Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Viñas Templado, Carla; César Galobardes, Eduardo, dir. Uso de técnicas de Machine Learning y Deep Learning para la generación de perfiles de usuarios de Twitter. 2023. (Enginyeria de Dades)

This version is available at https://ddd.uab.cat/record/281537

under the terms of the BY-NC-ND license

Uso de técnicas de *Machine Learning* y *Deep Learning* para la generación de perfiles de usuarios de Twitter

Carla Viñas Templado

Resumen– La simulación a partir de agentes es un proyecto que se está desarrollando en la universidad Autónoma de Barcelona y tiene como objetivo llegar a realizar simulaciones cada vez más complejas. La primera simulación que se quiere llevar a cabo es la de agentes siendo estos usuarios de Twitter que interactuan. El objetivo de estas simulaciones es permitir estudiar cómo se propagan las ideas, cómo se forman las comunidades y cómo se difunden las noticias y la información. Este proyecto surge en este escenario y busca capturar las características y la distribución de los diferentes grupos de usuarios que pueden existir en la red social de Twitter, con el propósito de poder generar los agentes para hacer simulaciones lo más realistas possibles. Por lo tanto, la finalidad de este trabajo es llegar a identificar los diferentes posibles grupos de usuarios existentes en la red social. Y para llegar a dicho objetivo se ha planteado caracterizar al usuario en dos niveles distintos. Por una parte, encontrando la distribución de los grupos originados junto con las características que los forman y por otra parte, obteniendo el contenido que escriben en sus *tweets*. Para ello se ha utilizado técnicas de *Machine Learning* y *Deep Learning*, respectivamente. Con el propósito final de conseguir tanto la información del tipo de cuenta cómo el contenido que esta ofrece.

Palabras clave– Simulación, agentes, Twitter, formación de comunidades, distribución, grupos de usuarios, simulaciones realistas, identificación de grupos, caracterización de usuarios, Machine Learning, Deep Learning.

Abstract– The agent-based simulation is a project being developed at the Autonomous University of Barcelona with the aim of conducting increasingly complex simulations. The first simulation to be conducted is that of agents who are Twitter users interacting with each other. The purpose of these simulations is to study how ideas propagate, how communities form, and how news and information spread. This project seeks to capture the characteristics and distribution of the different groups of users that can exist on the Twitter social network in order to generate agents for simulations that are as realistic as possible. The ultimate goal of this project is to identify the different possible groups of users existing on the social network. To achieve this objective, the user is characterized at two different levels. On the one hand, obtaining the distribution of the groups that have originated along with the characteristics that form them, and on the other hand, obtaining the content they write in their tweets. Techniques of Machine Learning and Deep Learning are used, respectively, with the ultimate purpose of obtaining both information on the type of account and the content it offers.

Keywords– Simulation, agents, Twitter, community formation, distribution, user groups, realistic simulations, group identification, user characterization, Machine Learning, Deep Learning.

1 Introducción

Actualmente, uno de los proyectos que se está desarrollando en la Universidad Autónoma de Barcelona es la simulación a partir de agentes. El propósito que se tiene es poder llegar a realizar simulaciones cada vez más y más complejas. Pero para poder llegar a

• E-mail de contacto: 1564580@uab.cat
• Trabajo tutorizado por: Eduardo César Galobardes (Arquitectura de Computadores y Sistemas Operativos)
• Curso: 2022/23
• Repositorio GitLab: https://gitlab.com/cvinaste/tfg.git

tal complejidad, primero se tiene que partir de simulaciones más sencillas. Es por este motivo, que la simulación que se está llevando a cabo es la de agentes como usuarios de una red social que interactúan, siendo Twitter la red social escogida.

Las redes sociales han revolucionado la forma en que las personas se comunican, interactúan y comparten información. Estas plataformas en línea han creado una gran cantidad de datos que se generan a diario, y han dado lugar a una enorme cantidad de posibilidades para analizar el comportamiento humano en línea.

Es por ello por lo que, las simulaciones de redes sociales se han convertido en una herramienta poderosa para comprender el comportamiento social y las dinámicas que ocurren en las redes. Estas permiten a los investigadores y analistas estudiar cómo se propagan las ideas, cómo se forman las comunidades y cómo se difunden las noticias y la información. Otorgando la posibilidad de observar cómo la puesta en marcha de una nueva política puede causar un tremendo impacto en el funcionamiento de la red. Al modelar y simular diferentes escenarios, los investigadores y los tomadores de decisiones pueden evaluar cómo estas políticas pueden afectar el comportamiento de los usuarios y el rendimiento de la red social, y tomar decisiones más informadas y efectivas en consecuencia.

En este contexto nace este proyecto, el cual pretende capturar las características (perfiles) y la distribución de los grupos de usuarios de Twitter para poder realizar simulaciones lo más realistas posibles.

El objetivo principal del trabajo es ser capaz de definir el conjunto y distribución de perfiles de usuarios de Twitter utilizando técnicas de *Machine Learning* (ML) y *Deep Learning* (DL).

Y para alcanzarlo, se han planteado los diferentes objetivos más específicos:

- Obtener una base de datos de usuarios de Twitter con un número suficiente de registros.
- Determinar la dimensión de la caracterización del usuario, es decir, en qué ámbitos se quiere hacer la clasificación de estos.
- Analizar cuales son los atributos más representativos de la base de datos generada.
- Identificar los distintos perfiles de usuarios de Twitter utilizando técnicas de ML y DL.

2 Metodología

Se ha seguido la metodología Scrum, la cual se basa en un enfoque *Agile*[1], para alcanzar los objetivos establecidos.

Los pasos realizados han sido: adquisición de la fuente de datos, exploración de los datos, procesamiento de estos para obtener el formato correcto, formación de *clusters* utilizando técnicas de *Machine Learning*, elaboración de gráficos para facilitar el análisis de los grupos formados y observación de los géneros más escritos por los usuarios de cada grupo utilizando un modelo de *Deep Learning*.

[1]Filosofía de ciclos de trabajo cortos, donde se centra en la mejora gradual continua mediante publicaciones pequeñas y frecuentes.

Todas estas tareas se han llevado a cabo en reuniones semanales dónde se han revisado los avances realizados y se han establecido los objetivos para la siguiente semana.

3 Desarrollo

3.1 Base de datos

3.1.1 Adquisición de los datos

Para la adquisición de los datos se ha generado un dataset propio accediendo a la API de Twitter (Tweepy).

Para poder llevarlo a cabo, se ha programado un *crawler*[2] para obtener un sub-grafo de usuarios de la red [1] [2]. Para ello se ha utilizado la API de Twitter con la librería de Python Tweepy [3] [4]. Esta librería permite realizar distintas consultas; desde información pertenecientes a datos de usuarios [5], hasta tweets que ellos mismos han publicado [6].

