abril, 2024

Informe de Trabajo
Minería Web

Alberto García Martín

VNiVERSiDAD D SALAMANCA

Máster en Sistemas Inteligentes
Universidad de Salamanca

Índice

1. Introducción

En este informe se documenta el trabajo realizado para la asignatura de Minería Web del máster en Sistemas Inteligentes. Este trabajo se ha centrado en la resolución de distintos problemas de minería web, centrados alrededor del dominio de la música. Para resolver estos problemas se han utilizado algoritmos pertenecientes a las distintas técnicas de la minería web, como la minería de contenido, la minería de uso y la minería de la estructura.

En el caso de la minería de contenido, el problema planteado ha sido el de la clasificación del género de una canción a partir de su letra. Este problema es análogo al un problema de clasificación de texto, como puede ser la clasificación de noticias, pero aplicado a un dominio donde la relación entre las letras de una canción y su género es no está tan clara.

Por otro lado, en el problema de la minería de uso se busca obtener recomendaciones de canciones a partir del historial de escucha de un usuario. Para resolver este problema se han utilizado, entre otras, técnicas de vecinos más cercanos, que permiten obtener recomendaciones teniendo en cuenta tanto las canciones escuchadas por el usuario en la sesión actual como las escuchadas en sesiones anteriores.

Finalmente, la minería de la estructura se aplica para identificar comunidades de artistas musicales en función de las colaboraciones que han realizado en diferentes canciones con otros artistas. Este análisis permite identificar grupos de artistas que han colaborado entre sí y que, por tanto, pueden tener un estilo musical similar. Además, se puede descubrir que artistas son los más influyentes en la red de colaboraciones.

Para abordar estos problemas, se han utilizado principalmente herramientas y bibliotecas de Python relacionadas con la minería de datos, como pandas para la manipulación y análisis de datos, spaCy para el procesamiento del lenguaje natural, scikit-learn para la implementación de algoritmos de aprendizaje automático, session-rec para la implementación de recomendadores basados en sesiones y NetworkX para el análisis de redes.

A lo largo de este trabajo, se describirán los algoritmos aplicados para cada una de las tareas mencionadas, justificando su elección y explicando el preprocesamiento necesario de los datos. Además, se presentarán y analizarán los resultados obtenidos, discutiendo su relevancia y posibles aplicaciones en el dominio de la música.

2. Minería de contenido

En esta sección se aborda el problema de la clasificación del género de una canción a partir de su letra. Para ello, se ha utilizado un conjunto de datos que contiene letras de canciones de distintos géneros musicales, como rock, pop, rap, R&B o country. El objetivo es entrenar un clasificador capaz de predecir el género de una canción a partir de su letra.

2.1. Descripción de los datos

El conjunto de datos utilizado para este problema es *Genius Song Lyrics* [2] subido a la plataforma Kaggle. Este conjunto de datos contiene información extraída de la página web Genius, sobre más de 5 millones de canciones hasta 2022. Además de incluir el nombre de la canción, el artista y el género musical, también contiene la letra de la canción. Las letras de las canciones están en su formato original, sin ningún tipo de preprocesamiento, de hecho incluye información adicional sobre la estructura de la canción entre corchetes que habrá que eliminar.

Para facilitar el procesamiento de lenguaje natural, se incluye en una columna el lenguaje de la letra, obtenido a partir de los modelos de detección de idioma CLD3 [7] y FastText [4], en caso de discrepancia entre los dos modelos o falta de confianza se marca como desconocido. El conjunto de datos contiene predominantemente canciones en inglés, aunque también incluye canciones en otros idiomas, principalmente occidentales. La distribución de las canciones por idioma se muestra en la tabla 1.

Idioma	**%**
Inglés	65.71 %
Español	5.36 %
Francés	3.69 %
Portugués	3.27 %
Ruso	3.23 %
Otros	18.73 %

Tabla 1: Distribución de canciones por idioma en el conjunto de datos.

En cuanto al género musical, el conjunto de datos incluye los más populares, estos son: pop, rap, rock, R&B, country y misc, siendo este último una categoría que engloba otros géneros menos populares, o incluso obras que no son estrictamente canciones. En total, el conjunto de datos contiene 5.1 millones de canciones, cuya distribución por género se muestra en la tabla 2. Como se puede observar, la representación de los géneros no es equitativa, siendo el pop y el rap los más representados con mucha diferencia respecto a los demás. Este es el caso incluso si se consideran solo las canciones en inglés.

Género	**%**	**% solo ENG**
Pop	41.64 %	41.3 %
Rap	33.59 %	28.59 %
Rock	15.44 %	18.77 %
R&B	3.83 %	4.60 %
Misc	3.53 %	4.18 %
Country	1.95 %	2.57 %

Tabla 2: Distribución de canciones por género en el conjunto de datos.

2.2. Preprocesamiento de los datos

El primer paso en el preprocesamiento de los datos es la eliminación de registros innecesarios. Para facilitar la tarea de procesamiento de lenguaje natural, se considerarán solo las canciones en inglés, ya que es el idioma predominante en el conjunto de datos. Por lo tanto se eliminarán las canciones en otros idiomas, así como aquellas canciones para las que no se ha podido detectar el idioma. Además, se eliminarán las canciones de género *misc*, ya que puede que no aporten información relevante para el problema de clasificación y solo introduzcan ruido.

Tras eliminar los registros innecesarios, se tienen 3.2 millones de canciones, de las cuales el género menos representado es el country con un total algo superior a 80000 canciones. Para balancear el conjunto de datos, se ha decidido submuestrear las canciones de los géneros más representados, de forma que todos los géneros tengan el mismo número de canciones. De esta forma, se obtiene un conjunto de datos equilibrado con 400000 canciones, 80000 por género.

A continuación, se realiza el preprocesamiento de las letras de las canciones. En primer lugar, se eliminan las palabras entre corchetes, ya que contienen información adicional sobre la estructura de la canción que no forma parte de la letra, como pueden ser indicaciones de versos o estribillos. Para lograr esto simplemente se usa una expresión regular que elimine las líneas que empiecen y terminen con corchetes.

