo el contexto del videojuego, estas no pretenden persuadir a los jugadores hacia un sentimiento positivo. Simplemente son nombres propios que se les dan a las temáticas de artículos cosméticos, por ejemplo, *spirit blossom* se traduce a “Flor Espiritual”. Estas temáticas de *skins,* como también los nombres de campeones que se encuentran en la visualización (Yone, Samira, kai sa, etc) en realidad son elementos del juego que deben ser considerados como neutrales.

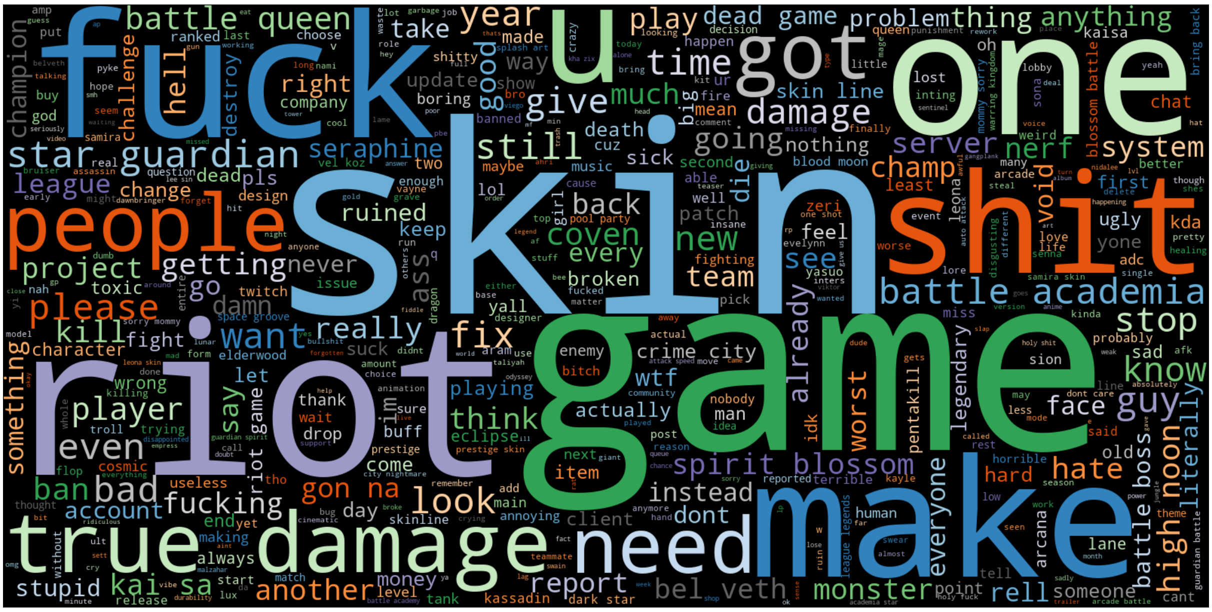

Figura 24: Nube de Palabras Negativas - VADER.

Fuente: Elaboración propia.

En la Figura 24 se visualizan las palabras encontradas en las instancias de *tweets* clasificados como negativos. De la misma manera, se observa que el clasificador hace un buen trabajo en identificar palabras negativas y que son utilizadas con frecuencia en un ámbito que refleja agresividad. Se pueden notar palabras como: *shit, fuck, fucking, stupid.* No obstante, se vuelve a presenciar nombres de líneas de temáticas de *skins* como *true damage,* y nombres de campeones, que opacan los rasgos de toxicidad esperados observar en esta fase del desarrollo del trabajo.

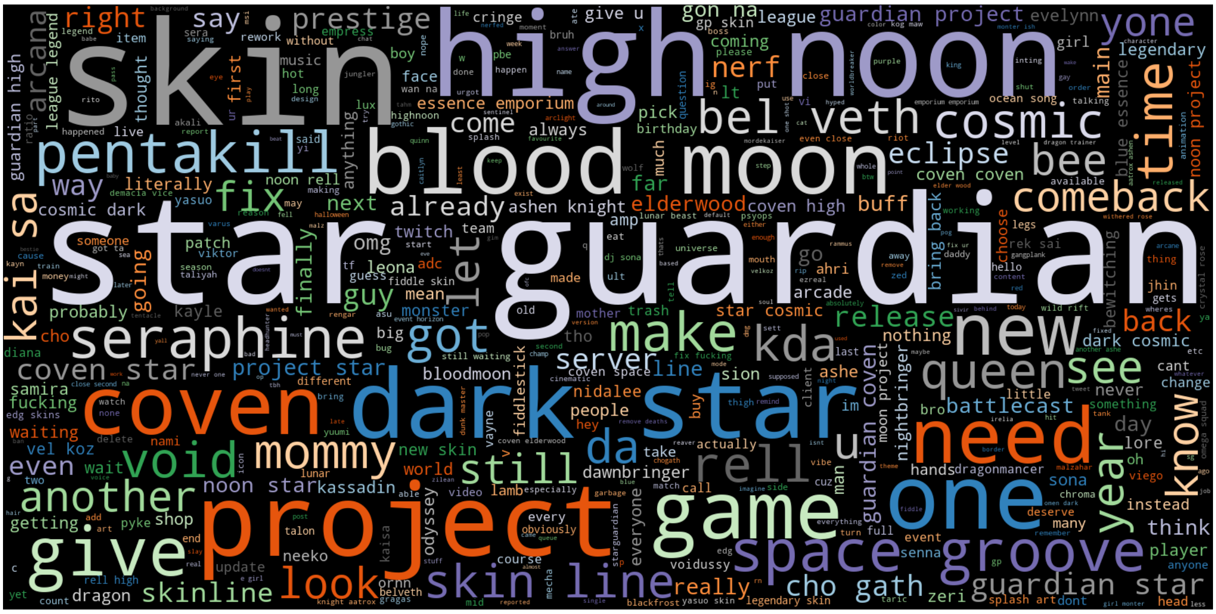

Figura 25: Nube de Palabras Neutrales - VADER.