La red se ha explorado a partir de un usuario semilla utilizando la exploración *Breadth-First Search (BFS)*[3]. Este tipo de exploración permite, por un lado, explorar totalmente las conexiones de los usuarios (seguidores y seguidos), y por otro, obtener el subgrafo más cercano al nodo. De esta forma, se ha podido capturar un contexto real del área de usuarios cerca de ese usuario semilla.

También, se han incorporado dos límites a la ejecución:

- Número de usuarios que el crawler explorará totalmente (condición de parada), 25.
- Número máximo de conexiones que un nodo no puede superar para realizar su exploración, 10000.

Los límites se han añadido para tener un control en la ejecución.

3.1.2 Almacenaje de los datos

Para poder tratar los datos de forma versátil, se ha realizado el almacenaje en formato grafo de NetworkX de Python [7].

Esta librería ofrece la posibilidad de crear grafos dirigidos, donde los nodos son los usuarios de la red, y las aristas las conexiones entre éstos (seguidores y seguidos). Asimismo, permite incorporar atributos a los nodos, que son los datos de los usuarios del medio social.

Por tanto, permite obtener un grafo con nodos (usuarios) que contienen atributos (datos de los usuarios) y estos están conectados por aristas con direcciones dependiendo si son seguidores o seguidos del nodo.

3.1.3 Unión de los datos

Para realizar la adquisición de los datos, se han utilizado diferentes semillas de partida junto a diferentes credenciales de acceso a la API. Esto ha sido debido a que la API de

[2]Algoritmo usado para analizar el código de un sitio web en busca de Informaciones.

[3]Algoritmo utilizado para recorrer o buscar un grafo o árbol en orden de anchura. El algoritmo empieza en un nodo raíz y explora todos los nodos adyacentes del mismo nivel antes de pasar a los nodos de profundidad siguiente, de esta manera, examina todos los nodos de un nivel antes de pasar al siguiente nivel.

Tweepy tiene varias restricciones; una de ellas es el límite de llamadas. Twitter limita el número de llamadas que se pueden hacer a su API en un período de tiempo determinado. Si un desarrollador supera los límites de llamadas, se le impide hacer consultas a la API durante un período de tiempo concreto.

Por ello se han utilizado 3 credenciales diferentes de acceso, con la finalidad de obtener el entorno de 3 semillas distintas (*@InyakiMoret*, *@zTorin* y *@KaylenDoss*). Estas cuentas se han seleccionado estratégicamente. Todas ellas tienen un número de seguidores y seguidos relativamente bajo, de esta manera la exploración no se disparaba.

De esta forma se han formado 3 grafos que hacen referencia a los círculos mas cercanos de los nodos semilla. La ejecución en cada caso ha finalizado cuando se han explorado totalmente (seguidores y seguidos) 25 nodos.

El primer grafo obtenido con la semilla @InyakiMoret ha encontrado 8794 usuarios y 12116 conexiones. El segundo con la semilla @zTorin, 9449 usuarios y 11463 conexiones. Y el tercero con @KaylenDoss 19965 usuarios y 22175 conexiones.

Una vez capturados, para poder tener todos los datos en un solo grafo, se ha utilizado la función *compose* de la librería NetworkX. Este método ha permitido unir los grafos sin réplicas de nodos ni de aristas. Así pues, se ha acabado obteniendo un solo grafo con todos los datos adquiridos anteriormente, un total de 38004 nodos y 45754 aristas.

3.1.4 Dataset obtenido

El dataset obtenido a partir del grafo originado en el apartado anterior contiene 38004 muestras con 13 atributos diferentes:

En la Tabla 1 observamos que el *dataset* consta de 6 atributos numéricos, 6 *boleanos* y una lista con diferentes *tweets*.

Para poder realizar la exploración del *dataset*, se han separado los atributos en 3 grupos distintos: los numéricos, los *boleanos* y los *tweets*.

De esta manera, a partir de los gráficos *boxplot* y *violinplot* se ha podido comprobar la distribución de las variables numéricas. Y de la misma manera se ha hecho con los datos boléanos a partir de gráficos de barras. Así es como se ha podido tener una primera visión de los datos.

3.1.5 *Data manipulation*

Una vez obtenida la base de datos, se ha tenido que hacer limpieza en esta, puesto que había información que algún usuario tenía incompleta. De esta manera, se ha acabado obteniendo una base de datos completa de 37991 registros.

También se ha necesitado que todos los atributos sean numéricos. Esto se debe a que posteriormente se utiliza la distancia euclidiana en las técnicas de ML para medir la similitud entre puntos de datos, y esta distancia sólo se puede calcular entre puntos con valores numéricos. Por esta razón, todas las variables *boleanas* descritas anteriormente, se han modificado a valores 0 (*false*) o 1 (*true*).

Y, además, los datos se han *normalizado*[4] para garantizar que las variables tengan una influencia equilibrada en el proceso de agrupamiento y que el resultado final no se vea afectado por la escala, varianza o valores atípicos de los datos.

Finalmente, se ha realizado un pre procesado en los *tweets* de los usuarios haciendo traducción al inglés del texto, quitando *emojis*, *urls* y menciones.

[4]Normalizar entre 0 y 1 las variables de un dataset permiten eliminar las diferencias de escala, mejorar la estabilidad numérica y permitir la comparación de variables con diferentes unidades de medida o magnitudes.

3.2 Caracterización del usuario

Para obtener la caracterización del usuario se ha pensado en dos fases diferentes:

1. Caracterizar al usuario a partir de sus atributos numéricos y boléanos con técnicas de *Machine Learning*.
2. Caracterizar al usuario a partir del contenido que generan en sus tweets con técnicas de *Deep Learning*.

3.2.1 Caracterización del usuario a partir de técnicas de ML

Para la primera parte, se ha planteado utilizar dos técnicas de aprendizaje no supervisado, *K-Means* [8][9] y *Agglomerative clustering* [10]. Estos tipos de aprendizajes se enfocan en encontrar patrones o estructuras ocultas en los datos sin la ayuda de etiquetas o respuestas previas. De esta manera, se pretende buscar patrones que permitan poder separar los usuarios en diferentes grupos.

3.2.1.1 *K-Means clustering*

El método de *k -Means clustering* es un método no jerárquico para agrupar objetos que particiona el set de datos en *k clusters* distintos y no solapantes, lo que significa que ninguna observación puede pertenecer a más de un grupo. El número de *clusters* o subgrupos requeridos se ha de establecer al inicio.