Para reducir la dimensionalidad del corpus de palabras, se realiza un procesamiento de lenguaje natural que incluye la tokenización, eliminación de *stopwords*, lematización, eliminación de palabras con signos no alfabéticos y normalización a minúsculas. Este proceso se realiza principalmente con la biblioteca spaCy [3], que permite realizar todas estas tareas de forma sencilla y eficiente. Para ello se ha creado un *pipeline* de procesamiento de texto que incluye los componentes necesarios para realizar estas tareas, optimizados para el idioma inglés. Los componentes de spaCy usados en este caso son el tokenizador, el *tagger* y el lematizador, provenientes de la *pipeline* `en_core_web_sm`.

El tokenizador se encarga de dividir el texto en unidades mínimas de procesamiento, llamadas tókenes, que dependiendo del tokenizador pueden ser varias palabras, una palabra, sílabas, etc. En este caso, el tokenizador de spaCy divide el texto en segmentos siguiendo diversas reglas y teniendo en cuenta las reglas lingüísticas del idioma del texto que se está procesando. Este tokenizador es no destructivo, por lo que inicialmente preserva los espacios y los saltos de línea del texto original. En la figura 1 se muestra un ejemplo ilustrativo de como funciona este tokenizador.

El *tagger* es un componente de spaCy necesario para que el lematizador funcione correctamente, ya que asigna a cada token una etiqueta gramatical que indica su categoría gramatical, como verbo, sustantivo, adjetivo, etc. Esto lo hace a partir de un modelo estadístico que predice la etiqueta de cada token en función de su contexto y de las reglas del lenguaje en cuestión. Por ejemplo, un token que venga después de un artículo es probable que sea un sustantivo y se indicará como tal. En la figura 2 se muestra un ejemplo de cómo funciona este etiquetador.

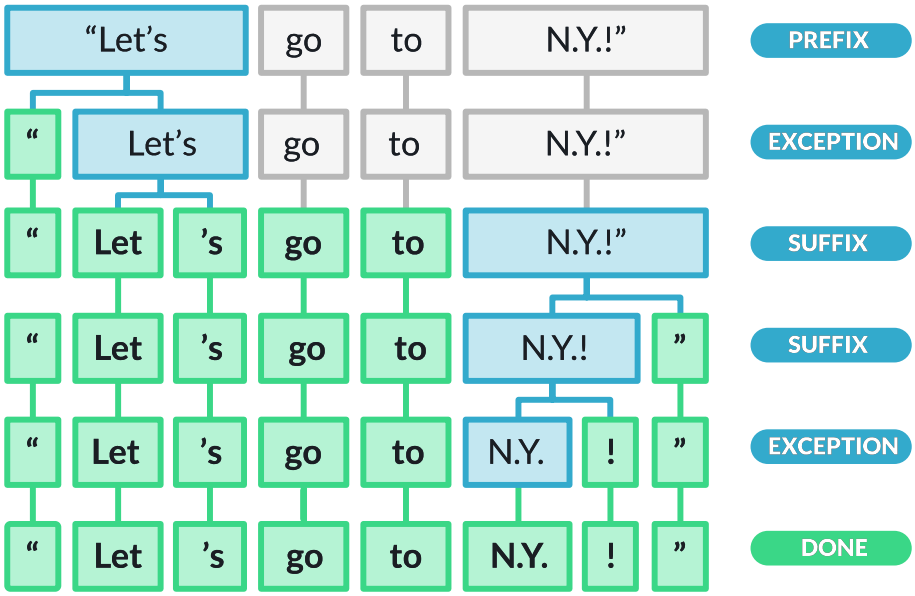

Figura 1: Ejemplo de tokenización de un texto con spaCy. (Fuente: spacy.io)

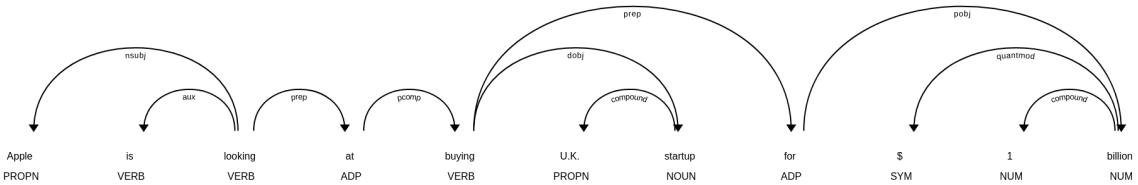

Figura 2: Ejemplo de etiquetado gramatical de un texto con spaCy. (Fuente: spacy.io)

Por último, el lematizador se encarga de reducir las palabras a su forma base, o lema, eliminando las derivaciones que pueda tener una palabra, por ejemplo eliminando prefijos o pasando a singular los sustantivos en plural. Esto es muy útil para reducir la dimensionalidad del corpus de palabras, ya que se reduce el número de palabras distintas que aparecen en el texto. En el caso de spaCy la lematización la realiza de dos formas: primero, mediante una tabla de búsqueda de lemas, que contiene las formas más comunes de las palabras y sus lemas correspondientes, y segundo, mediante reglas lingüísticas que permiten lematizar palabras que no estén en la tabla de búsqueda. Ambas técnicas también dependen del idioma del texto que se está procesando.

Una vez se ha procesado el corpus original, se obtiene una estructura de datos de la que se extraen los tókenes lematizados, que se pasan a minúsculas y se comparan con una lista de *stopwords* en inglés para eliminar aquellas palabras que no aportan información relevante para la clasificación, como artículos o preposiciones. Además, se los tókenes no alfabéticos, como números o símbolos de puntuación, ya que no suelen aportar información relevante. Finalmente, se obtiene un corpus de letras de canciones preprocesadas que se puede utilizarse en la siguiente etapa de clasificación, para la que se dividirán las canciones en un conjunto de entrenamiento y un conjunto de test, siendo el primero el 80 % de las canciones y el segundo el 20 %.

