

**SEGMENTACIÓN AUTOMÁTICA DE LETRAS
DE CANCIONES POPULARES
IBEROAMERICANAS PARA UNA
PLATAFORMA DE ANÁLISIS MUSICAL**

**AUTOMATIC IBEROAMERICAN FOLK SONG
LYRICS SEGMENTATION FOR A MUSICAL
ANALYSIS PLATFORM**



**TRABAJO FIN DE GRADO
CURSO 2024-2025**

**AUTORA
MARÍA SACHEZ CARRASCO**

**DIRECTORES
CARLOS LEÓN AZNAR, ALEJANDRO ROMERO HERNÁNDEZ**

**GRADO EN DESARROLLO DE VIDEOJUEGOS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID**

**SEGMENTACIÓN AUTOMÁTICA DE LETRAS
DE CANCIONES POPULARES
IBEROAMERICANAS PARA UNA
PLATAFORMA DE ANÁLISIS MUSICAL**

**AUTOMATIC IBEROAMERICAN FOLK SONG
LYRICS SEGMENTATION FOR A MUSICAL
ANALYSIS PLATFORM**

TRABAJO DE FIN DE GRADO EN DESARROLLO DE VIDEOJUEGOS

**AUTORA
MARÍA SACHEZ CARRASCO**

**DIRECTORES
CARLOS LEÓN AZNAR, ALEJANDRO ROMERO HERNÁNDEZ**

**CONVOCATORIA: JUNIO 2025
CALIFICACIÓN: 9.7**

**GRADO EN DESARROLLO DE VIDEOJUEGOS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID**

20 DE JUNIO DE 2025

DEDICATORIA

A mi madre, por enseñarme como a una alumna, guiarme como a una hija y quererme como a un alma gemela.

A mi padre, amante de la poesía y engendrador de sueños.

Hoy te vi, papá.
Al fijarme, observé
sólo un espejo.

María

A mí misma, por mi fuerza, dedicación y pasión.

AGRADECIMIENTOS

Gracias a Rome, por desenterrarme como a un yacimiento romano, por recorrer este camino conmigo sin prejuicios y regalarme el amor por la investigación.

Gracias a Carlos, por ver en mí la ilusión que ni yo misma conocía, por ayudarme e inspirarme siempre y cederme un hueco en el escenario.

Gracias a Jaime, por hacer que nazca en mí la pasión por el aprendizaje, por enseñarme a confiar en mis capacidades y a alzar la voz.

Gracias a Miky, por mostrarme que no hay que temer a lo nuevo y que merece la pena librarse de las cadenas para crear obras hermosas.

Gracias a la banda de la facultad, por haber sido lo mejor de mi año, por recordarme el placer de vivir y por acogerme como a una igual.

Gracias a Ainhoa, por darme la amistad de una hermana y compartir mi sensibilidad.

RESUMEN

Ante los avances tecnológicos actuales, surge la necesidad de automatizar el guardado y análisis del estudio de obras musicales. El material que existe hoy en día es completamente artesanal, y estructurarlo a mano es muy costoso. En este contexto surge el proyecto *EA-DIGIFOLK*, encargado del análisis, la preservación y la divulgación de cultura, sobre todo en el ámbito musical. Como parte del proyecto, aparece la idea de analizar la estructura lírica de las canciones populares de distintas culturas. Por ello, en este estudio se ha investigado y aplicado el uso de *LLMs* para segmentar estas canciones por versos. Tras realizar una serie de pruebas y analizar las respuestas obtenidas, se puede concluir que esta solución genera resultados favorables, aunque hace falta más experimentación para que el procedimiento sea robusto.

Palabras clave

LLM, canciones folclóricas, Inteligencia Artificial, versificación, poesía, aprendizaje automático, español, música, fine tuning.

ABSTRACT

Given current technological advances, there is a growing need to automate the storage and analysis of musical works. The existing material is completely handcrafted and structuring it by hand is very expensive. The *EA-DIGIFOLK* project was created in this context, tasked with the analysis, preservation, and dissemination of culture, particularly in the musical field. As part of the project, the idea of analyzing the lyrical structure of popular songs from different cultures appeared. We investigated and used *LLMs* to segment these songs by verse. After testing and analyzing the responses obtained, we can conclude that this solution can be useful, although further experimentation is needed to ensure the procedure is correct.

Keywords

LLM, folkloric songs, Artificial Intelligence, versification, poetry, machine learning, Spanish, music, fine tuning.

ÍNDICE DE CONTENIDOS

Capítulo 1 - Introducción.....	1
1.1 Motivación.....	3
1.2 Objetivos.....	4
1.2.1 Objetivos generales.....	4
1.2.2 Objetivos específicos.....	5
1.3 Plan de trabajo.....	6
1.3.1 Planificación de tareas.....	6
1.4 Metodología y tecnología.....	8
1.4.1 Tecnología base y herramientas utilizadas.....	8
1.5 Estructura del resto del documento.....	11
Capítulo 2 - Estado de la cuestión.....	13
2.1 Canciones populares y estructuras poéticas.....	13
2.2 Procesamiento automático del lenguaje natural.....	14
2.2.1 Historia y antecedentes del procesamiento automático del lenguaje natural.....	14
2.2.2 Arquitectura transformer y redes neuronales long short-term memory.....	18
2.2.3 Conexionismo y simbolismo en IA: la estructura de la mente como arte.....	20
2.2.4 Procesamiento automático del lenguaje en letras de canciones populares.....	24
2.3 Modelos extensos de lenguaje (<i>LLMs</i>) y su uso en el análisis y segmentación de canciones y poemas.....	25
2.3.1 Técnicas y metodologías utilizadas para generar y entrenar a los modelos.....	26
2.3.2 Tecnologías y herramientas utilizadas para aplicar las metodologías.....	27

Capítulo 3 - Modelo de análisis de segmentación de canciones.....	31
3.1 <i>LLMs</i> utilizados en el estudio	31
3.2 Preparación previa del <i>dataset</i> a utilizar	31
3.3 Problema derivado de la automatización del conteo de sílabas	32
Capítulo 4 - Segmentación por versos de letras de canciones populares usando <i>LLMs</i>	37
4.1 Segmentador usando un <i>prompt</i> exhaustivo y sin instrucciones de formato.....	37
4.1.1 Modelo segmentador específico creado a partir de Llama2	39
4.1.2 Modelo segmentador específico creado a partir de Gemma3	44
4.1.3 Modelo segmentador específico creado a partir de StableLM2.....	47
4.1.4 Conclusiones generales y soluciones propuestas ante los resultados obtenidos	52
4.2 Segmentador usando plantillas de LangChain, centrado en la segmentación de versos y no su conteo de sílabas	53
4.2.1 Modelo segmentador específico creado a partir de Llama2	55
4.2.2 Modelo segmentador específico creado a partir de Gemma3	60
4.2.3 Modelo segmentador específico creado a partir de StableLM2.....	63
4.2.4 Conclusiones generales y soluciones propuestas ante los resultados obtenidos	68
4.3 Segmentador utilizando <i>fine-tuning</i> con <i>LoRA</i> , centrado en la segmentación de versos y no su conteo de sílabas	68
4.3.1 Llama2	72
4.3.2 Conclusiones y soluciones propuestas a partir de los resultados de Llama2	76
Capítulo 5 - Servicio web para facilitar la interacción del usuario con las funcionalidades implementadas.....	77
Capítulo 6 - Discusión y análisis de limitaciones.....	79
Capítulo 7 - Conclusiones y trabajo futuro.....	81