El algoritmo parte de $C_1, ..., C_k$ sets, la varianza *intracluster* para el *cluster* C_k es una medida $W(C_k)$ de la cantidad que difieren las observaciones dentro del mismo. Por tanto, se busca minimizar

$$\sum_{K=1}^{K} W(C_k)$$

de manera que la varianza total dentro de cada *cluster*, sumada sobre todos los k *clusters*, sea lo más pequeña posible. La forma común de establecer esta varianza, y la que se ha utilizado, es mediante la distancia euclidiana, con lo que obtenemos

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i'=1 \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

siendo C_k el número de observaciones en el k-ésimo *cluster*. De esta manera la varianza se mide como la suma de todas las distancias euclidianas al cuadrado entre pares de observaciones del *cluster k*, dividido por el número total de observaciones en ese mismo *cluster*. Combinando ambas ecuaciones anteriores obtenemos el problema de

TABLA 1: DESCRIPCIÓN DE LOS ATRIBUTOS QUE FORMAN EL DATASET

Atributo	Descripción	Tipo de dato
created_at	Fecha de creación de la cuenta. Se ha tenido en cuenta solo el año de creación	*Integer*
friends_count	Número de seguidos por la cuenta	*Integer*
followers_count	Número de seguidores de la cuenta	*Integer*
listed_count	Número de listas públicas de la cual la cuenta es miembro	*Integer*
favourites_count	Número total de tweets escritos por el usuario	*Integer*
statuses_count	Número total de tweets publicados por el usuario	*Integer*
location	Localización introducida por el usuario	*Boolean*
verified	Si la cuenta está verificada	*Boolean*
protected	Si la cuenta es protegida	*Boolean*
geo_enabled	Si la cuenta tiene activada la geolocalización	*Boolean*
default_profile	Si el usuario tiene la cuenta por defecto	*Boolean*
default_profile_image	Si el usuario tiene la imagen de la cuenta por defecto	*Boolean*
tweets	5 últimos *tweets* (si el usuario tiene)	lista de *strings*

optimización que define *k -Means clustering*.

El problema es que existen al menos k_n maneras de particionar *n* observaciones en *k clusters*, por lo que este puede ser un número muy alto si *k* y *n* no son pequeños. En este caso, el algoritmo iterativo de *k -Means* proporciona un óptimo local:

1. Asignar un *cluster* inicial de 1 a k de manera aleatoria a cada observación.
2. Iterar hasta que la asignación de cada *cluster* deje de cambiar:
   (a) Para cada uno de los k *clusters*, calcular el centroide del *cluster* (vector de medias de las variables *j* para las observaciones del *cluster* k).
   (b) Asignar cada observación al *cluster* cuyo centroide esté más próximo.

3.2.1.2 *Agglomerative clustering*

El segundo método es un algoritmo de *clustering* jerárquico en el que los elementos del conjunto de datos se agrupan en base a su distancia mutua. El algoritmo comienza considerando cada elemento del conjunto de datos como un *cluster* separado, y luego los agrupa gradualmente en *clusters* más grandes en función de la similitud de sus características. Por esta razón, este no requiere seleccionar de manera previa un determinado número de grupos k.

El algoritmo procede de manera iterativa. Inicialmente, cada observación es tratada como su propio *cluster*. Los dos *clusters* más parecidos entre sí se unen, quedando n–1 *clusters*. A continuación, y partiendo de este cambio, los dos grupos más parecidos entre sí se unen de nuevo, quedando n–2 *clusters*. Se continúa de esta forma hasta que todas las observaciones pertenezcan a un único conjunto, o se llegue a un determinado número de *clusters* definido previamente (k).

Entonces, los pasos que sigue el algoritmo son los siguientes:

1. Se comienza con la medida (distancia euclidiana) de disimilitud entre pares de observaciones $n_2 = n(n-1)/2$, que dependerá del *linkage*[5] escogido. Se trata cada observación como su propio *cluster*.
2. Para $i = n, n-1, ..., k-1$:
   (a) Se examina la similitud entre pares de *clusters*, se identifica el par más similar y se une.
   (b) Se calcula nuevamente la similitud entre pares de *clusters* entre los i–1 *clusters* restantes.

3.2.1.3 Selección del método

Para definir el número de *clusters*, se ha utilizado el método del codo. Esta técnica se utiliza en *clustering* para determinar el número óptimo de grupos a utilizar en un conjunto de datos. El método del codo implica graficar la suma de las distancias cuadráticas *intra-cluster* (suma de cuadrados dentro de los grupos o SSD) en función del número de *clusters*, y elegir el número de grupos en el punto donde la disminución de la varianza se desacelera significativamente, lo que a menudo forma una curva en forma de codo.

A continuación, se muestran los resultados obtenidos:

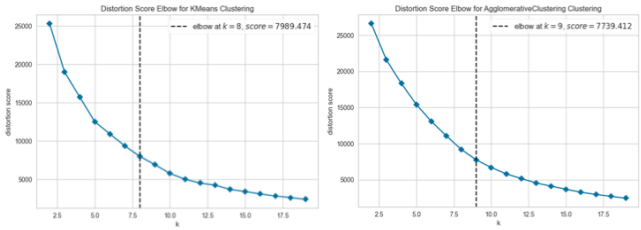

Fig. 1: Gráfico del método del codo para los algoritmos *k-Means* y *Agglomerative clustering*

En la Figura 1 se observa que con el método del codo en el rango entre $k = [2, 20]$, se ha obtenido *k = 8* con *k-Means* y *k = 9* con *Agglomerative clustering*.

Seguidamente, se han observado ventajas y desventajas de cada método [11] y se han comparado resultados con la finalidad de seleccionar uno de los dos métodos.

[5]Determina cómo se calcula la distancia entre dos grupos en función de las distancias entre sus elementos individuales.

Finalmente, se ha acabado escogiendo el método de *k-Means* por 3 motivos:

1. El algoritmo *k-Means* es un método de agrupamiento no jerárquico que divide un conjunto de datos en *k* grupos o *clusters* basados en las similitudes de los datos. Los centroides de los *clusters* se encuentran y los datos se agrupan en función de su proximidad a estos. Estos se pueden utilizar para hacer inferencias sobre nuevos datos no vistos calculando la distancia entre la nueva instancia a cada uno y asignando esta nueva observación al centroide más cercano. Por otro lado, el algoritmo de clustering jerárquico aglomerativo es un método de agrupamiento en el que los grupos se construyen a través de la unión de grupos más pequeños. En este algoritmo, los grupos se forman mediante la combinación de los objetos que son más similares entre sí. Una vez que se han formado los grupos, no se generan centroides o parámetros que se puedan utilizar para hacer inferencias sobre nuevas observaciones. En su lugar, si se desea hacer inferencias sobre nuevos datos, se debe volver a entrenar todo el algoritmo con los nuevos datos incluidos en el conjunto de datos original.