2.3. Algoritmos de minería de contenido

Para obtener una representación abstracta de los tókenes obtenidos en el paso anterior se han utilizado las siguientes técnicas de vectorización de texto, estas técnicas están ordenadas de menor a mayor complejidad de procesamiento y capacidad de representación:

- **Bag-of-Words**: Esta técnica consiste en representar cada documento (en este caso cada canción) como un vector de frecuencias de las palabras que aparecen en él. Para ello, se crea un vocabulario con todas las palabras distintas que aparecen en el corpus y se asigna un índice a cada palabra. A continuación, se cuenta el número de veces que aparece cada palabra en cada documento y se obtiene un vector de frecuencias absolutas. En este caso, se ha utilizado la implementación de scikit-learn [8] `CountVectorizer` para realizar esta vectorización.
- **TF-IDF**: Es una extensión de la técnica de Bag-of-Words que tiene en cuenta la importancia de las palabras en el documento y en el corpus. Para ello, se calcula el producto de dos valores: el término de frecuencia (TF), que mide la frecuencia de una palabra en un documento, y la frecuencia de documento inversa (IDF), que mide la importancia de una palabra en el corpus. De esta forma, se obtiene un vector de características que tiene en cuenta tanto la frecuencia de las palabras en el documento como su importancia en el corpus. En este caso, se ha utilizado la implementación de scikit-learn `TfidfVectorizer` para realizar esta vectorización.
- **Word2Vec**: En este caso se representa cada palabra como un vector denso de números reales, de forma que palabras similares tengan vectores similares. Estos vectores se obtienen a partir de un modelo de aprendizaje profundo que aprende a predecir una palabra a partir de su contexto. En este caso, se ha utilizado la implementación de la biblioteca Gensim [9] para entrenar un modelo Word2Vec con las letras de las canciones preprocesadas. Para ello, se ha utilizado un tamaño de ventana de 5 palabras, una dimensión de 300 para el tamaño de los vectores y un número mínimo de una palabra para considerarla en el modelo.
- **SBERT**: En este caso se aplica un modelo de aprendizaje profundo pre-entrenado sobre una gran cantidad de texto para obtener un vector. Estos modelos son capaces de capturar la semántica de las palabras y su contexto, lo que permite obtener representaciones de texto de alta calidad. En este caso, se ha utilizado la biblioteca Sentence-Transformer [10] para obtener representaciones de las letras de las canciones a partir del modelo `all-MiniLM-L6-v2`. En este caso se ha usado el texto original sin preprocesar, ya que el modelo aplica su propia tokenización.

Para la clasificación de las canciones se han utilizado los siguientes algoritmos de aprendizaje automático, en este caso no están ordenados por ningún criterio en particular:

- **Regresión logística**: Es un algoritmo de clasificación lineal que se utiliza para predecir la probabilidad de que una instancia pertenezca a una clase. En este caso, se ha utilizado la implementación de scikit-learn `LogisticRegression` con un número máximo de iteraciones de 1000 y el resto de parámetros por defecto. Este método se ha utilizado con todas las técnicas de vectorización de texto mencionadas anteriormente.
- **Naive Bayes**: Un algoritmo de clasificación probabilístico que se basa en el teorema de Bayes. En este caso, se ha utilizado la implementación de scikit-learn `MultinomialNB` con todos los parámetros por defecto. Este método solo se ha utilizado con las técnicas de vectorización de Bag-of-Words y TF-IDF, ya que no es compatible con las representaciones de Word2Vec y SBERT, al no estar basadas en probabilidades.

2.4. Resultados

Para evaluar los algoritmos de clasificación se han utilizado las métricas de precisión, exhaustividad y F1, que permiten evaluar el rendimiento de los clasificadores en función de los verdaderos positivos, falsos positivos y falsos negativos.

Los resultados obtenidos con la técnica de Bag-of-Words y los algoritmos de regresión logística y Naive Bayes se muestran en las tablas 3 y 4, respectivamente. Ambos algoritmos se comportan prácticamente igual, con un F1 medio de 0.55 y 0.56, respectivamente. La clase con peor rendimiento, con diferencia respecto a las demás, es la de pop, que tiene una precisión y exhaustividad muy bajas. Esto puede deberse a que el género pop es menos lírico comparado con otros géneros como el rap, lo que dificulta la clasificación en este caso.

	Precision	Recall	F1-Score	Support
Country	0.62	0.66	0.64	16000
Pop	0.35	0.21	0.27	16000
Rap	0.76	0.75	0.75	16000
R&B	0.53	0.59	0.56	16000
Rock	0.49	0.60	0.54	15999

Tabla 3: BoW con Naive Bayes

	Precision	Recall	F1-Score	Support
Country	0.62	0.68	0.65	16000
Pop	0.34	0.31	0.32	16000
Rap	0.78	0.72	0.75	16000
R&B	0.56	0.54	0.55	16000
Rock	0.49	0.54	0.51	15999

Tabla 4: BoW con regresión logística

Los resultados obtenidos con la técnica de TF-IDF y los algoritmos de regresión logística y Naive Bayes se muestran en las tablas 5 y 6, respectivamente. En este caso, los resultados son prácticamente iguales a los obtenidos con la técnica de Bag-of-Words, con un F1 medio de 0.54 y 0.59, respectivamente. La clase con peor rendimiento sigue siendo la de pop. Los resultados obtenidos con TF-IDF y regresión logística son ligeramente mejores que los obtenidos con Bag-of-Words.

	Precision	Recall	F1-Score	Support
Country	0.59	0.72	0.65	16000
Pop	0.39	0.17	0.24	16000
Rap	0.66	0.84	0.74	16000
R&B	0.53	0.55	0.54	16000
Rock	0.52	0.55	0.54	15999

Tabla 5: TF-IDF con Naive Bayes

	Precision	Recall	F1-Score	Support
Country	0.64	0.70	0.67	16000
Pop	0.38	0.31	0.34	16000
Rap	0.81	0.79	0.80	16000
R&B	0.58	0.59	0.59	16000
Rock	0.53	0.58	0.56	15999

Tabla 6: TF-IDF con regresión logística

Los resultados obtenidos con la técnica de Word2Vec y el algoritmo de regresión logística se muestran en la tabla 7. En este caso, los resultados vuelven a ser muy similares a los obtenidos con las técnicas anteriores, con un F1 medio de 0.56. La clase con peor rendimiento sigue siendo la de pop.