Fuente: Elaboración propia.

Por último, en la Figura 25 se observan las palabras de instancias de *tweets* asignadas a la categoría neutral. En esta visualización se obtuvo como resultado la clasificación de temáticas de *skins* y nombres de campeones como un elemento neutral. En esta ocasión es más claro identificar los elementos pertenecientes al contenido del videojuego, no obstante, se observa la presencia de malas palabras con una menor frecuencia a la observada en la Figura 24.

A modo de resumen, se utilizó un clasificador de sentimientos prediseñado con el fin de identificar características o rasgos de toxicidad en los mensajes de Twitter. El modelo fue alimentado con instancias de *tweets* que atravesaron un proceso de limpieza y pre-procesamiento de texto que facilitan la clasificación para dicho modelo. Como resultado se obtuvo una clasificación de instancias predominante en la región positiva y neutral. Después de analizar detenidamente las Figuras 23 – 25, no existe una distinción clara entre las categorías. En los tres casos se observan ejemplos de temáticas de skins, nombres de campeones e inclusive, palabras como *skin*s repetidas a lo largo de las 3 categorías. Las palabras pertenecientes al contenido del juego y que forman parte de la jerga común de los jugadores en este ámbito, deben ser consideradas como neutrales. Esto permitiría un análisis más adecuado con respecto a la identificación de toxicidad en los mensajes de Twitter. Para cumplir con este fin, se propone el diseño y entrenamiento de un nuevo modelo de clasificación que se entrene bajo reglas distintas y sea capaz de separar los elementos del juego y rasgos de agresividad.

4.3.4 Diseño y Entrenamiento del modelo

A causa de los impedimentos descritos en la sección anterior, se toma una muestra aleatoria de 3 158 instancias en la base de datos las cuales representan el conjunto de entrenamiento del modelo. Para obtener este valor se realiza el cálculo del valor de la muestra para una población finita dada por la Ecuación 1 en donde N es el tamaño de la población y Z corresponde al valor crítico.

$$n = \frac{NZ_\alpha^2}{e^2(N-1) + Z_\alpha^2}$$

Ecuación 1: Cálculo del tamaño de la muestra.

Fuente: (Saraí Aguilar-Barojas, 2005, p. 5)

Por lo tanto, se procedió a separar de las instancias en una muestra de entrenamiento (632) y otra de validación (2 526) representando el 20% y 80%, respectivamente. En primer lugar, se asigna una clasificación a cada instancia del conjunto de entrenamiento mediante un sistema de votación entre 2 individuos en la que cada uno lee el mensaje y asigna una categoría que corresponde a -1 para negativo, 0 neutral, y 1 positivo. En caso de existir un

desacuerdo en la categoría asignada, se discute para llegar a un consenso. La Figura 26 evidencia el resultado de la votación y, a diferencia con la Figura 22, se observa una distribución de *tweets* orientados hacia una clasificación neutral y negativa.

Figura 26: Polaridad de tweets - Clasificación Manual.

Fuente: Elaboración propia.

Seguidamente, se realizó el mismo tratamiento expuesto en el Capítulo 4.3.2 Limpieza y pre-procesamiento de los datos, con el fin de obtener resultados comparables. Esto quiere decir que se eliminaron las referencias a cuentas de otros usuarios, enlaces externos y se normalizó el texto de todos los *tweets.* Con esta premisa, se plantea diseñar un modelo de clasificación supervisado apoyado en los hallazgos de Kalaivani y Dinesh en su estudio explicado en el Capítulo 2. Como parte del procesamiento de los datos, se utiliza el modelo de frecuencia de términos y frecuencia inversa de documentos o TF-IDF por sus siglas en inglés.

Kalaivani y Dinesh (2020) definen al TF-IDF como «una métrica estadística que se usa para reflejar cuán importante es una palabra en un documento de la colección» (p. 109). TF se refiere número de veces que un término se repite en el documento, mientras que, IDF refleja un peso estadístico usado para medir la importancia del término en el texto del documento y se los calcula de la siguiente forma:

$$TF = 1 + \log(1 + \log(freq(w, D_w))$$

Ecuación 2: Cálculo de TF.

Fuente: Kalaivani, 2020, p. 109

$$IDF = \log(\frac{1+D}{D_w})$$

Ecuación 3: Cálculo de IDF.

Fuente: Kalaivani (2020, p. 109)

En donde *D* es el número total de documentos o instancias y D_W es el número total de términos *w* presentes en el documento. Al combinar estas dos ecuaciones, se puede obtener un valor final como se muestra a continuación:

$$TF - IDF = TF * IDF$$

Ecuación 4: Cálculo de TF-IDF.