2. Además, el clustering jerárquico es un algoritmo que construye una jerarquía de grupos de objetos. Cuando se aplica a conjuntos de datos más grande, el algoritmo puede tardar mucho tiempo en completar su ejecución y consumir muchos recursos computacionales, lo que puede reducir su rendimiento. Mientras que el algoritmo *k-Means* es un algoritmo de agrupamiento mucho más eficiente en términos de tiempo de ejecución y recursos computacionales. Debido a que no construye una jerarquía de grupos, es mucho más rápido y menos costoso.

3. Finalmente, se ha calculado la puntuación de silueta (*silhouette score*) para cada método. Esta es una métrica utilizada para evaluar la calidad del *clustering*. Esta se basa en la similitud de las instancias dentro de los *clusters* y en la diferencia entre las instancias de los diferentes *clusters*. Esta métrica comprende un valor entre -1 y 1, donde un valor más alto indica que las instancias dentro de un grupo son más similares entre sí y menos similares a las instancias de los demás. Una puntuación de silueta alta, indica que los *clusters* están bien definidos y separados entre sí. Al comparar la *silhouette score* de ambos algoritmos, encontramos que el algoritmo *k-Means* con $k = 8$ y el algoritmo de clustering jerárquico aglomerativo con $k = 9$ obtienen el mismo valor de puntuación de silueta, 0.58. Es decir, el primer método consigue el mismo *score* creando un grupo menos.

Por estos motivos, hemos seleccionado *k-Means* en lugar de *Agglomerative Clustering*.

3.2.1.4 Parámetros definidos en el método

Se ha utilizado el algoritmo de *k-Means* de la librería *Scikit-learn*[6]. En este modelo, los parámetros principales que se deben definir son:

- El número de *clusters* (k): El número de grupos en los que se agruparán los datos.
- El método de inicialización: El método utilizado para seleccionar los centroides iniciales de cada *cluster*.
- El número máximo de iteraciones: El número máximo de veces que se realizará el algoritmo *k-Means* antes de que se declare la convergencia. Si los *clusters* no convergen después del número máximo de iteraciones, el algoritmo se detiene.
- La tolerancia de convergencia: Es el valor de la diferencia de la inercia entre iteraciones que indica que se ha logrado la convergencia. Si la diferencia de la inercia entre iteraciones es menor que la tolerancia, se considera que el algoritmo ha convergido y se detiene.

Los parámetros que se han utilizado han sido $k = 8$, 5000 iteraciones, 0.000001 tolerancia de convergencia y *kmeans++*[7] como método de inicialización de *clusters*.

3.2.2 Caracterización del usuario a partir de técnicas de DL

Para la segunda parte, se han utilizado técnicas de *Deep Learning* para la extracción del contenido que existe en un *tweet*. Se ha utilizado un modelo de inferencia de lenguaje natural (NLI) para la clasificación de secuencias [12].

La idea principal de los modelos NLI es tomar la secuencia que interesa etiquetar como la "premisa"y convertir cada etiqueta candidata en una "hipótesis". Si el modelo de NLI predice que la premisa implica la hipótesis, se toma la etiqueta como verdadera.

A partir de estos resultados se ha pretendido comprobar el tipo de contenido que ofrece cada grupo de usuarios encontrados en la primera caracterización. De esta manera se obtiene tanto la información del tipo de cuenta como el contenido que esta ofrece.

3.2.2.1 Modelo utilizado

El modelo utilizado ha sido *facebook/bart-large-mnli*. Este forma parte de la familia de modelos BART, que utilizan la técnica de pre-entrenamiento de *denoising sequence-to-sequence* (*Denoising AutoEncoder* [8]) para mejorar la comprensión y generación de texto. Por ello, los modelos que utilizan BART son capaces de realizar tareas de generación de texto, traducción y procesamiento de lenguaje natural.

El modelo *facebook/bart-large-mnli* en concreto, es un modelo de lenguaje natural pre-entrenado desarrollado por *Facebook AI Research* (FAIR) utilizando el marco de trabajo BART [13] y entrenado con el conjunto de datos

[6]Biblioteca de aprendizaje automático para el lenguaje de programación Python. Proporciona una amplia gama de herramientas y algoritmos para realizar tareas relacionadas con el aprendizaje supervisado y no supervisado, la selección de modelos, la evaluación de rendimiento y el procesamiento de datos.

[7]Se utiliza para inicializar los centroides de los clústers. Se elige un punto que esté lejos del primer centroide para asegurar que los centroides se distribuyan adecuadamente. Y de esta manera, el algoritmo converja a un mínimo global.

[8]La técnica de *Denoising Sequence-to-Sequence* busca entrenar modelos de lenguaje natural que sean capaces de generar texto más coherente y limpio, eliminando el ruido y las incoherencias que puedan aparecer en las secuencias de palabras generadas.

MultiNLI (MNLI)[14].

El conjunto de datos MNLI en el que se ha entrenado el modelo, es un conjunto de datos grande y diverso que contiene más de 400k ejemplos de varios géneros, lo que permite que el modelo tenga un buen desempeño en la tarea de clasificación de texto para múltiples tópicos. Este conjunto de datos se ha utilizado principalmente para la tarea de clasificación de texto, en la que se trata de clasificar un fragmento de texto en una de las tres categorías: *neutral* (neutralidad), *entailment* (implicación) o *contradiction* (contradicción).

Por ello, la salida del modelo es un vector de tres valores que corresponden a las probabilidades de la premisa de pertenecer a cada una de las tres categorías. Cada valor representa la probabilidad estimada que tiene la premisa de pertenecer a cada categoría correspondiente. La categoría con la probabilidad más alta se considera la predicción del modelo.

Por ejemplo, si el modelo da como salida un vector de $[0.1, 0.4, 0.5]$, significa que la premisa tiene un 10% de probabilidad de pertenecer a la categoría *neutral*, un 40% de probabilidad de pertenecer a la categoría *entailment* y un 50% de pertenecer a *contradiction*.

En nuestro caso, se ha definido un *threshold*[9] de 0.85 en la categoría de implicación para determinar que un género en concreto pertenece a un *tweet* determinado.

Además el modelo tiene la opción de activar el modo *multi label*. Esto le permite poder manejar múltiples etiquetas de salida en lugar de una sola etiqueta cómo se había visto en el apartado anterior. Esto significa que en lugar de clasificar una pareja de oraciones en una sola categoría de relación semántica, como *contradiction*, *neutral* o *entailment*, el modelo puede clasificar la pareja de oraciones en función de múltiples categorías de género y proporcionar una medida de confianza para cada etiqueta.

En el contexto de MNLI, donde el modelo se entrena y evalúa en varios géneros, esto es especialmente útil ya que en una misma oración pueden aparecer más de un género.

Por tanto, por cada *tweet*, cada tópico que tuviera un *entailment* con un valor más grande o igual a 0.85 se ha tomado por válido.