Introduction.....	83
Conclusions and future work.....	91
<i>Apéndice A - Spanish folk music lyrics segmentation with large language models and verse metrics..</i>	<i>97</i>

ÍNDICE DE FIGURAS

Figura 1. Planificación del curso representada mediante un diagrama de Gantt.....	7
Figura 2. Organización del perceptrón según Rosenblatt.....	15
Figura 3. Perceptrón Mark I de Rosenblatt.....	16
Figura 4. Codificador en una red neuronal recurrente	19
Figura 5. Decodificador en una red neuronal recurrente	20
Figura 6. Ilustraciones de Ramón y Cajal de dos teorías contrastantes sobre la composición cerebral: la teoría reticular y la doctrina de la neurona que él propuso	21
Figura 7. Secuencia de pasos que debería realizar el Advice Taker según John McCarthy.....	23
Figura 8. Número de veces que se ha repetido cada segmentación en las respuestas de Llama2.....	41
Figura 9. Número de veces que se ha repetido cada conteo en las respuestas de Llama2	43
Figura 11. Número de veces que se ha repetido cada conteo en las respuestas de Gemma3.....	46
Figura 12. Número de veces que se ha repetido cada segmentación en las respuestas de StableLM2	49
Figura 13. Número de veces que se ha repetido cada conteo en las respuestas de StableLM2.....	50
Figura 14. Número de veces que se ha repetido cada segmentación en las respuestas de Llama2	57
Figura 15. Número de veces que se ha repetido cada conteo en las respuestas de Llama2.....	59
Figura 17. Número de veces que se ha repetido cada conteo en las respuestas de Gemma3.....	62
Figura 18. Número de veces que se ha repetido cada segmentación en las respuestas de StableLM2	65
Figura 19. Número de veces que se ha repetido cada conteo en las respuestas de StableLM2	66
Figura 20. Capturas de la interfaz del servicio web	77
Figura 21. Ejemplo de uso del servicio web solicitando una canción del dataset	78
Figura 22. Ejemplo de uso del servicio web enviando la letra completa de la canción.....	78

Figura 23. Planning represented by a Gantt chart88

ÍNDICE DE TABLAS

Tabla 1. Algunas diferencias claves entre IA conexionista y simbólica	24
Tabla 2. LLMs utilizados en el estudio	31
Tabla 3. Ejemplo de canción almacenada en el dataset.....	32
Tabla 4. Distintos conteos de versos realizados por cada biblioteca de Python	35
Tabla 5. Ejemplo de formato y contenido final del dataset a utilizar.....	35
Tabla 6. Cinco respuestas del modelo segmentador de Llama2	40
Tabla 7. Segmentaciones distintas devueltas por el modelo de Llama2	40
Tabla 8. Ejemplo de conteo sin sentido por el segmentador de Llama2	42
Tabla 9. Algunos conteos distintos devueltos por el segmentador de Llama2	42
Tabla 10. Cinco respuestas del modelo segmentador de Gemma3.....	44
Tabla 11. Todos los conteos distintos devueltos por el segmentador de Gemma3	45
Tabla 12. Cinco respuestas del modelo segmentador de StableLM2	47
Tabla 13. Segmentaciones distintas devueltas por el modelo de StableLM2	48
Tabla 14. Algunos conteos distintos devueltos por el segmentador de StableLM2.....	49
Tabla 15. Características de las salidas de los modelos usando un prompt exhaustivo	52
Tabla 16. Cinco respuestas del modelo segmentador de Llama2.....	56
Tabla 17. Segmentaciones distintas devueltas por el modelo de Llama2.....	57
Tabla 18. Algunos conteos distintos devueltos por el segmentador de Llama2	58
Tabla 19. Segmentaciones distintas devueltas por el modelo de Gemma3.....	61
Tabla 20. Segmentaciones distintas devueltas por el modelo de Gemma3.....	61
Tabla 21. Todos los conteos distintos devueltos por el segmentador de Gemma3	62
Tabla 22. Segmentaciones distintas devueltas por el modelo de StableLM2	64

Tabla 23. Segmentaciones distintas devueltas por el modelo de StableLM2	65
Tabla 24. Algunos conteos distintos devueltos por el segmentador de StableLM2	66
Tabla 25. Características de las salidas de los modelos usando plantillas de LangChain	68
Tabla 26. Canciones utilizadas para probar los modelos entrenados	71
Tabla 27. Seguimiento resumido del error de entrenamiento del modelo Llama2	72

Capítulo 1 - Introducción

La música es una de las manifestaciones artísticas más extendidas por todo el planeta. La humanidad ha producido obras creativas desde tiempos inmemoriales, evolucionando y generando nuevas corrientes, estilos y técnicas. Este arte no ha variado sólo a lo largo de la cronología de nuestra historia, sino que cada cultura y lugar ha ido adoptando una serie de estilos característicos, con rasgos únicos e inspiraciones de distintos orígenes.

Tras siglos de creación artística musical, la cantidad de contenido existente es inmensurable. Este enorme montón de arte sigue creciendo a un ritmo exponencial, ya que las herramientas para la creación y producción de música van en aumento, facilitando la diversidad en la recopilación universal de obras existentes.

Aunque hoy en día mantener a salvo el contenido audiovisual está al alcance de todo el mundo, no siempre ha existido la posibilidad de guardar las obras artísticas para preservarlas durante el paso de los años. Esto genera un problema actual.

A la hora de recopilar y analizar música, no siempre se cuenta con los recursos necesarios. La música folclórica es una manifestación artística diversa y personal que se va transmitiendo de generación en generación. Cada región y cultura mantiene su propio esquema, aunque muchos de estos estilos obtienen ciertos rasgos de otros pueblos y gentes, lo que aporta riqueza y diversidad a sus obras. El principal inconveniente que surge al tratar de recopilar información sobre las canciones populares es que se transmiten mayoritariamente por vía oral. Esto hace que gran parte del contenido cantado se pierda o se conserve parcialmente al no estar escrito. Por otra parte, aunque cada pueblo es un mundo, muchas canciones se transmiten de unas gentes a otras, lo que desemboca en variaciones en la interpretación de estas obras, aunque la base se mantenga.

En este contexto se desarrolla el *proyecto "EA-DIGIFOLK-An European and Ibero-American approach for the digital collection, analysis and dissemination of folk music (101086338)"*, financiado

por la Unión Europea, en el que participa la UCM¹, cuyo principal objetivo es preservar y compartir la cultura folclórica popular europea e iberoamericana, sobre todo del ámbito musical. *EA-DIGIFOLK* busca crear una plataforma que facilite la difusión de esta cultura a entornos educativos y públicos. Por otra parte, y más en relación con este proyecto, otro de los principales objetivos es el análisis musical y etnomusicológico del material audiovisual y escrito recopilado. Este trabajo se ha llevado a cabo como parte de este proyecto de investigación. Mientras se desarrollaba el proyecto europeo *EA-DIGIFOLK* apareció la posibilidad de redactar un artículo científico en relación con los problemas presentados en esta investigación. El artículo completo se encuentra adjunto al final de esta memoria.