Fuente: Kalaivani (2020, p. 109)

Afortunadamente *sklearn,* una librería de Python especializada en los tópicos de *machine learning* e inteligencia artificial, cuenta ya con los métodos necesarios para realizar este cálculo de forma sencilla. Simplemente se instancia un nuevo objeto del tipo `TfidfVectorizer()` como se muestra en línea 1 del Listado 13. De inmediato se utiliza la función `transform()` la cual recibe como argumento el texto plano para ser analizado.

Listado 13: Implementación de TF-IDF.

```
1.  Tfidf_vect = TfidfVectorizer(max_features=5000)
2.  vect = Tfidf_vect.transform(text)
```

Fuente: Elaboración propia

En la Tabla 9 se muestra un ejemplo de cinco palabras con su respectivo valor de TF-IDF. Marius Borcan (2020) explica que la filosofía de esta métrica se fundamenta bajo dos observaciones. La primera, una palabra que aparece en un documento frecuentemente tiene más relevancia para dicho documento, es decir, que existe más probabilidad de que el documento se trate acera esa palabra. La segunda, es que una palabra que aparece frecuentemente en varios documentos pierde relevancia. En otras palabras, TF-IDF es un valor que se calcula para todas las palabras de todos los documentos que aumenta con cada aparición dentro de un documento, pero decrece gradualmente con cada aparición en otros documentos.

Tabla 9: Valor de TF-IDF para 5 palabras.

Palabra	TF-IDF
Legendary	0.393297
Always	0.346587
Get	0.279653
Skins	0.262165
Mains	0.227522

Fuente: Elaboración propia.

Una vez realizado este proceso, se procede a crear un modelo de clasificación supervisada. En el Listado 14 se visualiza una búsqueda de hiperparámetros que generan la mejor precisión. En la línea 1, se crea una lista de los posibles *kernels* que el algoritmo acepta como parámetro, y en la línea 2 se crea una lista de la fórmula que se utiliza para el cálculo del coeficiente que utilizará para la clasificación. Después, desde la línea 5 hasta la 19, se crean dos iteradores que se aplicarán cada una de las opciones de la definición del modelo y se realiza una predicción.

En la línea 7, se muestra el uso de la librería de *sklearn* nuevamente para instanciar el modelo de clasificación de SVM y en la línea 8 se le entrega las instancias y sus valores correspondientes para el entrenamiento a la función *fit()*. Luego, en la línea 9 se utiliza el método *predict()* con las instancias separadas previamente para medir la precisión del modelo de clasificación. En la línea 13 se obtiene el valor correspondiente a la precisión y se almacena la información del *kernel* y *gamma.* En la línea 20, se almacenan los parámetros y el resultado para su posterior análisis.

Listado 14: Búsqueda de Hiperparámetros.

```
1.  kernels = ['linear', 'poly', 'rbf', 'sigmoid']
2.  gamma = ['scale', 'auto']
3.  results = pd.DataFrame()
4.  for k in kernels:
5.      for g in gamma:
6.          model = SVC(kernel=k, gamma=g)
7.          model.fit(train_x_vect, train_Y)
8.          predictions_SVM = model.predict(test_x_vect)
9.          test_prediction = pd.DataFrame()
10.         test_prediction['text'] = test_X
11.         test_prediction['Label'] = predictions_SVM
12.         SVM_accuracy = accuracy_score(predictions_SVM, test_Y)*100
13.         SVM_accuracy = round(SVM_accuracy,2)
14.         dict = {
15.             'Kernel': k,
16.             'gamma': g,
17.             'accuracy': SVM_accuracy,
18.         }
19.         results = results.append(dict, ignore_index=True)
```

Fuente: Elaboración propia

En la Figura 27, se visualizan los resultados obtenidos al ejecutar el código del Listado 14. La precisión más alta, corresponde a los parámetros: (*Kernel, linear)* y *(gamma, scale|auto)*. Estos valores son los escogidos para la definición final del modelo.

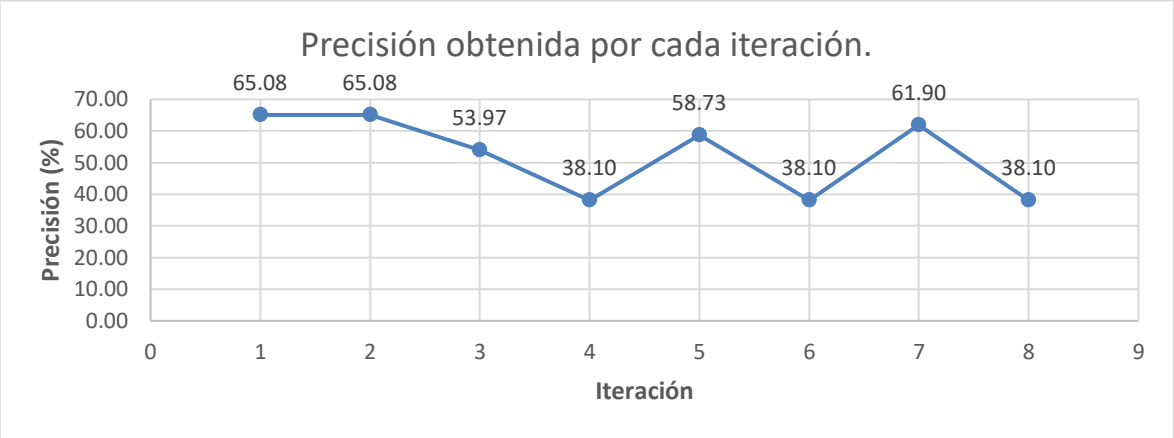

Figura 27: Precisión obtenida en cada iteración.