	Precision	Recall	F1-Score	Support
Country	0.61	0.69	0.65	16000
Pop	0.36	0.26	0.30	16000
Rap	0.75	0.79	0.77	16000
R&B	0.55	0.57	0.56	16000
Rock	0.52	0.57	0.55	15999

Tabla 7: Word2Vec con regresión logística

Por último, los resultados obtenidos con la técnica de SBERT y el algoritmo de regresión logística se muestran en la tabla 8. Una vez más, los resultados son muy similares a los obtenidos con las técnicas anteriores, con un F1 medio de 0.56.

Como se ha observado en las métricas obtenidas, los resultados de los clasificadores son similares independientemente de la técnica de vectorización de texto o algoritmo de clasificación utilizados. En general, el género con mejores resultados es el rap,

	Precision	Recall	F1-Score	Support
Country	0.59	0.68	0.63	16000
Pop	0.37	0.26	0.30	16000
Rap	0.75	0.78	0.77	16000
R&B	0.55	0.58	0.56	16000
Rock	0.52	0.55	0.54	16000

Tabla 8: SBERT con regresión logística

mientras que el género con peores resultados es el pop. Esto puede deberse a que el género pop es menos lírico y más variado en cuanto a temáticas. Por otro lado, el género rap tiene un estilo más marcado y unas temáticas más concretas, lo que facilita la clasificación.

También se ha comprobado que los resultados al utilizar el texto original sin preprocesar son muy similares a los obtenidos con el texto preprocesado. Esto puede deberse a que el conjunto de datos es de un tamaño lo suficientemente grande como para que el ruido introducido por las palabras no relevantes sea despreciable.

3. Minería de uso

En este apartado se pretende resolver un problema de recomendación de canciones a partir del historial de escucha de un usuario. Para ello, se ha utilizado un conjunto de datos extraído de la plataforma Last.FM que contiene información sobre las canciones escuchadas por un conjunto de mil usuarios. El objetivo es predecir qué canciones escuchará un usuario en función de las canciones que ha escuchado en la sesión actual y en sesiones anteriores.

3.1. Descripción de los datos

El conjunto de datos empleado para esta tarea es el *Last.FM 1K User Dataset* [1] este es un conjunto de datos muy popular para el dominio de la música en el ámbito de los sistemas de recomendación. Sus filas contienen información sobre el usuario que ha escuchado una canción, la hora exacta en la que la ha escuchado, el nombre de la canción y del artista.

En total hay más de 19 millones de registros en el conjunto de datos, que corresponden a 992 usuarios. Aparte del identificador de usuario y de la canción, el campo más importante es el que contiene la marca de tiempo, ya que de este se pueden extraer las sesiones de escucha de cada usuario. Esto se llevará a cabo en la etapa de preprocesamiento de los datos.

3.2. Preprocesamiento de los datos

El primer paso en el preprocesamiento de los datos es la creación de las sesiones de escucha de cada usuario. Para ello, se agrupan los registros por usuario y se ordenan por marca de tiempo. A continuación, se dividen los registros en sesiones de escucha, considerando que una sesión termina cuando hay un tiempo de inactividad superior a 30 minutos entre dos canciones. De esta forma, se obtiene un conjunto de sesiones de escucha para cada usuario, donde cada sesión contiene una lista de canciones escuchadas en orden cronológico. A cada sesión de escucha se le asigna un identificador único que servirá para la posterior evaluación de los algoritmos de recomendación.

Una vez se han creado las sesiones de escucha, se procede a la limpieza del conjunto de datos. En primer lugar, se eliminan las canciones que han sido escuchadas por menos de 5 usuarios, ya que no aportan suficiente información para realizar una recomendación personalizada. Posteriormente se eliminan los usuarios que tienen menos de 2 sesiones de escucha, ya que no tienen el número mínimo de sesiones para formar los conjuntos de entrenamiento y prueba. Después se eliminan aquellas sesiones de escucha que contienen menos de 3 canciones, ya que no aportan suficiente información para realizar una recomendación. Por último, se limita la longitud de las sesiones a 20 canciones, ya que se considera que es un número razonable de canciones para realizar una recomendación, y de esta forma se reduce la complejidad del problema.

Para dividir el conjunto de datos en un conjunto de entrenamiento y un conjunto de test, se selecciona la última sesión de escucha de cada usuario para el conjunto de test y el resto de sesiones para el conjunto de entrenamiento. De esta forma, se simula un escenario real en el que se quiere recomendar canciones al usuario en función de las sesiones de escucha anteriores.

3.3. Algoritmos de minería de uso

Para construir un sistema de recomendaciones basado en sesiones, se ha utilizado la herramienta de Python session-rec [6], esta herramienta conforma un marco de trabajo integral para la implementación y evaluación de algoritmos de recomendación basados en sesiones. En este caso, se han utilizado dos algoritmos de recomendación *lazy*, basados en vecinos más cercanos, y otro más complejo que combina la factorización de cadenas de Markov con la factorización clásica de matrices:

- **Session-based kNN (SKNN)**: Es una implementación de vecinos más cercanos para recomendación basada en sesiones. En vez de tener en cuenta solo el último evento en la sesión actual, este algoritmo compara la sesión actual con las sesiones anteriores en el conjunto de entrenamiento para obtener una recomendación. En este caso, se ha utilizado la implementación de session-rec `SessionKNN` con un número `k` de vecinos de 100 y empleando la similitud coseno como medida de similitud.

- **Vector Multiplication Session-Based kNN (V-SKNN)**: Esta es una extensión del algoritmo SKNN que da más peso a las canciones que se han escuchado más recientemente en la sesión actual a la hora de calcular la medida de similitud. Para ello los ítems de la sesión se codifican en un vector donde el ítem más reciente tiene un valor más alto y el valor del resto decae linealmente. En este caso, se ha utilizado la implementación de session-rec `VMContextKNN` con un número `k` de vecinos de 50.
- **Session-based Matrix Factorization (SMF)**: Es un algoritmo que combina cadenas de Markov factorizadas con la factorización de matrices clásica para obtener una recomendación [5]. En este caso, se ha utilizado la implementación de session-rec `SessionMF` con los parámetros por defecto.