La necesidad de desarrollar este proyecto de investigación nace del interés por preservar la cultura popular que lleva viajando tanto tiempo entre generaciones, aportándonos información sobre las gentes, las tradiciones y los lugares. Esto implica la digitalización de contenido artístico y un análisis de este para completar la información parcial o perdida de la mejor manera posible. Ya que nos encontramos en una era con infinidad de avances tecnológicos, especialmente en el ámbito de la inteligencia artificial, que va creciendo continuamente de manera exponencial, el procesamiento del lenguaje natural y la automatización de tareas de análisis encuentra en los *LLMs* una herramienta clave para mejorar y facilitar la investigación en estos campos.

La inteligencia artificial nunca ha tenido tanto peso como hoy día. Los avances que se han ido realizando durante los últimos años han permitido que herramientas y modelos cada vez más precisos y accesibles se utilicen para una gran variedad de tareas en diversos campos. Su capacidad para procesar grandes volúmenes de datos e identificar patrones, entre otras muchas otras funcionalidades, ha permitido crear aplicaciones que reformulan nuestra manera de interactuar con la información y con el mundo que nos rodea.

Las dos funciones de la inteligencia artificial mencionadas en el párrafo anterior son precisamente dos de las habilidades necesarias para realizar análisis y predicciones para preservar y recuperar información sobre la cultura popular.

¹ <https://cordis.europa.eu/project/id/101086338>

Por otra parte, los algoritmos tradicionales no pierden importancia. Con un buen estudio e implementación pueden conseguirse resultados próximos a los esperados. Es posible acercarse a una solución adecuada mediante algoritmos tradicionales. Sin embargo, estos resultados nunca proporcionan una respuesta perfecta para todos los casos analizados. Cada canción tiene una estructura distinta y, como se ha mencionado anteriormente, pueden incluso aparecer variaciones a partir de la transmisión oral de estas obras. Esto complica mucho el proceso de segmentación de las letras; saber dividir las perfectamente mediante un algoritmo tradicional no es algo trivial.

Por ello, este proyecto está mayoritariamente orientado a la investigación y desarrollo de *LLMs* especializados que sean capaces de analizar, segmentar y comparar letras de canciones populares. Empleando estos modelos, los resultados obtenidos no siempre se acercan a los esperados. Además, hay montones de técnicas y metodologías que pueden utilizarse para sacarle provecho a los modelos de lenguaje. Durante este proceso se indagará con el fin de encontrar la implementación óptima que se acerque más a la solución que se busca.

Esta investigación es una pequeña extensión del proyecto europeo *EA-DIGIFOLK*, cuyo objetivo principal la creación de una plataforma digital abierta que recopile, analice y difunda música folklórica de las culturas europea e iberoamericana, facilitando así el acceso público a recursos multimedia como partituras y grabaciones, promoviendo la comprensión de la música folklórica desde perspectivas etnomusicológicas, musicales y educativas.

1.1 Motivación

En este contexto surge el problema concreto del análisis de textos de canciones populares. Automatizar esta tarea no es trivial, ni utilizando algoritmos tradicionales ni con modelos de lenguaje.

Para empezar, como ocurre en la poesía, muchas reglas ideadas como estándar, como el uso de la sinalefa o la sinéresis [1], varían en función del contexto, de las métricas del resto de versos o del tipo de estrofa manejada en cada caso. Esto hace imposible una generalización que proporcione una solución correcta en cada uno de los casos analizados, ya que la aplicación o no aplicación de estas licencias poéticas dependen mucho del contexto, de la época de creación, del autor, de la estructura de la obra, de los acentos regionales, la pronunciación, etc.

Además, los textos de canciones populares son cantados. Esto hace que los versos y las sílabas no sigan una estructura concreta determinada por reglas gramaticales, sino que se amoldan a las melodías y a los ritmos. En ocasiones, cada nota interpretada puede ir acompañada de una sílaba diferente no repetida, mientras que, en otros casos, una misma sílaba puede pronunciarse reiteradamente a lo largo de distintas figuras rítmicas como recurso artístico. Al escribir la letra de una canción, estas variaciones en la transmisión oral de la obra no se ven reflejadas en la transcripción. Los algoritmos tradicionales y *LLMs* no pueden “escuchar” la música de estas obras, o al menos, no sin ser entrenados o pasar por un proceso específico que los prepare para ello.

Un problema específico de los *LLMs* es que generalmente son entrenados con lenguaje cotidiano, no con textos poéticos o de canciones. Este tipo de material tiene estructuras especiales que se diferencian enormemente del resto de tipos de contenido textual. Por lo tanto, estos modelos no están completamente preparados de base para afrontar tareas de análisis y segmentación de versos y aportar resultados correctos o válidos en todos los casos.

Aquí nace la motivación del proyecto, donde se pretende analizar diversas formas de automatizar la segmentación de canciones populares por versos y el conteo por sílabas de estos mediante algoritmos tradicionales y distintas técnicas aplicadas a *LLMs*. Encontrar una solución exacta no es posible con la propuesta actual, pero analizar los resultados obtenidos en este estudio permitirá abordar el problema de otra manera, con mejores recursos y mucho más conocimiento al respecto.

1.2 Objetivos

1.2.1 Objetivos generales

El objetivo principal de este proyecto es investigar e implementar distintas formas de manejar y entrenar *LLMs* para estudiar y comparar sus comportamientos y rendimientos. Como se ha tratado en la introducción, la automatización de ciertas tareas de análisis como la segmentación de las canciones por versos no es trivial, y mucho menos cuando se usan modelos de lenguaje. Con este estudio se busca implementar una serie de sistemas de análisis y segmentación entre canciones populares partiendo de letras de obras sin versificar. La principal finalidad de esta propuesta es encontrar la implementación óptima que se acerque más a los resultados esperados.

Ante la existencia de distintas técnicas, herramientas y metodologías, otro de los objetivos de este proyecto es la comparación entre los resultados recibidos por parte de los *LLMs* y de los algoritmos tradicionales. Esto no sólo permitirá valorar qué enfoque se adapta mejor a la segmentación y análisis de letras de canciones, sino que también permitirá comprender mejor las limitaciones y posibilidades de cada método.

1.2.2 Objetivos específicos

- Implementar la técnica óptima que permita segmentar letras de canciones populares de la manera que más se acerque a la original.
- Implementar la técnica óptima que permita analizar y comparar letras de canciones populares con los resultados más veraces.
- Las técnicas que se van a aplicar para comparar los distintos resultados obtenidos son las siguientes:
 - *Prompt* simple: moldear un *prompt* que permita crear una serie de modelos específicos dedicados a desempeñar las tareas mencionadas con anterioridad.
 - Plantillas de *LangChain*: mejorar los modelos creados usando plantillas preparadas para realizar peticiones directas a los modelos locales especializados en las tareas.
 - *RAG*: proporcionar a los modelos información extra relacionada con la petición enviada por el usuario para que el *LLM* base en ella su respuesta.
 - *Fine-tuning*: entrenar a los *LLMs* con la base de datos proporcionada con el objetivo de obtener respuestas que se asemejen más a la información de entrenamiento.
 - *Reflexión*: hacer “reflexionar” a los modelos, siendo así conscientes de su respuesta, dándoles espacio para mejorar la salida y asemejarla más a la información esperada.
- Comparar los resultados obtenidos por parte de los modelos de lenguaje y de algoritmos tradicionales.