Fuente: Elaboración propia.

Para evaluar el modelo de mejor forma, se procede a realizar tres nuevas visualizaciones utilizando la clasificación asignada a cada instancia.

Figura 28: Nube de Palabras Positivo - Modelo SVM.

Fuente: Elaboración propia.

En la Figura 28, se visualiza una pequeña agrupación de palabras con referencia a elementos positivos por ejemplo las palabras: felicidad, amor y cumpleaños. Sin embargo, aún se puede observar la presencia de nombres de campeones que el clasificador se encuentra considerando como positivos. Sin embargo, en la Figura 29 se puede notar una presencia de

un mayor número de palabras al compararlas con la Figura 28 y 30, sobre todo, de palabras negativas. Un aspecto que apenas se lograba observar con el modelo de clasificación anterior en la Figura 24. En esta ocasión es claro como las conversaciones de los usuarios en la comunidad de *League Of Legends* recurren a la agresividad. Por ejemplo, aparte de las palabras ya mencionadas cuando se realizó el análisis de la Figura 24, se evidencian nuevas como: *thrash*, *fart*, *shitters* y entre otras. Estos nuevos hallazgos permiten comprobar que el comportamiento tóxico de los jugadores se evidencia en las redes sociales y, que adicionalmente, son críticas hacia el juego y su contenido. Finalmente, en la Figura 30 se observa el acercamiento esperado al implementar el nuevo modelo. Las palabras plasmadas en la visualización son elementos o referencias a los artículos cosméticos del videojuego o a nombres de campeones de este. Además, se puede inferir que existe un fanatismo por la línea *skins* relacionadas a las estrellas como, por ejemplo, *star guardians.*

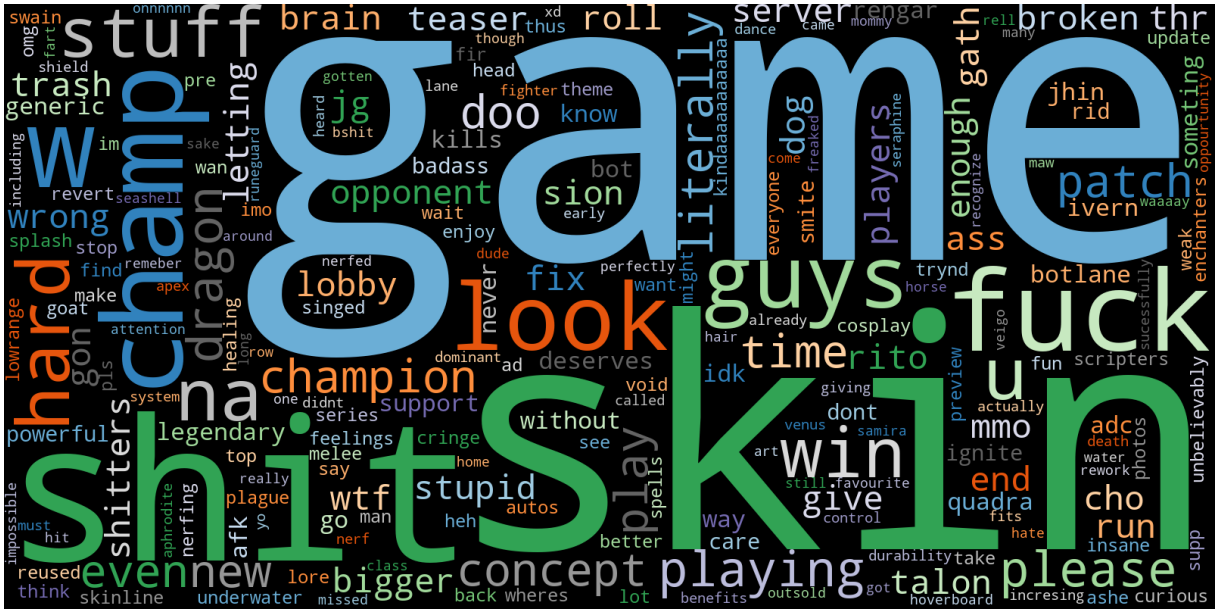

Figura 29: Nube de Palabras Negativo - Modelo SVM.

Fuente: Elaboración propia.