3.2.2.2 Tópicos utilizados

El número de tópicos que se han utilizado para realizar el análisis han sido un total de 16. A continuación se nombran y se muestran algunos ejemplos de temas que podrían ser populares dentro de cada categoría:

- *News*: últimas noticias locales o internacionales.
- *Travel*: destinos turísticos, consejos de viaje, experiencias personales, fotografías de lugares interesantes, etc.
- *Food*: recetas, recomendaciones de restaurantes, reseñas de alimentos, tendencias culinarias, etc.
- *Politics*: análisis político, discusiones sobre temas candentes, opiniones sobre políticos y políticas, etc.
- *Entertainment*: películas, programas de televisión, música, celebridades, eventos de entretenimiento, etc.
- *History*: datos históricos interesantes, efemérides, curiosidades sobre eventos pasados, etc.
- *Sports*: resultados de partidos, noticias deportivas, debates sobre equipos y jugadores, análisis de estrategias, etc.
- *Cosmetic*: consejos de belleza, reseñas de productos cosméticos, tutoriales de maquillaje, tendencias de cuidado de la piel, etc.
- *Art*: obras de arte, exposiciones, entrevistas a artistas, técnicas artísticas, etc.
- *Fashion*: tendencias de moda, consejos de estilo, reseñas de marcas y diseñadores, *looks* del día, etc.
- *Automobiles*: noticias del mundo automotriz, reseñas de vehículos, consejos de mantenimiento, debates sobre modelos y marcas, etc.
- *Health*: consejos de bienestar, noticias de salud, rutinas de ejercicio, recomendaciones de dieta, etc.
- *Technology*: últimas novedades tecnológicas, reseñas de dispositivos, debates sobre plataformas y aplicaciones, consejos de productividad, etc.
- *Science*: descubrimientos científicos, avances tecnológicos, debates sobre temas científicos, etc.
- *Gossip*: chismes de celebridades, rumores de la industria del entretenimiento, noticias del mundo del espectáculo, etc.
- *Advertising*: anuncios creativos, estrategias de marketing, casos de estudio publicitarios, análisis de campañas, etc.

[9] Valor límite que se utiliza para realizar una clasificación.

4 RESULTADOS

4.1 *Clusters* obtenidos con *k-Means*

Una vez ejecutado el algoritmo, se han obtenido los siguientes *clusters* con las siguientes distribuciones:

En la Figura 2 podemos ver que se han generado 8 grupos distintos como teníamos predefindo. Se puede observar que los *clusters* 1 y 3 representan prácticamente el 50% de los usuarios de la red. Mientras que los 6 grupos restantes representan un porcentaje más pequeño. Siendo el grupo 5 el grupo menos significativo dentro de la comunidad.

4.2 Representación gráfica de los atributos

A continuación se muestra a partir de gráficas *volinplot* y *barplot* las distribuciones de las variables numéricas y *boleanas* en cada *cluster* generado.

En la Figura 3 se osberva que se obtenien 6 gráficas *violinplot* distintas, donde se representan los atributos numéricos del dataset. Se puede ver que en la mayoría de ellas existen bastantes *outliers* es por ello que se ha generado el gráfico con más precisión donde recaen el gran porcentage de los datos.

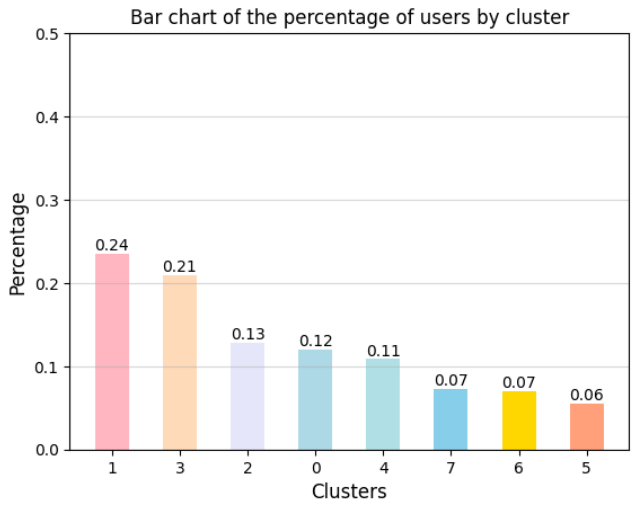

Fig. 2: Porcentaje de usuarios obtenidos por *cluster* con el algoritmo

Posteriormente, en la Figura 4 se enseñan los gráficos de barras obtenidos para representar las variables *boleanas*. Estos representan de cada grupo el porcentaje de usuarios que tienen activado el atributo (color naranja) y el porcentaje que no lo tienen (color gris).

Y, a partir de los 3 gráficos se ha realizado un análisis de las características que conforman cada grupo.

4.3 Caracterización de los grupos generados

Los gráficos obtenidos han permitido poder hacer un análisis de los grupos formados. A continuación se explica las características de cada uno:

El grupo 7 representa el 7'28% de las usuarios de la red obtenida. Esta formado por cuentas creadas entre los años 2009 y 2011. Son cuentas verificadas, sin proteger y sin el perfil ni la imagen predeterminada de Twitter. Además son las que contienen más seguidos y a su vez más seguidores. Las que son miembros de mas listas públicas y las que por mayoría generan más *tweets*. A este grupo se le ha llamado los **influenciadores** porque su descripción se ajusta a la de los usuarios que típicamente se consideran *influencers* en las redes sociales. Es decir, cuentas que tienen un gran número de seguidores y a su vez siguen a muchos usuarios, es decir, que tienen una audiencia activa y comprometida. Además, son miembros de muchas listas públicas, lo que nos hace pensar que su contenido es relevante para muchos usuarios y se comparte ampliamente. El hecho de que estas cuentas sean verificadas y estén activas desde hace varios años también sugiere que tienen cierta autoridad y credibilidad en la plataforma.

El grupo 5 es el grupo opuesto al 7. Esta formado por cuentas sin verificar y con imagen y perfil definido por defecto. Son cuentas creadas recientemente, entre los años 2017 y 2022. Además son cuentas con pocos seguidores, que escriben entre pocos y ningún *tweet*. Y que siguen a pocas cuentas. Además el gran porcentaje no son miembros prácticamente de ninguna lista pública. Este representa el 5'52%, el porcentaje menor. A este grupo se le ha llamado los **inactivos** porque su descripción indica que estas cuentas no están siendo utilizadas activamente. Es decir, son cuentas que se crearon recientemente y tienen pocos seguidores, por tanto, no han logrado atraer una audiencia significativa. Además, escriben muy pocos o ningún tweet y siguen a pocas cuentas, por tanto, no están interactuando activamente con otros usuarios de la plataforma.