- Proporcionar a los usuarios un servicio accesible, comprensible e intuitivo que les permita analizar, segmentar y comparar las canciones de la base de datos.

1.3 Plan de trabajo

Para poder completar todos los objetivos propuestos, estos han sido divididos en una serie de tareas concretas, facilitando así el proceso de investigación e implementación:

- Implementación de modelos personalizados a partir de modelos existentes y *prompts* específicos para analizar, segmentar y comparar letras de canciones.
- Uso de los modelos personalizados para analizar, segmentar y comparar letras de canciones únicamente con un *prompt* simple.
- Uso de los modelos personalizados para analizar, segmentar y comparar letras de canciones usando plantillas de *LangChain*.
- Uso de los modelos personalizados para analizar, segmentar y comparar letras de canciones aplicando *RAG* con letras de las canciones de la base de datos.
- Uso de los modelos personalizados para analizar, segmentar y comparar letras de canciones aplicando *fine-tuning*.
- Uso de los modelos personalizados para analizar, segmentar y comparar letras de canciones usando “reflexión”.
- Comparación de los resultados obtenidos por parte de los modelos de lenguaje y de algoritmos tradicionales.
- Puesta en marcha de un servicio web que integre todas las funcionalidades implementadas para facilitar el acceso y uso por parte del usuario.

1.3.1 Planificación de tareas

En esta sección se muestra el diagrama de Gantt en el que se han distribuido las distintas tareas a realizar a lo largo del transcurso del proyecto, basado en un sistema de objetivos delimitados por reuniones con los tutores e integrantes del proyecto *EA-DIGIFOLK*:



Figura 1. Planificación del curso representada mediante un diagrama de Gantt

A lo largo del curso han surgido eventos e imprevistos. Este diagrama de Gantt se ha actualizado para hacerlo fiel a la realidad y a cómo ha sido finalmente la organización y las tareas realizadas en el tiempo estipulado.

Durante el proceso de investigación se realizaron al menos dos reuniones cada mes con los tutores y, en ocasiones, con parte del equipo del proyecto *EA-DIGIFOLK*. El objetivo de estos encuentros era hacer un seguimiento del trabajo desarrollado hasta el momento, resolver dudas emergentes y aportar más ideas al planteamiento.

Se ha trabajado en este proyecto europeo colaborando con otras instituciones y empresas como la Universidad de Salamanca, la Universidad de Coímbra, la Universidad Nacional Autónoma de México o la empresa *PTNera Consulting*, mayoritariamente centrada en el desarrollo de proyectos relacionados con la divulgación e investigación científica, cultural, social y educativa.

Uno de los principales problemas que surgieron durante el desarrollo fue la necesidad de cambiar la tecnología base utilizada hasta el momento por una serie de problemas.

Además, aunque al comienzo del proyecto se propuso implementar *RAG* como técnica a aplicar a los modelos propuestos, se ha decidido finalmente no incluirla ante la complicación del proceso y la ausencia de resultados.

1.4 Metodología y tecnología

Para cumplir con los objetivos propuestos para el proyecto, se ha ideado un sistema autogestionado adaptado a la investigación y desarrollo individual. La metodología utilizada se ha basado en un itinerario de trabajo de objetivos semanales y fases de desarrollo incrementales.

1.4.1 Tecnología base y herramientas utilizadas

Con el objetivo de mantener un control exhaustivo sobre el código y material trabajado durante el desarrollo del proyecto, se han utilizado GitHub y Google Drive como sistemas de control de versiones.

1.4.1.1 GitHub

GitHub se ha utilizado como principal sistema de control de versiones para manejar toda la implementación del manejo de modelos.

GitHub presenta una interfaz intuitiva que facilita su uso y, además, permite mantener el código organizado, actualizado y al alcance de todos los miembros del proyecto. El seguimiento íntegro de cambios proporciona seguridad, permitiendo consultar en cualquier momento cuáles han sido las modificaciones realizadas sobre la implementación.

Se han usado principalmente dos repositorios:

- [Repositorio](#) principal de este proyecto, donde se encuentra toda la implementación que incluye las distintas metodologías utilizadas para la segmentación de canciones populares por versos mediante *LLMs* y el código pertinente para poner en marcha el servicio web que conecta todas las funcionalidades y permite que el usuario pueda acceder fácilmente a ellas.
- [Repositorio](#) donde se encuentra toda la implementación de comparación de canciones mediante algoritmos tradicionales, así como otro servicio web desde donde sólo puede se puede acceder a la comparación de textos.

1.4.1.2 Google Drive

Google Drive se ha utilizado mayoritariamente para almacenar y acceder a los *datasets* de canciones proporcionados, así como información similar. También se ha usado a la hora de redactar el artículo científico presentado a *Digital Humanities 2025*, por ofrecer un entorno colaborativo desde el que trabajar simultáneamente con el resto del equipo, pudiendo ver todos los cambios de cada uno de los integrantes en tiempo real.

Esta herramienta es ideal para almacenar archivos de forma remota, permitiendo que otros participantes del proyecto accedan a ellos. Por otra parte, brinda comodidad a la hora de redactar de manera paralela y en tiempo real, proporcionando acceso continuo al historial de cambios realizados y la posibilidad de realizar anotaciones sobre el texto escrito. Que todo el equipo tenga una versión siempre actualizada del proyecto en el que se está trabajando es esencial y *Google Drive* es una herramienta perfecta que asegura la sincronización de sus ficheros.

1.4.1.3 Python

Para desarrollar las implementaciones con las que preparar, entrenar y manejar los modelos de lenguaje se ha optado por utilizar *Python*. Esta decisión se ha tomado por todas las ventajas que ofrece este lenguaje de alto nivel. *Python* permite utilizar una infinidad de bibliotecas programadas por la comunidad. Estos recursos son muy diversos; pueden encontrarse *APIs* para llevar a cabo casi cualquier tipo de aplicación, desde procesamiento básico de texto hasta síntesis musical pasando por uso de *LLMs*.

A la hora de desarrollar este proyecto se han utilizado varias bibliotecas, aunque algunas de ellas no han llegado a la implementación final. Algunas de estas bibliotecas han sido las siguientes:

- Ollama: permite utilizar modelos de lenguaje locales, crearlos, generar sistemas de chat con los *LLMs*... Esto facilita la comunicación con los modelos y su integración en aplicaciones que manejan otras funcionalidades, pudiendo, por ejemplo, retocar o preparar el *prompt* antes de enviarlo al modelo, o aplicar reflexión ante la respuesta proporcionada por el *LLM*.