3.4. Resultados

Para evaluar los algoritmos de recomendación se han utilizado las métricas exhaustividad (*Recall*), MAP (*Mean Average Precision*) que mide la calidad de las recomendaciones, MRR (*Mean Reciprocal Rank*) que mide la calidad de las recomendaciones teniendo en cuenta el orden de las mismas, NDGC (*Normalized Discounted Cumulative Gain*) que mide la calidad de las recomendaciones teniendo en cuenta el orden y la relevancia de las mismas, y HitRate que mide la proporción de recomendaciones que son relevantes.

En la tabla 9 se muestran las métricas obtenidas con el algoritmo SKNN, se puede observar que las métricas obtenidas son bastante decentes, teniendo en cuenta la amplitud del conjunto de datos y la complejidad de la tarea de recomendación. El valor de NDCG llega casi al 25 % en el caso de las 20 primeras recomendaciones, lo que indica que el algoritmo es capaz de recomendar canciones relevantes en las primeras posiciones.

Metric	@3	@5	@10	@15	@20
Recall	0.1016	0.1653	0.2993	0.3647	0.3994
MAP	0.0235	0.0365	0.0509	0.0500	0.0424
NDCG	0.1457	0.1689	0.2144	0.2357	0.2466
HitRate	0.1628	0.2537	0.4114	0.4814	0.5150
MRR	0.0983	0.1189	0.1398	0.1455	0.1474

Tabla 9: Métricas obtenidas con el algoritmo SKNN

En la tabla 10 se muestran las métricas obtenidas con el algoritmo V-SKNN, se puede observar que las métricas obtenidas son ligeramente mejores que las obtenidas con el algoritmo SKNN, en todos los apartados. Por lo que se puede determinar que este algoritmo es más efectivo a la hora de recomendar canciones en este conjunto de datos, alcanza un NDCG del 27.5 % en las 20 primeras recomendaciones.

La tabla 11 muestra las métricas obtenidas con el algoritmo SMF, se puede observar que las métricas obtenidas son mucho peores que las obtenidas con los

Metric	@3	@5	@10	@15	@20
Recall	0.1283	0.2116	0.3483	0.3953	0.4147
MAP	0.0281	0.0447	0.0614	0.0556	0.0452
NDCG	0.1753	0.2076	0.2535	0.2685	0.2748
HitRate	0.2280	0.3472	0.4698	0.5108	0.5291
MRR	0.1301	0.1573	0.1742	0.1774	0.1784

Tabla 10: Métricas obtenidas con el algoritmo V-SKNN

algoritmos SKNN y V-SKNN, en todos los apartados. Este algoritmo es el único que requiere de un modelo de aprendizaje automático para realizar la recomendación, pero es evidente que no ha sido capaz de aprender correctamente la estructura de los datos. Por lo que se puede determinar que este algoritmo no es efectivo a la hora de recomendar canciones en este conjunto de datos.

Metric	@3	@5	@10	@15	@20
Recall	0.0236	0.0307	0.0451	0.0518	0.0584
MAP	0.0068	0.0075	0.0074	0.0063	0.0053
NDCG	0.0354	0.0354	0.0386	0.0406	0.0427
HitRate	0.0404	0.0493	0.0609	0.0683	0.0765
MRR	0.0290	0.0310	0.0325	0.0330	0.0335

Tabla 11: Métricas obtenidas con el algoritmo SMF

Las métricas obtenidas demuestran que los algoritmos de recomendación basados en vecinos más cercanos, siendo a priori los más simples, son bastante capaces a la hora de recomendar canciones en este conjunto de datos.

4. Minería de la estructura

En esta última sección se pretenden encontrar comunidades de artistas musicales según las colaboraciones que han realizado en sus canciones con otros artistas. Esto puede ser útil para descubrir nuevas posibles colaboraciones entre artistas o para encontrar artistas que tengan un estilo musical similar.

4.1. Descripción de los datos

El conjunto de datos es el mismo utilizado en la sección de minería de contenidos, el *Genius Song Lyrics* [2]. En este caso solamente nos interesa la información sobre el artista, la canción, y las colaboraciones que puedan aparecer bajo la columna *featured*.

4.2. Preprocesamiento de los datos

En este caso para la limpieza de registros se eliminarán todas aquellas canciones que no tengan ninguna colaboración, ya que no aportan información relevante para el problema de minería de la estructura. Esto reduce el total de canciones a 1.1 millones.

A partir de estos registros se extraen los nombres de los artistas sin duplicados, para crear la información que representa a los nodos de la red de colaboraciones. Junto al nombre de los artistas se agrupan y contabilizan los géneros de las canciones que han realizado, con el objetivo de extraer el género predominante de cada artista y guardarlo como atributo del nodo. Además a cada artista se le asigna un identificador único.

Para extraer los arcos de la red se construye una lista de adyacencia donde cada artista está conectado con los artistas con los que ha colaborado a lo largo de su discografía.

Con el objetivo de reducir el tamaño de la red y hacerla más manejable, se eliminan aquellos artistas que tienen igual o menos de 10 relaciones de colaboración con otros artistas. De estos, se hace un submuestreo aleatorio de 10000 artistas. Esta lista de artistas y la lista de adyacencia anterior se utilizan para construir una lista de arcos que representan las colaboraciones entre artistas. Esta lista de arcos solo contendrá artistas (ya sean colaboradores o autores) que se encuentren en la lista filtrada de artistas.

De esta forma se consigue una lista con 911 nodos (artistas) y 592 arcos (colaboraciones), algunos de ellos con un peso superior a 1, ya que algunos artistas han colaborado en más de una canción. A partir de esta lista se construye un grafo no dirigido que representa la red de colaboraciones entre artistas.

4.3. Algoritmos de minería de la estructura

En la figura 3 se muestra una representación gráfica de la red de colaboraciones entre artistas obtenida mediante el programa Gephi. En esta representación, los nodos son los artistas y los arcos son las colaboraciones entre ellos. Los nodos están coloreados en función del género predominante del artista, que se ha obtenido a partir de las canciones que ha realizado. El tamaño de los nodos es proporcional al número de colaboraciones que ha realizado el artista, obtenido a partir del cálculo del *pagerank* de los nodos en la red. El grosor de los arcos es proporcional al número de colaboraciones entre los artistas (el peso del arco).