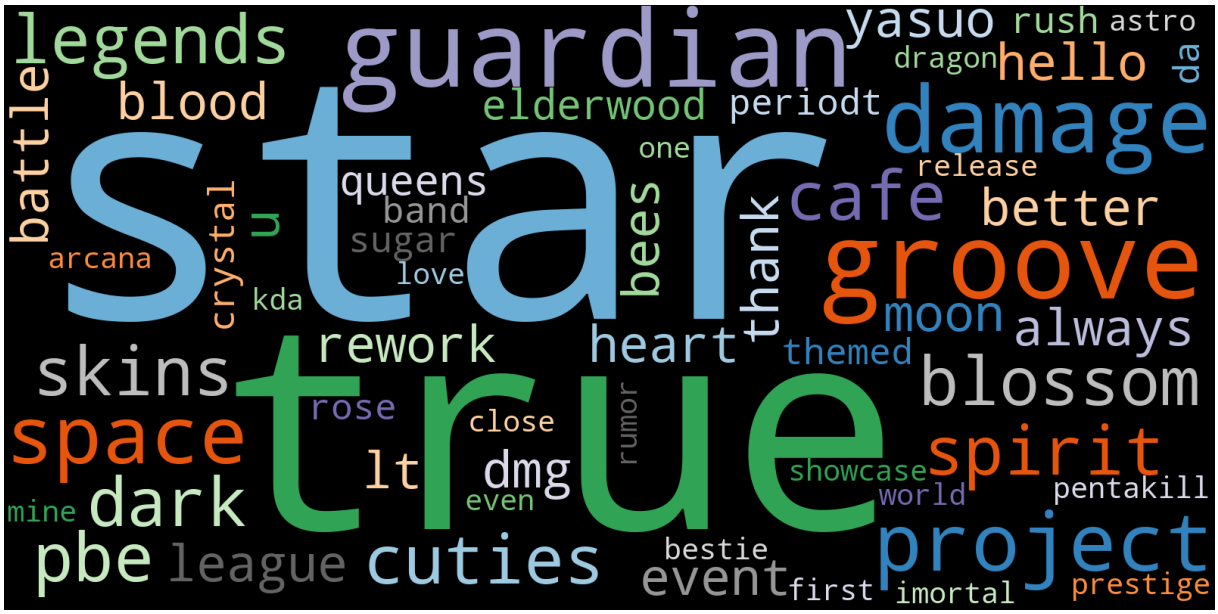

Figura 30: Nube de Palabras Neutral - Modelo SVM.

Fuente: Elaboración propia.

Recapitulando, se diseñó e implemento un modelo de clasificación supervisado bajo nuevas reglas que le permite identificar la polaridad de instancias de mensajes de Twitter. Para ello, el modelo utilizó una muestra aleatoria del conjunto de datos a la cual se le asignó una clasificación manualmente. Una vez entrenado, se procedió a graficar nuevas nubes de palabras en las que se pudo comprobar la presencia de comportamientos tóxicos, ya que el modelo es capaz de discernir palabras que tienen el contexto de contenido en el juego de palabras que en verdad evocan positivismo o negativismo.

5. Conclusiones y trabajo futuro

En este apartado se presentan y discuten las conclusiones del trabajo. Se elabora un breve recuento de los objetivos planteados, como fueron alcanzados y se presentan limitaciones encontradas a lo largo del desarrollo. Posteriormente, se presentan posibles lineamientos para trabajo futuro.

5.1. Conclusiones

En el trabajo se realizó un modelo de recolección y limpieza de datos que permite alimentar un algoritmo de *Machine Learning* capaz de analizar y clasificar la polaridad de mensajes emitidos por usuarios mediante Twitter. La motivación del trabajo se fundamenta en primer lugar, en observar el comportamiento de la comunidad de *League Of Legends* en un espacio virtual externo al del videojuego, y, en segundo lugar, en generar un modelo de clasificación de sentimientos capaz de identificar el comportamiento tóxico en los mensajes de Twitter. El clasificador alcanzo un 65.08% de precisión y con ello, se ha logrado un avance interesante en cuanto a exploración y análisis antropológico de la comunidad de *League of Legends*.

Como fue presentado en el Capítulo 2, existen varios autores que se encuentran interesados por la línea de análisis del procesamiento del lenguaje natural y el análisis de sentimiento, por lo cual, han propuesto varios modelos. No obstante, existe poco interés por la exploración del comportamiento dentro de las comunidades de juegos en línea. De todos los modelos, los autores Kalaivani y Dinesh (2020) presentaron un análisis acerca de respuesta emocional ante un evento utilizando técnicas de clasificación supervisada, la cual se acerca a la exploración del presente trabajo, pero, en un ámbito político.

En cuanto al desarrollo del trabajo, se puede concluir que se han alcanzado de forma exitosa los objetivos planteados. La investigación y elaboración del estado del arte permitieron tener un mayor entendimiento sobre la temática y acerca de técnicas de tratamiento de datos y su análisis. Luego, se cumplió con la elaboración de un modelo de bases de datos que permite no solo analizar la polaridad de los mensajes de Twitter, sino, expandir la investigación e incorporar temas relacionados al compromiso de los usuarios, la fidelización y, sobre todo, las interacciones que se dan entre ellos. En consecuencia, se logró cumplir con el siguiente objetivo de extraer y almacenar los metadatos de *tweets* utilizando el modelo COSA-DLC. Esto permitió tener un flujo claro sobre el camino de la información, desde que se realizan las peticiones hasta que llegan a manos del analista, además, es un modelo que puede ser usado de forma continua.

Para la segunda fase, se puede concluir que el análisis exploratorio de los datos permitió analizar la interactividad y el compromiso de la comunidad con la cuenta de *League Of Legends* a parte de brindar una clara visón sobre la naturaleza de los datos. Seguido, se utilizaron técnicas de limpieza y preprocesamiento de datos útiles para facilitar el trabajo del clasificador y eliminar elementos que causen un ruido innecesario para el análisis. Posteriormente, y para evidenciar el cumplimiento de los últimos dos objetivos restantes, se concluye sobre los modelos de clasificación.