- LangChain: al igual que Ollama, facilita la creación de aplicaciones que usan *LLMs*, aunque LangChain proporciona muchas más bibliotecas y funcionalidades. LangChain también incluye clases y funciones que permiten conectar Ollama y sus modelos locales con la aplicación, pudiendo acceder a ellos y realizar peticiones a través de las funcionalidades que ofrece.

- Flask: utilizada para crear aplicaciones web. En el proyecto, esta biblioteca se ha empleado para desarrollar el servicio web que permite centralizar todas las funcionalidades implementadas y ofrecerlas al usuario como una herramienta intuitiva y fácil de usar.

1.4.1.4 Ollama

La principal necesidad de este proyecto es la obtención y ejecución de modelos de lenguaje para probar las distintas implementaciones. Aunque utilizar los *LLMs* de manera remota es una buena opción, para la creación de modelos personalizados, el guardado local facilita el proceso y uso.

[Ollama](#) permite almacenar y utilizar modelos de lenguaje de manera local, con la opción de obtener una gran diversidad de *LLMs*.

1.4.1.5 Modelos de lenguaje utilizados en el estudio

En esta sección se presentan los tres modelos de lenguaje utilizados en este proyecto y se profundizan en sus características y ventajas. Aunque los tres han sido entrenados para desempeñar tareas relacionadas con el procesamiento de lenguaje natural, cada uno tiene sus propias particularidades.

- Llama2 [2]: modelo de lenguaje desarrollado por *Meta*. Una de las grandes ventajas que aporta es su entrenamiento a partir de diversos idiomas, lo que lo hace un modelo multilingüe apto para desarrollar las tareas planificadas en este proyecto. La interacción con este modelo se basa en la comunicación por texto, recibiendo un *input* textual y devolviendo un *output* con el mismo formato.
- Gemma3 [3]: modelo de lenguaje desarrollado por *Google*. Este *LLM* no sólo es capaz de procesar texto, sino que también puede recibir imágenes e incluso audio. Sus respuestas siempre se formulan con texto. Al igual que Llama2, Gemma3 también ha sido entrenado en varios idiomas, incluyendo el español.
- StableLM2 [4]: modelo de lenguaje desarrollado por *Stability AI*. Este *LLM* también ha sido entrenado en algunos idiomas, aunque no en tantos como los dos modelos mencionados anteriormente. Al igual que Llama2, este modelo sólo se comunica mediante texto. Se ideó para ser entrenado con *fine-tuning*.

1.4.1.6 Unslloth

Una de las soluciones propuestas para mejorar el rendimiento de los modelos a la hora de analizar y segmentar canciones ha sido entrenar a los *LLMs* para que puedan desempeñar tareas concretas.

[Unslloth](#) ofrece la posibilidad de entrenar a estos modelos mediante *fine-tuning* para mejorar su rendimiento. Además, mejora y agiliza el proceso de entrenamiento gestionando mediante *software* el uso de *GPUs*.

1.4.1.7 Zotero

Con el objetivo de incluir y organizar toda la bibliografía que ha servido de inspiración y apoyo para este proyecto, se ha utilizado Zotero como gestor bibliográfico. El uso de esta herramienta es intuitivo y cómodo, permitiendo añadir automáticamente artículos y libros directamente desde el propio navegador, manteniendo la recopilación de información siempre sincronizada.

1.5 Estructura del resto del documento

En esta sección se ofrece un resumen de los apartados del proyecto, presentando de manera esquemática los temas a tratar y su orden de aparición.

En el [Capítulo 2](#) se presenta una introducción del contexto en el que se desarrolla este proyecto, un poco de la historia del procesamiento del lenguaje natural, conceptos importantes sobre métricas y reglas poéticas y la presentación de los modelos de lenguaje que se utilizan para desarrollar las pruebas de implementación.

Para comprender la preparación previa a las pruebas realizadas, en el capítulo de [Capítulo 3](#) se presentan los *LLMs* utilizados y se habla del *dataset* del que se han sacado los ejemplos analizados en cada implementación.

Las implementaciones preparadas para probar el comportamiento y rendimiento de los modelos comienzan a aparecer en el [Capítulo 4](#). Aquí se puede observar cómo se han llevado a cabo una serie de pruebas con modelos de lenguaje usando *prompts* y plantillas y, posteriormente, entrenándolos para obtener respuestas que se asemejen más a las esperadas.

Para que los usuarios puedan acceder a las funcionalidades implementadas y realizar peticiones a los modelos se ha implementado un servicio web que se presenta en el [Capítulo 5](#).

Antes de concluir con el proyecto, se ha realizado una discusión con los resultados obtenidos en el [Capítulo 6](#), donde se examinan a fondo las razones por las que los modelos ofrecen distintas respuestas y varían su rendimiento.

Finalmente, tras comprobar y comprender los resultados obtenidos, en el [Capítulo 7](#) se presentan las conclusiones de la investigación donde, además de establecer premisas a partir de la discusión previa, se presentan líneas de trabajo futuras que permitan avanzar y expandir el proyecto.

Capítulo 2 - Estado de la cuestión

Esta sección está dedicada a tratar tres puntos principales en los que se puede resumir el problema expuesto y cómo se ha abordado para este proyecto, teniendo como objetivo comprender mejor la propuesta de la solución.

Por una parte, se aclararán conceptos relacionados con las estructuras poéticas con el objetivo de conocer cuáles son las estructuras poéticas clásicas y más utilizadas. De esta manera puede comprenderse más claramente el problema que presenta la necesidad de analizar y segmentar las canciones.

En segundo lugar, se hablará del procesamiento automático de lenguaje natural por su relación directa con los temas a tratar en este proyecto. Se tratará su historia desde su origen, pasando por los eventos más significativos y personas que han aportado sus investigaciones y conocimientos a la materia. En esta sección se explicarán, además, conceptos necesarios para comprender mejor el funcionamiento y enfoque de este proyecto.

Por último, y en relación con el punto anterior, se tratará la aplicación del procesamiento del lenguaje natural mediante *LLMs* y las metodologías utilizadas en este proyecto.

2.1 Canciones populares y estructuras poéticas

Los datos manejados para aplicar las distintas técnicas y metodologías son diversos. Hay una gran colección de estructuras poéticas de orígenes distintos y, en algunos casos, desconocidos. En ocasiones, las estrofas almacenadas ni siquiera pueden clasificarse dentro de las categorías ya existentes; las hay de verso libre, y otras siguen estructuras base tradicionales, pero aplican ligeros cambios y matices.

Sin embargo, cabe destacar que, en la interpretación lírica de canciones, las métricas no dejan de ser una manera de materializar y organizar las letras cuando estas no son cantadas. Intentar definir las estrofas de las canciones populares de manera absoluta y fija puede funcionar en algunos casos, pero no puede usarse como generalización para todas las obras. En la teoría, la letra de una canción puede analizarse como una estructura poética [5], teniendo en cuenta las sílabas y el número

y longitud de sus versos, entre muchas otras cosas; pero esto no es aplicable a todas las obras musicales.

Esta ambigüedad presente al analizar letras de canciones como estructuras poéticas teóricas dificulta enormemente la automatización del conteo de sílabas. Si la tarea ya presenta cierta complicación al tener que decidir entre aplicar o no ciertas licencias poéticas de manera manual, hacer de esto un proceso automático parece, a simple vista, algo imposible.