Con el objetivo de simplificar el análisis de la red de colaboraciones, se ha filtrado la red para quedarse únicamente con el subgrafo más grande de la red, este es el que tiene un mayor número de componentes conectados. Este subgrafo se puede ver en la figura 4, en esta representación se han eliminado los nodos que no pertenecen al subgrafo más grande, además en este caso se ha empleado la biblioteca NetworkX para representarla.

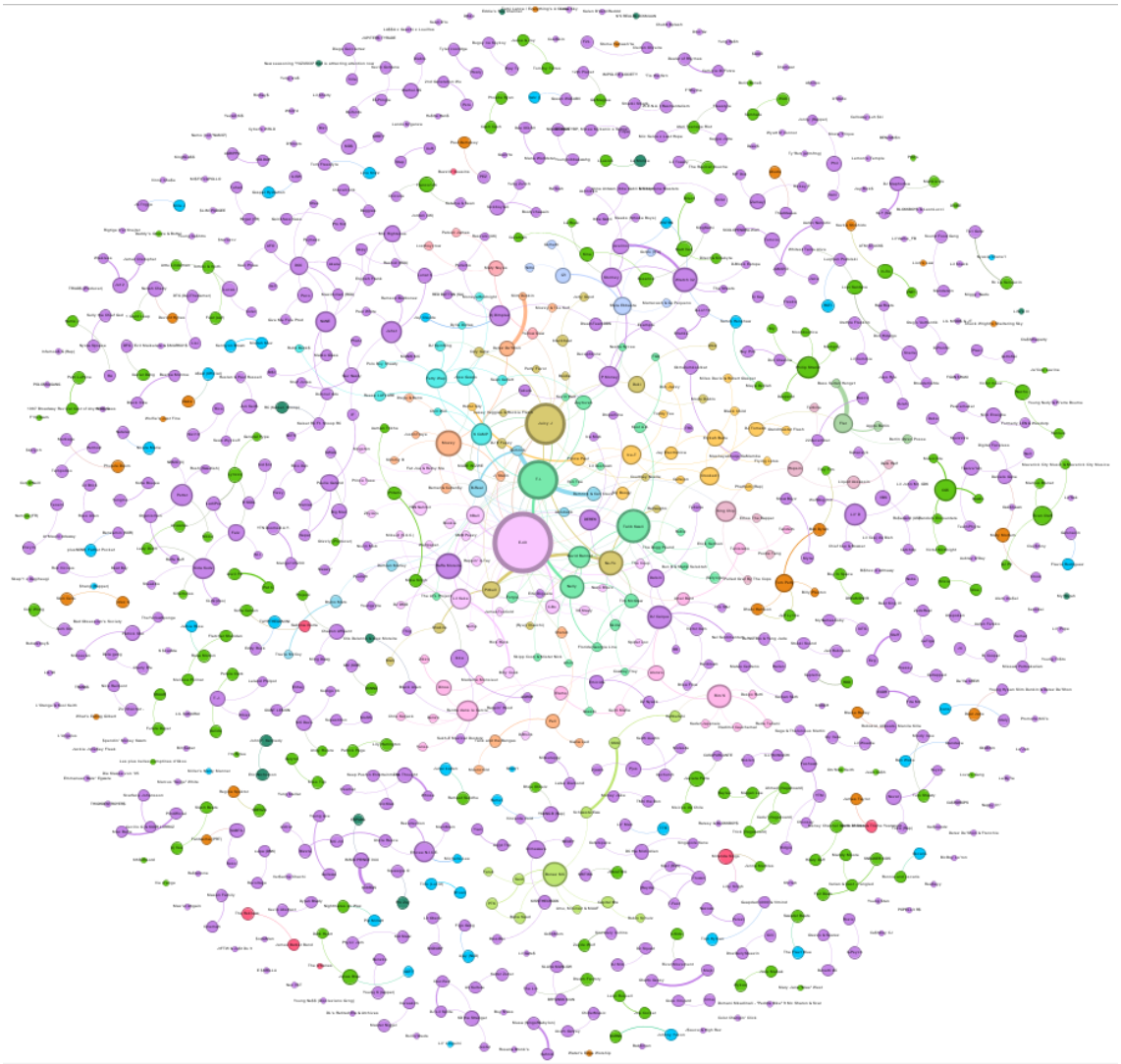

Figura 3: Red de colaboraciones entre artistas

Los algoritmos de minería de la estructura que se han utilizado para analizar esta red de colaboraciones entre artistas son los siguientes:

- **Clique Percolation Method (CPM)**: Es un algoritmo de detección de comunidades que se basa en la búsqueda de cliques en la red y en la unión de cliques que comparten nodos en común. En este caso, se ha utilizado la implementación de NetworkX `find_cliques` para encontrar los cliques en la red y `k_clique_communities` para detectar las comunidades, con un tamaño mínimo de 3 nodos por comunidad.

- **Louvain Method**: Es un algoritmo de detección de comunidades que se basa en la maximización de la modularidad de la red. En este caso, se ha utilizado la implementación de NetworkX `louvain_communities` para detectar las comunidades.