El grupo 0 representa el 12'02%. Esta definido por cuentas no verificadas y sin el perfil ni la imagen por defecto. Todas ellas tienen activada la geolocalización y tienen registrada la localización. Son cuentas con menos seguidores que seguidos. Donde la gran mayoría forman miembro de alguna lista pública y son usuarios bastante activos tanto escribiendo *tweets*, como dando *likes*. A este grupo se le ha llamado **exploradores** ya que el nombre sugiere que son usuarios que están activos en la plataforma y están explorando y descubriendo contenido. La activación de la geolocalización y la localización registrada, como el hecho de que estén escribiendo *tweets* y dando *likes* indica que estos usuarios están interesados en la interacción activa con el contenido de la plataforma.

El grupo 2 representa el 12'84% y es parecido al grupo de exploradores con alguna diferencia. Son cuentas sin verificar, sin geolocalización activada y sin el perfil ni la imagen predefinida por defecto. Pero el 73% de las cuentas tienen activada la localización. Son usuarios que tienen bastantes amigos pero muy pocos seguidores en comparación. Donde la gran mayoría no forman miembro de ninguna lista pública y son usuarios que escriben algún *tweet*, y van dando *likes*, pero con menos actividad que los exploradores. A este grupo se le ha llamado los **conectados**. Este nombre se basa en que la presencia de una cantidad considerable de amigos pero pocos seguidores sugiere que estos usuarios no están buscando activamente aumentar su presencia en la plataforma, sino que se centran más en interactuar con un grupo cercano de amigos. El hecho de que la mayoría de estas cuentas tengan activada la localización muestra que están interesadas en compartir información sobre su ubicación y posiblemente en conectarse con otros usuarios en su área geográfica.

El grupo 1 representa el 23'55% de los usuarios de la red. Esta compuesto por cuentas prácticamente sin verificar dónde el 13% de ellas están protegidas. Todas elles tienen el perfil por defecto, sin localización registrada y la geolocalización desactivada. La mayoría de las cuentas son creadas entre 2011 y 2015. Son cuentas que pertenecen a alguna lista pública y que van generando de vez en cuando algún *tweet* y van haciendo algún *like*, pero sin llegar a ser muy activos. El número de seguidos y seguidores es relativamente bajo en ambos casos. Los usuarios que forman parte de este cluster se les ha llamado los **básicos**. Se caracterizan por tener perfiles y configuraciones básicas, sin demasiada actividad ni verificación, pero aún así forman parte de la red y contribuyen con su presencia.

Los grupos 3, 4 y 6 representan el 20'89%, 10'87% y el 7'01%, respectivamente. Conjuntamente casi el 40% de los usuarios de la red. Estos *clusters* son muy similares entre ellos. Tienen una distribución muy parecida en número de seguidos y seguidores, cómo de *likes* y generación de *tweets*. La diferencia está en que los usuarios del grupo 3 forman parte de un número un poco menos elevado de cuentas públicas y en general contienen un poco menos de

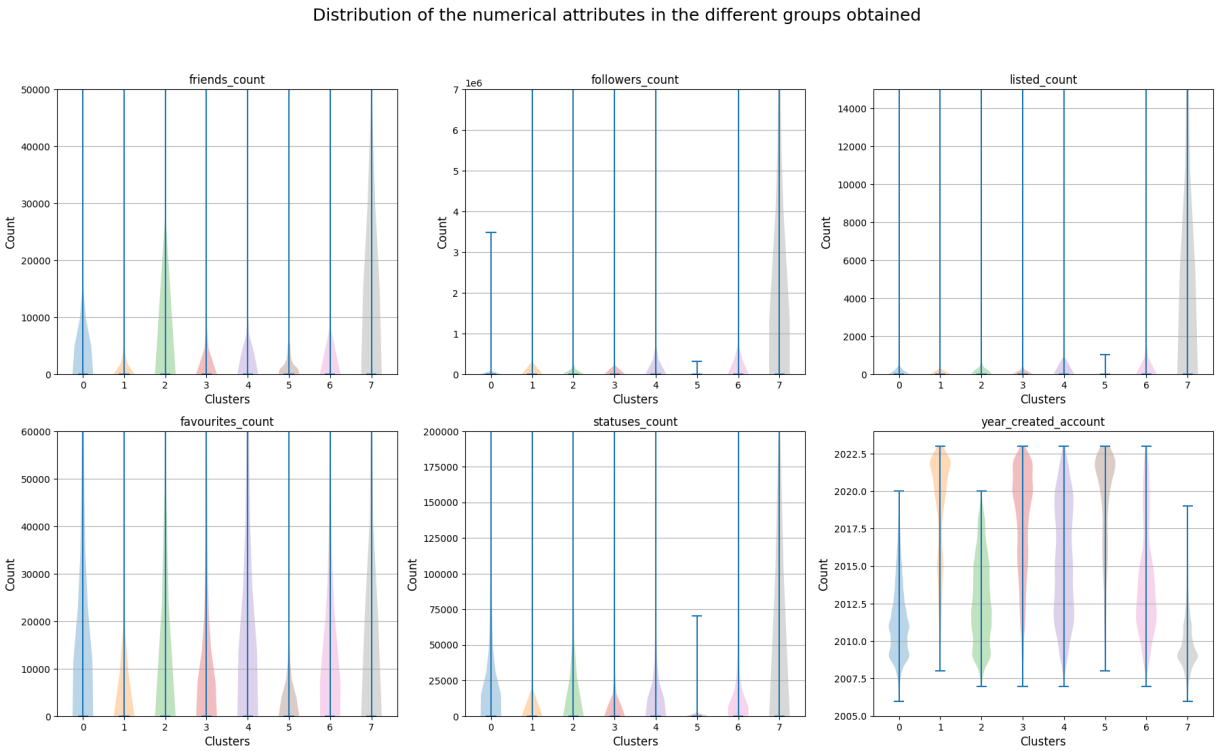

Fig. 3: Distribución de los atributos numéricos en los diferentes grupos obtenidos