2.2 Procesamiento automático del lenguaje natural

2.2.1 Historia y antecedentes del procesamiento automático del lenguaje natural

El procesamiento automático del lenguaje natural lleva estudiándose durante más de 70 años. Ya en 1950 *Alan Turing* ideó y propuso el ahora conocido como “test de *Turing*” [6], donde se formula y discute la pregunta que cuestiona la capacidad de pensamiento de las máquinas. En esta prueba, con el afán de esclarecer la duda “¿*Pueden pensar las máquinas?*”, están involucrados tres individuos: un interrogador humano y dos participantes ocultos que responden a las peticiones que el interrogador realiza a ciegas: un humano y una máquina. El principal objetivo de este experimento hipotético es determinar si el interrogador puede o no distinguir a los dos participantes, determinando cuál es el humano y cuál es la máquina. De esta forma se podría evaluar si la máquina tiene la capacidad de pensar y comportarse de manera inteligente.

Este estudio, *Turing* presenta infinidad de propuestas interesantes que sirvieron de antecedente para aplicaciones y metodologías utilizadas hoy en día que han servido para mejorar los modelos de lenguaje actuales y su aprendizaje. Por ejemplo, *Turing* propone que, en lugar de intentar simular una mente adulta en una máquina, se debería partir de la de un infante, para expandir así su conocimiento poco a poco, como ocurre con los humanos. Esto anticipó la actual visión que tenemos sobre el aprendizaje automático.

Sin embargo, *Turing* no fue el único que se basó en el funcionamiento humano para proponer una solución a un problema en este ámbito. Poco después, en esa misma década, *Frank Rosenblatt* comenzó a investigar y a dirigir sus primeros trabajos sobre perceptrones. Al igual que la de *Turing*, esta idea estaba directamente relacionada con el aprendizaje automático. El planteamiento del

perceptrón de *Rosenblatt* sí que desembocó de manera más directa en la aplicación del aprendizaje mediante conexiones neuronales similares a las humanas.

El funcionamiento del perceptrón ideado por *Rosenblatt* [7] se basa en una red compuesta por neuronas conectadas entre sí cuya estructura es inicialmente aleatoria. Este sistema, al igual que el humano, presenta cierta plasticidad. Esta característica le permite responder y cambiar al recibir estimulación. Tras pasar cierto tiempo expuestas a esta actividad sensorial, las neuronas se reorganizan en función de su similitud, formando subconjuntos y estructuras más pequeñas. La distribución y activación de las neuronas dentro del perceptrón va evolucionando en función de los refuerzos positivos y negativos que va recibiendo. El objetivo que se pretende alcanzar al alterar esta estructura es conseguir que el perceptrón aprenda y mejore su habilidad para reconocer patrones en las entradas proporcionadas. Esta capacidad desarrollada por el perceptrón es una funcionalidad fundamental en el procesamiento de lenguaje natural. Las investigaciones de *Rosenblatt* en torno al perceptrón están muy relacionadas con las redes neuronales modernas, como las arquitecturas tipo *transformer*.

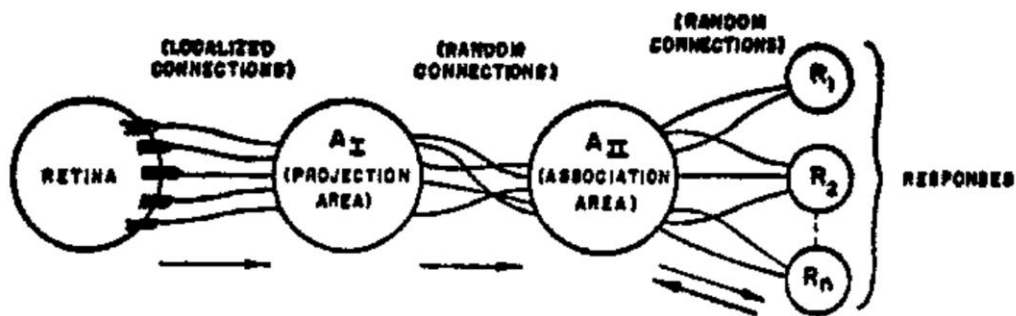


Figura 2. Organización del perceptrón según Rosenblatt

Los proyectos de *Rosenblatt* culminaron con la construcción del *hardware* del Perceptrón *Mark I* en 1960. Este modelo marcó un hito histórico en el campo de la inteligencia artificial y el aprendizaje automático.

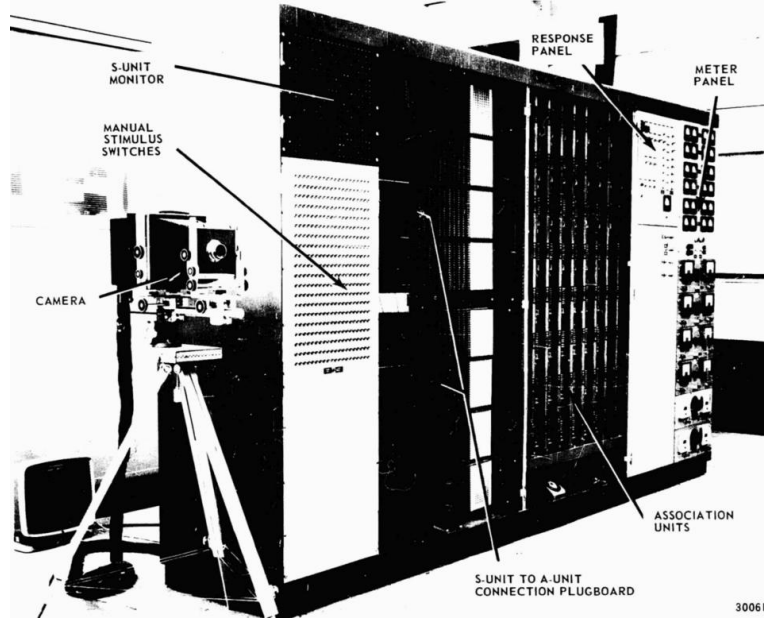


Figura 3. Perceptrón Mark I de Rosenblatt

En esa misma década, *Marvin Minsky* y *Seymour Papert* analizaron a fondo las capacidades del perceptrón [8]. Determinaron que, aunque esta invención supuso un gran avance en el campo del aprendizaje automático y el procesamiento de lenguaje, los perceptrones de una sola capa tenían limitaciones severas, siendo, por ejemplo, incapaces de resolver ciertos problemas no lineales.

Esto desembocó en uno de los Inviernos IA, generando el Abandono del [Conexionismo](#). Esta corriente estuvo profundamente apoyada por muchos científicos y pensadores. Sin embargo, no todas las perspectivas desde las que se percibía a la potencial inteligencia artificial eran positivas, sobre todo ante la llegada del Invierno. *Hubert Dreyfus* tuvo una visión muy pesimista del progreso de la IA, la cual plasmó en sus obras escritas. *Dreyfus* no creía en el avance en el campo de la inteligencia artificial tal y como se proponía en el momento. Él consideraba que este tipo de inteligencia nunca sería capaz de captar el funcionamiento del cerebro humano y cómo esto condiciona nuestra manera de pensar y actuar [9]. Según *Dreyfus*, una máquina nunca conseguiría percibir el mundo tal y como lo hacemos los humanos.