En primera instancia, se entregó el conjunto de datos al primer clasificador de sentimiento. Como resultado (Figura 22) el modelo había clasificado las instancias asignando la mayoría en las categorías positiva y neutral. Sin embargo, al observar cada una de las palabras encontradas en cada categoría se observó que no existe una distinción clara entre las categorías. Tal parece, que el clasificador asignó las instancias de manera aleatoria. Bajo este ámbito, se concluye que el clasificador prediseñado no se encontraba preparado para analizar textos de mensajes con contenido de videojuegos. Existen instancias de palabras que se repetían a lo largo de las tres categorías sin una aparente justificación de dicha asignación. Por otro lado, se logró entrenar un nuevo modelo de clasificación capaz de distinguir elementos clave de contenido del videojuego tras proporcionar nuevas reglas de clasificación. Al crear un conjunto de datos con una asignación de categoría previa, le permitió al modelo realizar un mejor trabajo. En la Figura 29, se puede observar de forma clara más elementos con un significado turbio o que apuntan hacia un comportamiento tóxico de la comunidad. Adicionalmente, se concluye que el análisis permite conocer la opinión pública de los jugadores respecto al videojuego como tal, la nube de palabras en la Figura 30 indica una clara preferencia por una temática de artículos cosméticos, información que puede ser de bastante valor para el continuo desarrollo y mejora del juego.

Por último, es importante mencionar que se enfrentaron a algunas limitaciones durante el desarrollo del trabajo. Primero, las limitaciones de la API de Twitter ocasionaron un retraso en cuanto al periodo de desarrollo ya que se esperaba extraer un conjunto de datos mucho mayor. Segundo, debido a que se extrajo una muestra aleatoria, la clasificación manual de las instancias no fue balanceada, es decir, no se obtuvo 33.3% de instancias neutrales, 33.3% positivas y 33.3% negativas. Por consecuencia, el clasificador tuvo pocos elementos en la categoría positiva para aprender de ellos. Finalmente, el acontecimiento del 3 de mayo del 2022 generó una alta cantidad de respuestas orientadas a un elemento en específico del juego, las *skins*, esto provoco que exista un mayor contenido de información neutral e irrumpió en el flujo de interacciones regular de la cuenta.

5.2. Líneas de trabajo futuro

Con respecto al trabajo futuro, se puede concluir que se deben recolectar datos por un periodo más prolongado de tiempo, al menos un año. En este periodo se debe trabajar con un equipo de más personas para realizar la clasificación manual de los tweets de forma más acertada. Esto ayudaría a mejorar el modelo de clasificación ya que contaría con un número mayor de instancias y un amplio conocimiento del material del juego. De la misma manera, se propone construir un proceso de mejora del modelo de clasificación iterativo, con el fin de que sea capaz de aprender nuevos conceptos introducido por el juego y por los jugadores. Esto le permitiría está aprendiendo constantemente y determinando cual es la opinión real de la comunidad.

Los resultados obtenidos en este análisis pueden ser el inicio de varias aplicaciones a nivel empresarial. El modelo construido puede ser utilizado para obtener información acerca del estado actual del juego y ayudaría a en la toma de decisiones para la empresa *Riot Games.* Adicionalmente, es una herramienta que permite tener un control sobre la cantidad de toxicidad encontrada actualmente. Un avance a largo plazo sería el de poder incluir una herramienta en la plataforma de Twitter que utilice los mensajes de interacción con la cuenta y analice en tiempo real los distintos comportamientos y reacciones que se tiene con cada respuesta. Finalmente, durante el proceso de recolección de datos, se diseñó un modelo de bases de datos que permite almacenar diversos metadatos de los *tweets*, a modo de trabajo futuro se propone realizar una investigación acerca de la capacidad de influencia que tiene la cantidad de *likes* en un *tweet*.

Bibliografía

Anupama, B. et al. (2020). *Real Time Twitter Sentiment Analysis using Natural Language Processing.* International Journal of Engineering Research & Technology. Vol. 9: 07.

Beres, N., Frommel, J., Reid, E., Mandryk, R., Klarkowski, M. (2021). *Don't You Know That You're Toxic: Normalization of Toxicity in Online Gaming*. Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, Article 438, 1–15. https://doi.org/10.1145/3411764.3445157

Borcan, M. (2020). *TF-IDF explained and python sklearn implementation*. Medium. Recuperado junio 24, 2022, de https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275

Calderon, P. (2017). *VADER Sentiment Analysis Explained.* Medium. Recuperado julio 4, 2022, de https://medium.com/@piocalderon/vader-sentiment-analysis-explained-f1c4f9101cd9

Canales, J. L. (2022). *Top programming languages for Data Scientists in 2022*. DataCamp. Recuperado junio 26, 2022, de https://www.datacamp.com/blog/top-programming-languages-for-data-scientists-in-2022

Christopher, A. (2021). *Python vs R for data science projects*. Medium. Recuperado junio 26, 2022, de https://medium.com/analytics-vidhya/python-vs-r-for-data-science-projects-7a56a26c2e4b

Computer History Museum. (s.f.). Timeline of Computer History. 1981 | Timeline of Computer History. Retrieved July 8, 2022, from https://www.computerhistory.org/timeline/1981/

DataFlair. (2021). Pros and cons of R programming language - unveil the essential aspects! DataFlair. Recuperado junio 26, 2022, de https://data-flair.training/blogs/pros-and-cons-of-r-programming-language/

Giachanou A., Crestani, F. (2016). *Like It or Not: A Survey of Twitter Sentiment Analysis Methods*. ACM Comput. Surv. 49, 2, Articulo 28 (junio 2017), 41 páginas. https://doi.org/10.1145/2938640

Glancey, P. (1996*). Computer and Video Games: Then and Now*. In The Complete History of Computer and Video Games. EMAP IMAGES. (Vol. 1, Pág. 04–10).