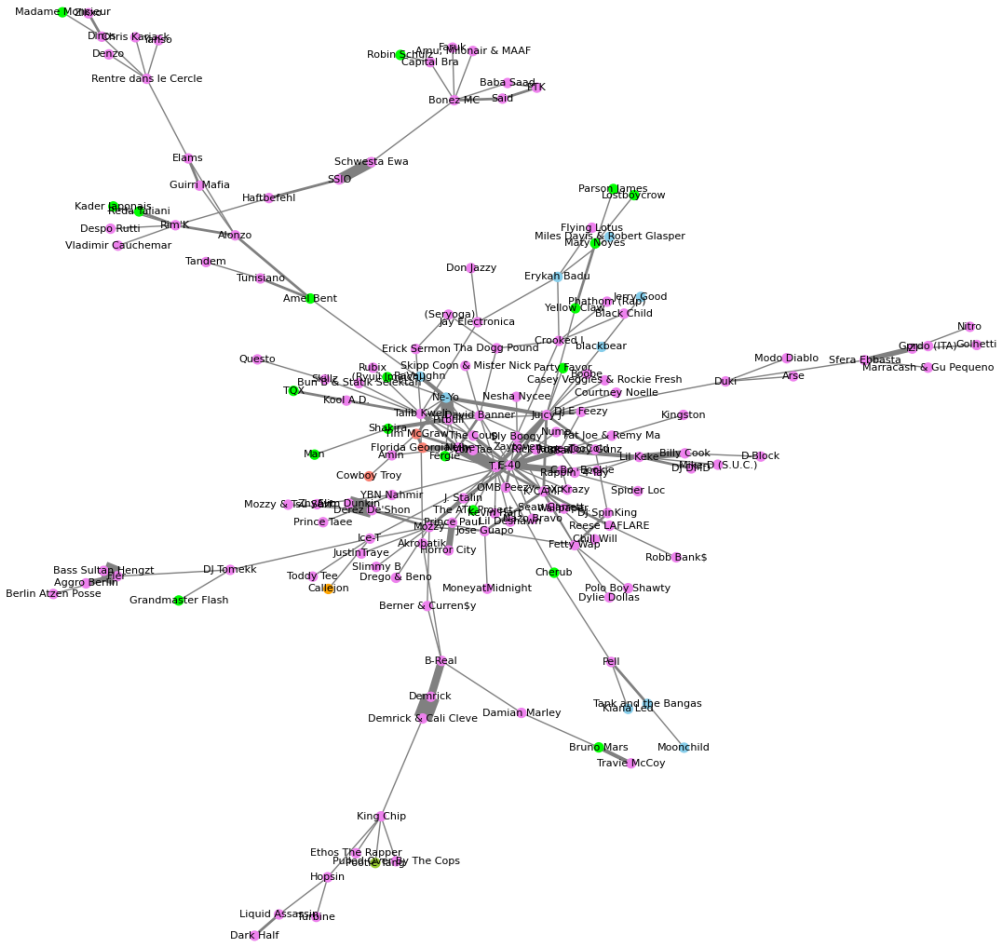

Figura 4: Subgrafo más grande de la red de colaboraciones entre artistas

4.4. Resultados

Al aplicar el algoritmo CPM a la red de colaboraciones entre artistas se han obtenido 7 comunidades de cliques, que se muestran en la figura 5. En esta representación, los nodos son los artistas y los arcos son las colaboraciones entre ellos. Los nodos están coloreados en función de la comunidad a la que pertenecen. Como se puede observar los cliques obtenidos son bastante pequeños. Esto puede deberse a que la red de colaboraciones entre artistas es bastante dispersa.

Al aplicar el algoritmo Louvain a la red de colaboraciones entre artistas se han obtenido 14 comunidades, que se muestran en la figura 6. Los nodos están coloreados en función de la comunidad a la que pertenecen. Se puede ver que en este caso las comunidades son más grandes y más homogéneas que en el caso anterior. Las comunidades tienen una clara relación con las colaboraciones.

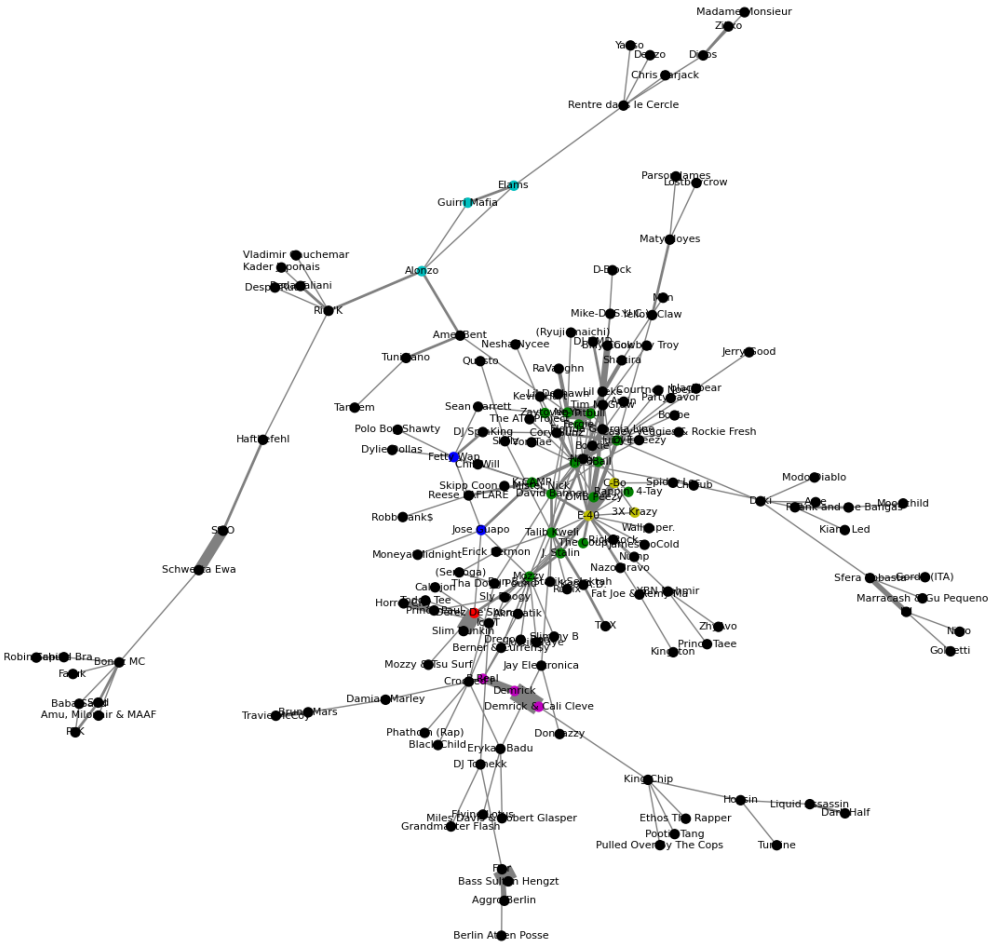

Figura 5: Comunidades de artistas obtenidas con el algoritmo CPM

En general, los resultados obtenidos con el algoritmo Louvain son más satisfactorios que los obtenidos con el algoritmo CPM. Las comunidades detectadas son más grandes y su relación es más fácil de ver a simple vista basándose solo en las colaboraciones entre artistas, que reflejan los arcos del nodo. El género de los artistas no parece haber influido mucho en la detección de las comunidades, ya que los artistas de este grafo pertenecen en su gran mayoría al género rap. Aun así algunos artistas de pop que colaboran con otros del mismo género sí que han formado comunidades propias.