Sin embargo, esta crisis no duró mucho. En la década de 1980 volvió a extenderse el conexionismo como corriente, introduciendo el concepto de retropropagación (*backpropagation*) [10]. Este algoritmo de aprendizaje fue popularizado por *Rumelhart* y comenzó a utilizarse para

entrenar a las redes de neuronas en función del error cometido, iterando desde la última capa de neuronas hacia la primera para ajustar los pesos de las distintas unidades individuales de procesamiento. Aunque el perceptrón multicapa ya se había propuesto con anterioridad, la idea empezó a tomar forma cuando se combinó con el tipo de aprendizaje supervisado presentado con la retropropagación. Además, tras el Invierno, *Minsky y Papert* se atrevieron a publicar una versión expandida de *Perceptrons* [8], donde se enfrentaban a todas las críticas recibidas.

El aprendizaje profundo (*Deep learning*) [11] completó el funcionamiento del perceptrón multicapa. Antes del Invierno, a finales de la década de 1960, el neurocientífico *Shun-ichi Amari* publicó el primer perceptrón multicapa entrenado con el descenso de gradiente estocástico, de cinco capas de neuronas. Mediante el aprendizaje profundo, los modelos transforman la información a procesar en datos cada vez más abstractos, trabajando así sobre unidades mínimas, más fáciles de manejar e interpretar. Cada capa va analizando los datos de manera progresiva, uniendo la información en conjuntos más grandes e identificables a medida que las capas avanzan.

Como se puede comprobar, los avances en este campo no dejan de surgir. El procesamiento del lenguaje natural es un concepto actual y a la vez arcaico; siempre ha sido estudiado. Como se ha podido comprobar al cabo de los años, el procesamiento de lenguaje natural se basa en la comprensión interpretación y generación significativa del lenguaje humano por parte de las máquinas.

Como definen *Augusto Cortez, Hugo Vega y Jaime Pariona* [12]:

“El procesamiento de lenguaje natural (PLN) consiste en la utilización de un lenguaje natural para comunicarnos con la computadora, debiendo esta entender las oraciones que le sean proporcionadas.”

Hoy día, estas técnicas no se limitan al análisis de textos simples, morfológico o sintáctico, sino que se aplican además para desarrollar tareas más complejas, como comprender textos más complejos e interpretar emociones. El análisis de las estructuras poéticas y letras de canciones populares podrían realizarse con modelos de procesamiento de lenguaje natural.

2.2.2 Arquitectura transformer y redes neuronales long short-term memory

Como se ha podido comprobar anteriormente, los sistemas de redes neuronales artificiales han revolucionado el mundo desde su origen, abriendo puertas en el campo del aprendizaje automático y la inteligencia artificial. Sin embargo, el entrenamiento de estas redes desemboca en ciertos problemas, como el desvanecimiento de gradiente [13]. Este inconveniente supone la pérdida gradual del gradiente, impidiendo el aprendizaje de la red de neuronas al no tener una referencia para corregir su desempeño, lo que desemboca en el aumento del error.

Para mitigar este problema se propuso un nuevo tipo de red neuronal recurrente que incorporaba una especie de memoria a corto plazo selectiva, permitiendo así recordar la información útil e importante sin llegar a saturar el sistema, llegando a convertirse en una memoria a largo plazo. Las denominadas “*Long short term memory*” (*LSTM*) [14] se diseñaron específicamente para resolver las limitaciones de las redes neuronales recurrentes tradicionales. Estas unidades están compuestas por una célula de memoria que contiene tres puertas: una de “olvido”, una de entrada, una de salida. La célula es la encargada de gestionar la información que se acumula a lo largo de tiempo, y son las puertas las que deciden cómo se desarrolla el proceso de selección de los datos a almacenar. La puerta de “olvido” determina cuál es la información que debería descartarse de la recopilación actual, la puerta de entrada establece cuáles son los nuevos datos que deberían añadirse a la colección y la puerta de salida define la respuesta que debe proporcionar la red.

Las *LSTM* superaron a las redes neuronales recurrentes tradicionales, preservando el gradiente durante más iteraciones y pudiendo conservar información más a largo plazo. Las aplicaciones de estas redes son muy variadas y útiles. Por ejemplo, en 2006 se estudió el uso de esta arquitectura para realizar cirugías de corazón mediante un robot utilizando redes neuronales recurrentes [15]. A partir de esta investigación pudo concluirse que, tras entrenar a las *LSTM* con *Evolino* [23], un algoritmo específico de entrenamiento de redes, la eficiencia del anudado de suturas en cirugías mínimamente invasivas (*MIS*) mejoraba notablemente con respecto a un control preprogramado. Estas redes también pueden aplicarse en el ámbito de creación y análisis artístico. En 2002, otro estudio presentó la posibilidad de automatizar la composición musical, específicamente de *blues* [16]. Los resultados fueron favorables. Se concluyó que el uso de las *LSTM* frente a las redes neuronales recurrentes tradicionales generaba música coherente y agradable para

muchos usuarios, lo que supuso un gran avance en la composición automática de música a partir de redes neuronales.

Sin embargo, aunque las *LSTM* tienen muchas ventajas, también presentan ciertas limitaciones. Ya que su procesamiento es secuencial, no tiene la capacidad de paralelizar tareas durante su entrenamiento, lo que reduce su eficiencia en comparación con otras arquitecturas. Por otra parte, su rendimiento disminuye a la hora de procesar grandes volúmenes de datos. Estas limitaciones desembocaron en el desarrollo de nuevas arquitecturas más eficientes y escalables, como es el caso de la arquitectura *transformer*.

La arquitectura *transformer* nació en 2017 con el objetivo de mejorar las redes neuronales tradicionales, eliminando los inconvenientes que estas presentaban. Esta solución resultó ser mucho más eficaz a la hora de desempeñar tareas de lenguaje natural, eliminando el procesamiento secuencial de la información proporcionada y permitiendo así manejar la entrada de manera paralela y simultánea [17].

La estructura de esta arquitectura se basa principalmente en un codificador-decodificador (*seq2seq* o *sequence-to-sequence*) que se ocupa de transformar una secuencia de entrada en otra de salida. El codificador se encarga de procesar la entrada recibida, convirtiéndola en un vector que representa la secuencia mediante una serie de valores de tamaño fijo [18].

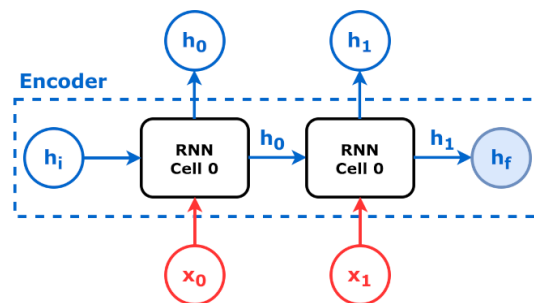


Figura 4. Codificador en una red neuronal recurrente

Por otra parte, el decodificador se ocupa de traducir el vector generado por el codificador para generar una salida, prediciendo una a una las palabras por las que debería estar formada antes de usarla como respuesta.