Gupta, M., Vishwakarma, G., Badhani, P. (2017). *Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python.* International Journal of Computer Applications NY USA:Foundation of Computer Science (FCS). Vol. 165. Pág. 29-34.

History.com Editors. (2017, September 1). *Video game history*. History.com. Recuperado mayo 10, 2022, de https://www.history.com/topics/inventions/history-of-video-games#section_6

IBM Cloud Education. (2019). *Cap Theorem*. IBM. Recuperado junio 24, 2022, de https://www.ibm.com/ar-es/cloud/learn/cap-theorem

IBM Cloud Team. (2021). *Python vs. R: What's the difference?* IBM. Recuperado junio 26, 2022, de https://www.ibm.com/cloud/blog/python-vs-r

Kalaivani, P., Dinesh, D. (2020). *Machine Learning Approach to Analyze Classification Result for Twitter Sentiment*. International Conference on Smart Electronics and Communication (ICOSEC). Pág. 107-112.

Kashfia, S. (2018). *Emotion and Sentiment Analysis from Twitter Text.* University of Calgary.

Kent, S. L. (2021). *The ultimate history of video games* v*olumen 2: Nintendo, Sony, Microsoft, and the billion-dollar battle to shape modern gaming.* Vol. 2. Crown.

Kharde, V., Sonawane, S. (2016). *Sentiment Analysis of Twitter Data: A Survey of Techniques.* International Journal of Computer Applications 139(11): Pag. 5-15. https://doi.org/10.48550/arXiv.1601.06971

Kowert R (2020) *Dark Participation in Games*. Psychol. 11:598947. 10.3389/fpsyg.2020.598947

Laroia, G., Scurato, C. (2019). *Toxic Twitter: The state of hateful activities on the platform.* Free Press.

League Of Legends (@LeagueOfLegends). (s.f). Perfil de Twitter. Recuperado junio 24, 2022, de https://twitter.com/leagueoflegends?s=21&t=R69hLsbE2hT1GYJ_TmiGUg

League Of Legends (@LeagueOfLegends). (s.f). Perfil de Twitter. Recuperado junio 24, 2022, de https://twitter.com/LeagueOfLegends/status/1521519967966924800

League Of Legends (@LeagueOfLegends). (s.f). Perfil de Twitter. Recuperado junio 24, 2022, de https://twitter.com/LeagueOfLegends/status/1527725849797074945

League Of Legends (@LeagueOfLegends). (s.f). Perfil de Twitter. Recuperado junio 24, 2022, de https://twitter.com/LeagueOfLegends/status/1529122550415958017

Liu, Y. Ma, Y. Wang, T. (2022). *The Spread of League of Legends.* Atlantis Press. https://doi.org/10.2991/assehr.k.220110.026

Loshin, P., & Sirkin, J. (2022). What is structured query language (SQL)? SearchDataManagement. Recuperado junio 26, 2022, de https://www.techtarget.com/searchdatamanagement/definition/SQL

Märtens, M., Shen, S., Iosup, A., & Kuipers, F. A. (2016). *Toxicity detection in multiplayer online games*. In 2015 International Workshop on Network and Systems Support for Games, NetGames 2015, Zagreb, Croatia, diciembre 3-4, 2015 (Vol. 2016-Enero, Pág.

1-6). [7382991] ACM, IEEE Computer Society. https://doi.org/10.1109/NetGames.2015.7382991

MongoDB, Inc. (2022). *NoSQL vs SQL databases*. MongoDB. Recuperado junio 24, 2022, de https://www.mongodb.com/nosql-explained/nosql-vs-sql

Naumenko, V. (2022). *Best 11 data science programming languages in 2022.* Jelvix. Recuperado junio 26, 2022, de https://jelvix.com/blog/top-data-science-programming-languages

Neha et al. (2021). *Twitter Sentiment Analysis using Deep Learning.* IOP Conf. 10.1088/1757-899X/1022/1/012114

Oracle. (2022). *What is a database?* Oracle. Recuperado junio 26, 2022, de https://www.oracle.com/database/what-is-database/#link5

Python Software Foundation. (2022*). Welcome to Python.org*. Python.org. Recuperado junio 26, 2022, de https://www.python.org/about/

Ranganathan, J., Tzacheva, A. (2019). *Emotion Mining in Social Media Data.* Procedia Computer Science. Vol. 159. Pág. 58-66.

Saraí Aguilar-Barojas. (2005). Fórmulas para el cálculo de la muestra en investigaciones de salud. Secretaría de Salud del Estado de Tabasco. Vol. 11. Pág. 333-338.

Sinaeepourfard, A.,Garcia, J., Masip-Bruin, X., Marín-Tordera, E. (2016). *Towards a Comprehensive Data LifeCycle Model for Big Data Environments*. IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT), Pág. 100-106.