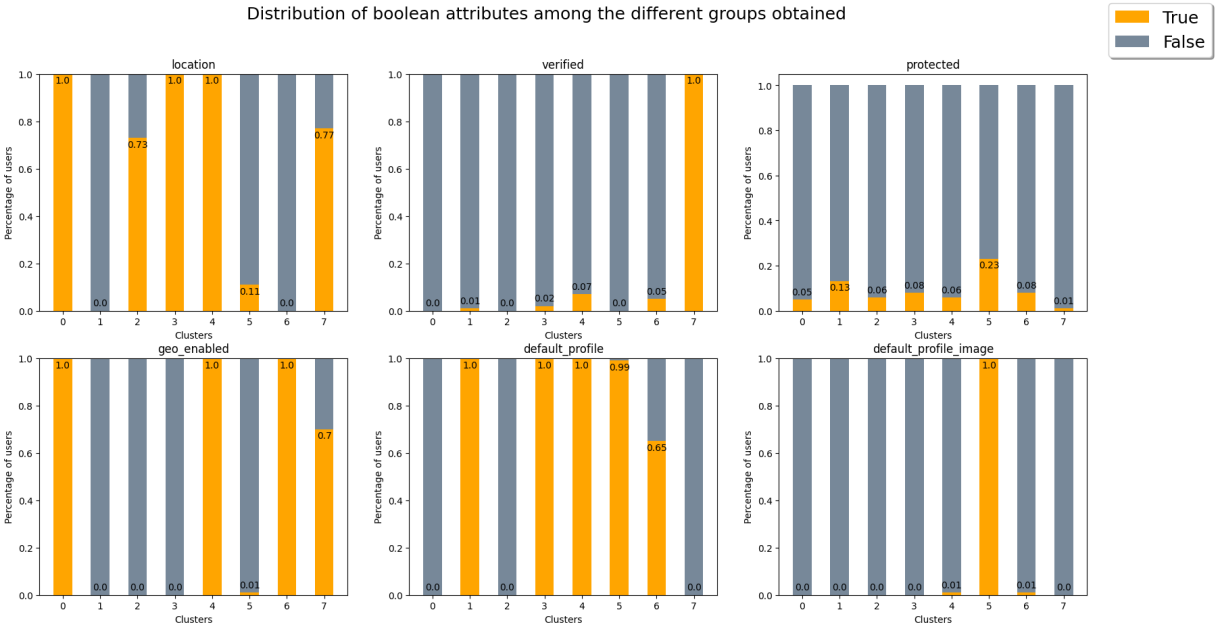

Fig. 4: Distribución de los atributos *booleanos* en los diferentes grupos obtenidos

seguidores. De la misma manera, el grupo 3 tiene habilitada la localización y desactivada la geolocalización, el grupo 6 tiene deshabilitada la localización y habilitada la geolocalización, y el grupo 4 habilitada ambas. Además todas las cuentas del grupo 3 y del 4 son cuentas con el perfil por defecto, y más del 60% del grupo 6 también. Las gran mayoría de las cuentas del grupo 3 han sido creadas entre 2015-2022, las del grupo 6 entre 2008-2017, y las del grupo 4 abarcan un rango más amplio de años de creación de 2009-2022. Por ello, se podría llamar a los usuarios de los grupos 3, 4 y 6 usuarios **estándares** puesto que se ha considerado que no difieren tan significativamente como para caracterizar cada grupo independientemente.

4.3.1 Análisis final de los grupos obtenidos

Como se ha podido observar se han identificado 6 grupos diferentes:

- El grupo 7 son los **influenciadores**: cuentas verificadas, sin protección, con muchos seguidores y actividad constante.
- El grupo 5 son los **inactivos**: cuentas sin verificar, con

pocos seguidores, sin actividad y sin miembros en listas públicas.

- El grupo 0 son los **exploradores**: cuentas no verificadas, sin imagen ni perfil predeterminado, con geolocalización activada y actividad constante, y son miembros de alguna lista pública.
- El grupo 2 son los **conectados**: cuentas sin verificar, sin geolocalización activada, pero con la mayoría de cuentas con localización registrada, con bastantes amigos y actividad bastante constante. Muchas de ellas son o miembro de pocas listas públicas o de ninguna.
- El grupo 1 son los **básicos**: cuentas sin verificar, con perfil predeterminado, sin geolocalización activada, con actividad en la red, con pocos seguidores y miembros en listas públicas.
- Los grupos 3, 4 y 6 no tienen características tan distintivas como los otros grupos. Por ello se le denominan **estándares**.

4.4 Contenido que generan los usuarios dentro de los grupos obtenidos

4.4.1 Porcentaje de usuarios con *tweets* por grupo

TABLA 2: PORCENTAJE DE USUARIOS CON *tweets* POR GRUPO

Grupo	% usuarios con *tweets*	% usuarios sin *tweets*
Influenciadores	91'87%	8'13%
Inactivos	11'2%	88'8%
Básicos	74'13%	25'87%
Exploradores	65'55%	34'45%
Conectados	80'47%	19'53%
Estándares	55'51%	44'49%

La Tabla 2 muestra el porcentaje de usuarios de cada grupo los cuales se han podido obtener sus últimos *tweets*.

Como podemos observar el porcentaje de usuarios tiene relación con la descripción realizada anteriormente por cada grupo. Vemos que el porcentaje más elevado, con un 91'87%, ha sido el grupo de los influenciadores mientras que el porcentaje más bajo ha sido el de los inactivos, como era de predecir, con un 11'2%.

A partir de estos datos se ha hecho el análisis del contenido que genera cada grupo.

4.4.2 Top 5 géneros que escriben los usuarios de cada grupo

En la Figura 5 se pueden ver los 5 géneros más escritos por los usuarios de cada grupo. Si hacemos un análisis vemos que los tópicos que se repiten en todos los grupos son *news*, *travel*, *entretainment* y *technology*. Que el género *sport* se repite en 5 y *food* solo en inactivos. Y que precisamente esta última es la única diferencia.

Por ello, se ha podido comprobar que la distribución es muy similar en todos ellos y que el tópico del cual más escriben la mayoría son de noticias.

Por tanto, podemos afirmar que no hay diferencias de contenidos en los diferentes grupos, es decir, que en todos ellos hay una variedad muy similar.

5 CONCLUSIÓN

Este proyecto ha tenido como objetivo principal identificar los diferentes grupos de usuarios que pueden existir en la red social de Twitter y caracterizarlos en función de sus propiedades y comportamientos específicos.

Para ello, se ha generado una base de datos de un subgrafo de la red utilizando la librería Tweepy, que ha permitido extraer información tanto de la cuenta como de los últimos escritos de cada usuario en la plataforma si es que tenían.

Una vez obtenidos los datos, se han utilizado técnicas de *Machine Learning*, como el algoritmo *k-Means*, y de *Deep Learning*, en concreto el modelo *facebook/bart-large-mnli*, para realizar la caracterización de los usuarios en dos niveles distintos. Por un lado, se ha descubierto la distribución de los *clusters* creados por *k-Means* junto con sus características, y por otro lado, se ha identificado el contenido que cada grupo de usuarios ofrecía en sus publicaciones.

La primera caracterización ha permitido identificar a 6 grupos principales de usuarios en la red social, todos ellos con perfiles distintos, con diferentes propósitos y motivaciones en la plataforma. Estos grupos han sido los influenciadores, los inactivos, los básicos, los exploradores, los conectados y los estándares.