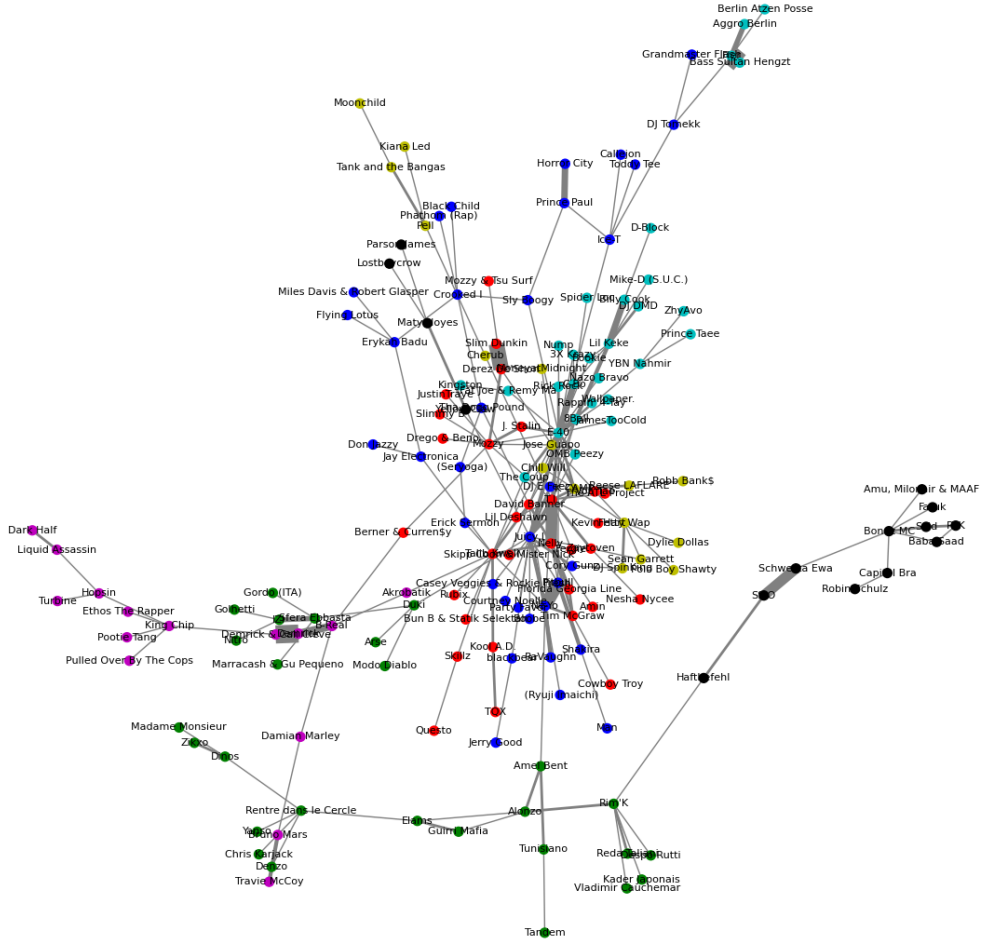

Figura 6: Comunidades de artistas obtenidas con el algoritmo Louvain

5. Conclusiones

En este trabajo se han aplicado técnicas de minería de contenido, uso y estructura a distintos conjuntos de datos del dominio de la música. En el caso de la minería de contenido, se han utilizado algoritmos de clasificación para clasificar canciones en función de su género musical a partir de sus letras, obteniendo resultados buenos para las canciones pertenecientes al género de rap, pero muy malos para las de pop; independientemente del algoritmo de representación de la información o de clasificación.

En la minería de uso, se han utilizado algoritmos de recomendación basados en sesiones para recomendar canciones a partir del historial de escucha de un usuario, obteniendo resultados bastante buenos para los algoritmos de vecinos más cercanos,

pero muy malos para el algoritmo de cadenas de Markov y factorización de matrices. En la minería de la estructura, se han detectado comunidades de artistas a partir de las colaboraciones que han realizado en sus canciones, obteniendo resultados satisfactorios con el algoritmo de Louvain, pero no tan buenos con el algoritmo de CPM.

Para concluir, los resultados obtenidos en los distintos problemas de minería de contenido, uso y estructura han sido variados y satisfactorios en al menos un caso. Esto demuestra la utilidad de los algoritmos de minería web en un dominio tan complejo y amplio como el de la música.

Referencias

[1] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[2] Genius Song Lyrics. URL: `https://www.kaggle.com/datasets/carlosgdcj/genius-song-lyrics-with-language-information` (visitado 15-04-2024).

[3] Matthew Honnibal, Ines Montani, Sofie Van Landeghem y Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. DOI: `10.5281/zenodo.1212303`.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski y Tomas Mikolov. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759*, 2016.

[5] Malte Ludewig y Dietmar Jannach. Evaluation of Session-based Recommendation Algorithms. *CoRR*, abs/1803.09587, 2018. arXiv: `1803.09587`. URL: `http://arxiv.org/abs/1803.09587`.

[6] Malte. Rn5l/Session-Rec, 14 de abril de 2024. URL: `https://github.com/rn5l/session-rec` (visitado 15-04-2024).

[7] Jeroen Ooms. *cld3: Google's Compact Language Detector 3*. R package version 1.6.0. 2024. URL: `https://docs.ropensci.org/cld3/%20https://github.com/ropensci/cld3%20https://ropensci.r-universe.dev/cld3`.

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

[9] Radim Řehůřek y Petr Sojka. Software Framework for Topic Modelling with Large Corpora. English. En *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, páginas 45-50, Valletta, Malta. ELRA, mayo de 2010.

[10] Nils Reimers e Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, noviembre de 2019. URL: `http://arxiv.org/abs/1908.10084`.