The R Foundation. (2022). *What is R?* R. Recuperado junio 26, 2022, de https://www.r-project.org/about.html

Twitter Developer Platform. (2022). Twitter's search endpoints | docs | twitter developer platform. Twitter. Retrieved May 22, 2022, from https://developer.twitter.com/en/docs/twitter-api/search-overview

Watson, M. (2015). *A medley of meanings: Insights from an instance of gameplay in League of Legends*. Journal of Comparative Research in Anthropology and Sociology. Volumen 6. Pág. 225-243. ISSN 2068 – 0317.

Anexo I. Listas de Palabras Eliminadas Temporalmente (Análisis de nube de palabras)

```
skins_themes_list = [
    'skin',
    'skins',
    'spirit',
    'blossom',
    'star',
    'guardian',
    'high',
    'noon',
    'blood',
    'moon',
    'pool',
    'party',
    'space',
    'groove',
    'guardian',
    'spirit',
    'coven',
    'true',
    'damage',
    'elderwood',
    'guardians',
    'project',
    'battle',
    'academia',
    'bloodmoon',
    'cosmic',
    'dragonmancer',
    'highnoon',
    'nightbringer',
    'dawnbringer',
    'skinline',
    'void',
    'dragon',
    'bee',
    'odyssey',
    'battlecast',
    'demacia',
    'vice',
    'kda',
    'arcana',
    'bewitching',
    'ocean',
    'song',
    'ashen',
    'knight',
    ''

    ]

champ_list = [
    'Aatrox',
    'Ahri',
    'Akali',
    'Alistar',
    'Amumu',
    'Anivia',
    'Annie',
    'Ashe',
    'Azir',
    'Blitzcrank',
    'Brand',
    'Braum',
```

```
64.     'Caitlyn',
65.     'Cassiopeia',
66.     'Cho\'Gath',
67.     'Corki',
68.     'Darius',
69.     'Diana',
70.     'Dr. Mundo',
71.     'Draven',
72.     'Elise',
73.     'Evelynn',
74.     'Ezreal',
75.     'Fiddlesticks',
76.     'Fiora',
77.     'Fizz',
78.     'Galio',
79.     'Gangplank',
80.     'Garen',
81.     'Gnar',
82.     'Gragas',
83.     'Graves',
84.     'Hecarim',
85.     'Heimerdinger',
86.     'Irelia',
87.     'Janna',
88.     'Jarvan IV',
89.     'Jax',
90.     'Jayce',
91.     'Jinx',
92.     'Kalista',
93.     'Karma',
94.     'Karthus',
95.     'Kassadin',
96.     'Katarina',
97.     'Kayle',
98.     'Kennen',
99.     'Kha\'Zix',
100.      'Kog\'Maw',
101.      'LeBlanc',
102.      'Lee Sin',
103.      'Leona',
104.      'Lissandra',
105.      'Lucian',
106.      'Lulu',
107.      'Lux',
108.      'Malphite',
109.      'Malzahar',
110.      'Maokai',
111.      'Master Yi',
112.      'Miss Fortune',
113.      'Mordekaiser',
114.      'Morgana',
115.      'Nami',
116.      'Nasus',
117.      'Nautilus',
118.      'Nidalee',
119.      'Nocturne',
120.      'Nunu',
121.      'Olaf',
122.      'Orianna',
123.      'Pantheon',
124.      'Poppy',
125.      'Quinn',
126.      'Rammus',
127.      'Rek\'Sai',
128.      'Renekton',
129.      'Rengar',
130.      'Riven',
131.      'Rumble',
132.      'Ryze',
133.      'Sejuani',
```

```
    'Seraphine',
    'Shaco',
    'Shen',
    'Shyvana',
    'Singed',
    'Sion',
    'Sivir',
    'Skarner',
    'Sona',
    'Soraka',
    'Swain',
    'Syndra',
    'Talon',
    'Taric',
    'Teemo',
    'Thresh',
    'Tristana',
    'Trundle',
    'Tryndamere',
    'Twisted Fate',
    'Twitch',
    'Udyr',
    'Urgot',
    'Varus',
    'Vayne',
    'Veigar',
    'Vel\'Koz',
    'Vi',
    'Viktor',
    'Vladimir',
    'Volibear',
    'Warwick',
    'Wukong',
    'Xerath',
    'Xin Zhao',
    'Yasuo',
    'Yorick',
    'Zac',
    'Zed',
    'Ziggs',
    'Zilean',
    'Zyra',
    'rell',
    'kaisa',
    'yone',
    'zeri',
    'senna',
    'vel koz',
    'jhin',
    'kai',
    'sa',
    'vel',
    'koz',
    'rek',
    'sai',
    'bel',
    'veth',
    'neeko',
    'yuumi',
    'cho',
    'gath',
    'samira',
    'viego',
    'belveth',
    'taliyah',
    'ornn',
    'kindred',
    'pyke',
    'kog maw'
]
```

```
204.
205.  words_to_remove = [
206.      'splash',
207.      'art',
208.      'champ',
209.      'game',
210.      'player',
211.      'league',
212.      'legend',
213.      'legends',
214.      'happy',
215.      'birthday',
216.      'champions',
217.      'champion',
218.      'pentakill',
219.      'dark',
220.      'blue',
221.      'escence',
222.      'essence',
223.      'new',
224.      'look',
225.      'one',
226.  ]
227.
```