En la segunda caracterización, se ha utilizado un modelo de inferencia de lenguaje natural (NLI) para clasificar las secuencias de texto y obtener información valiosa sobre el contenido que cada grupo de usuarios ofrecía en Twitter. Para ello, se ha utilizado un total de 16 tópicos para analizar el contenido de los *tweets* y se ha descubierto que los géneros más populares y que más se repiten en todos los grupos son *news*, *travel*, *entertainment* y *technology*. Se ha encontrado que el género *food* solo aparece en cuentas inactivas mientras que *sport* aparece en todas las demás. Sin embargo, se ha concluido que la distribución del contenido en los diferentes grupos de usuarios obtenidos es muy similar, ya que en todos ellos hay una variedad muy parecida de información generada, con un enfoque predominante en las noticias.

En resumen, este proyecto ha permitido identificar y caracterizar los diferentes usuarios de la red social de Twitter en 6 grupos distintos, y se ha estudiado que la distribución del contenido en cada uno de ellos es la misma.

6 SIGUIENTES PASOS

Además de los resultados obtenidos en la identificación y caracterización de los diferentes grupos de usuarios en la red social de Twitter, es importante considerar la posibilidad de adquirir otra base de datos con semillas distintas. Esta acción permitiría analizar si los resultados obtenidos son consistentes y se mantienen similares en diferentes conjuntos de usuarios.

También, sería interesante explorar el uso de técnicas de DL para generar usuarios nuevos a partir de los datos previamente clasificados con el algoritmo *k-Means*. El

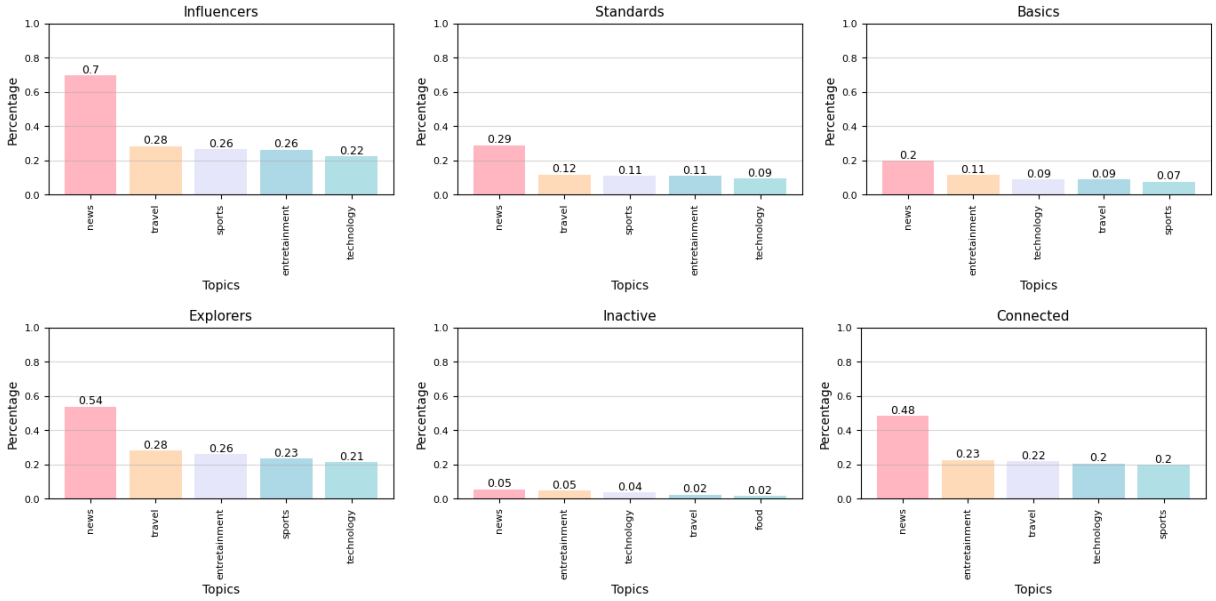

Fig. 5: Top 5 géneros más escritos por los usuarios de cada grupo

aplicar modelos como los *generative adversarial networks* (GANs)[10], permitiría crear perfiles de usuarios ficticios que se asemejarían a las características que originan los distintos grupos identificados y así se podrían utilizar en las simulaciones.

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor, Eduardo César, y a la profesora, Anna Sikora, sus orientaciones, apoyo y confianza durante todo este proyecto. También agradecer a todo mi entorno su constante respaldo. Y por último, pero no menos importante, dar las gracias a todos aquellos amigos y compañeros de clase por sus ideas y sugerencias que han enriquecido completamente el trabajo.

Referéncias

[1] GeeksforGeeks. (2022, 2 noviembre). Python Status object in Tweepy.https://www.geeksforgeeks.org/python-status-object-intweepy/

[2] F F., and F. (s. f.). Extracción y gestión básica de datos de Twitter con Python. Observatorio BI and Analytics. http://spaceanalytics.blogspot.com/2018/04/tweepy-python.html

[3] API reference index. (s. f.). Docs — Twitter Developer Platform. https://developer.twitter.com/en/docs/twitter-api/api-reference-index

[4] API — tweepy 4.12.1 documentation. (s. f.) https://docs.tweepy.org/en/latest/api.html

[5] User object. (s. f.) Docs Twitter Developer Platform. https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/user

[6] GET — statuses user timeline. (s. f.). Docs Twitter Developer Platform. https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline

[7] Tutorial — NetworkX 3.1 documentation. (s. f.) https://networkx.org/documentation/stable/tutorial.html

[8] Machine Learning con Python y Scikitlearn. (s. f.) https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.html

[9] Heras, J. M. (2020, 20 septiembre). Clustering (Agrupamiento), KMeans con ejemplos en python - IArtificial.net. IArtificial.net. https://www.iartificial.net/clustering-agrupamiento-kmeansejemplos-en-python/

[10] sklearn.cluster.AgglomerativeClustering. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

[11] When should we choose agglomerative clustering over K-means clustering? (s. f.). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/91182/when-should-wechoose-agglomerative-clustering-over-k-means-clu

[12] Blog, J. D. (2020, 29 mayo). Zero-Shot Learning in Modern NLP. Joe Davison Blog. https://joeddav.github.io/blog/2020/05/29/ZSL.html

[13] facebook/bart-large-mnli — Hugging Face. (2001, 5 abril). https://huggingface.co/facebook/bart-large-mnli

[14] multi-nli — Datasets at Hugging Face. (2022, 16 noviembre). https://huggingface.co/datasets/multi_nli

[10] Son un tipo de modelo de aprendizaje automático que se utiliza para generar datos nuevos y realistas a partir de una distribución de datos existente. Las GANs se componen de dos componentes principales: el generador y el discriminador, que se entrenan de manera adversarial.