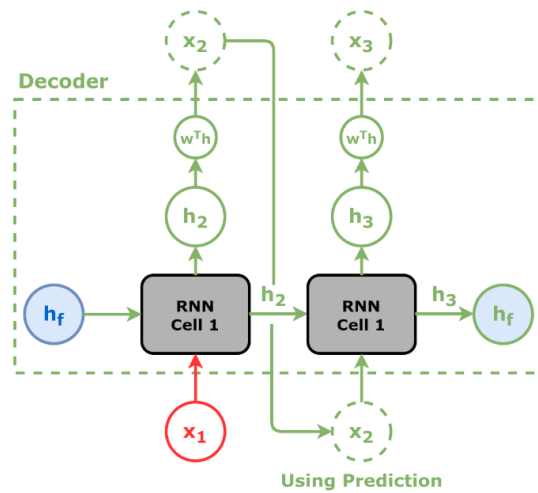


Figura 5. Decodificador en una red neuronal recurrente

Estas estructuras codificador-decodificador se han utilizado en infinidad de campos dentro y fuera del procesamiento de lenguaje natural, como la traducción entre idiomas [19], el reconocimiento del habla o la descripción de imágenes mediante texto.

Inicialmente, la arquitectura *seq2seq* no poseían un mecanismo de atención, pero acabó implementándose con el objetivo de resolver problemas derivados de esas redes. Esta técnica se basa en la atención cognitiva humana, sobre todo en la capacidad para seleccionar la información relevante recibida y descartar la menos significativa [20]. En el ámbito de las redes neuronales, el mecanismo de atención se basa en la ponderación de la información recibida, asignando más o menos peso a las distintas partes de los datos para procesarlos y poder proporcionar una salida precisa. Al analizar oraciones, las palabras que las conforman no comparten el mismo significado en todos los contextos. Gracias a la atención el modelo puede enfocarse en distintas partes de la información, seleccionando lo más significativos. Esto permite que la red pueda procesar secuencias más largas y complejas, ya que esta puede centrarse en la información significativa en lugar de analizar cada detalle.

2.2.3 Conexionismo y simbolismo en IA: la estructura de la mente como arte

Ya se ha tratado con anterioridad el fenómeno del perceptrón, las redes neuronales y cómo siempre se ha tendido a buscar la semejanza con el funcionamiento del cerebro humano para desarrollar los modelos de lenguaje. En este apartado se trata la orientación de la inteligencia

artificial hacia corrientes más conexionistas y simbólicas con el objetivo de parecerse al funcionamiento de la mente de las personas.

El conexionismo es una corriente y enfoque de diversas ciencias y disciplinas, como la inteligencia artificial, la ciencia cognitiva o la neurociencia, que explica los procesos mentales como conjuntos de redes formados por unidades sencillas conectadas entre sí. Aunque hay muchas formas de conexionismo, la más extendida es la que presenta la estructura de las redes neuronales.

Santiago Ramón y Cajal [21] podría considerarse como uno de los primeros precursores e investigadores de las ideas conexionistas. Él sentó las bases para el estudio de las redes neuronales y las conexiones entre ellas. Sus investigaciones se remontan a finales del siglo XIX. Ramón y Cajal propuso que las neuronas funcionaban como unidades individuales que se comunicaban entre sí mediante neurotransmisores que atravesaban la hendidura que separaba estos elementos nerviosos, conocida como hendidura sináptica.

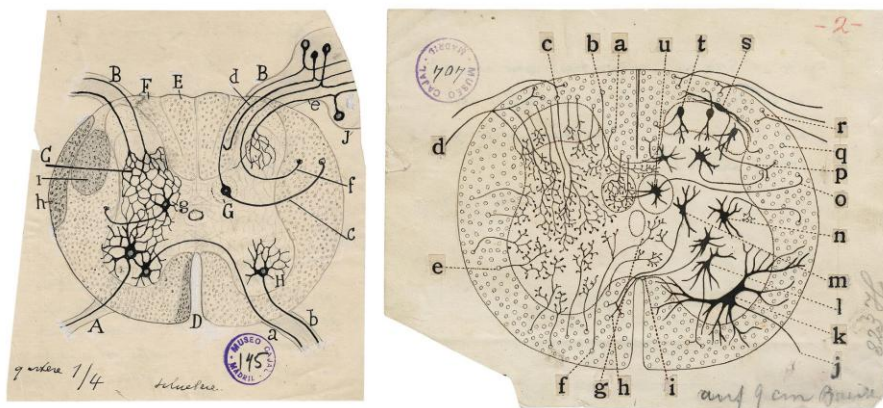


Figura 6. Ilustraciones de Ramón y Cajal de dos teorías contrastantes sobre la composición cerebral: la teoría reticular y la doctrina de la neurona que él propuso

Ya a mediados del siglo XX, *Donald Hebb* [22] propuso su postulado de aprendizaje, donde explicaba que la transmisión de información entre neuronas era más eficaz si estas se activaban simultáneamente múltiples veces. *Hebb* defendía en su obra:

"La idea es antigua, que dos células o sistemas de células que están continuamente activas al mismo tiempo tenderán a convertirse en 'asociadas', de manera que la actividad de una facilitará la de la otra."

"Cuando una célula ayuda en repetidas ocasiones a que otra se dispare, en el axón de la primera célula se desarrollan botones sinápticos (o se agrandan si ya existen) en contacto con el soma de la segunda célula."

Esta presentación de la variación de los pesos de las neuronas en función de su coactivación, es decir, la simultaneidad de la activación de las neuronas involucradas está muy relacionada con nuestra visión actual del aprendizaje automático. A la hora de aprender, un sistema constituido por una red de neuronas tiende a dar más importancia a las neuronas activadas en cadena. Cuantas más neuronas hayan interactuado de manera positiva en el mismo recorrido, más eficiente será la asimilación del conocimiento por parte del sistema.

Minsky y Papert, de los que se ha hablado anteriormente, también presentaban ideas claramente conexionistas en sus estudios y publicaciones [23], [24].

Por otra parte, la corriente simbólica, que también aparece en varios campos de estudio como la inteligencia artificial, está basada en la manipulación de símbolos y reglas lógicas. En lugar de centrarse en el análisis de grandes volúmenes de datos, estos sistemas buscan aprender a partir de ejemplos en los que poder observar los patrones y reglas que los caracterizan.

Uno de los precursores de esta corriente fue *John McCarthy*, considerado como uno de los padres de la inteligencia artificial. Recibió el Premio *Turing* en 1971 por sus aportaciones al campo de la IA. *McCarthy* abordaba en su obra [25] la posibilidad de que la inteligencia artificial incorporara sentido común a su forma de tomar decisiones, que pudiera pensar de manera racional para encontrar una solución factible. La idea de crear su *Advice Taker* era generar un programa que siguiera una serie de pasos lógicos y secuenciales, formulando cada parte del proceso de manera formal, llegando así a una respuesta digna de un humano. La propuesta de un programa cuyo pensamiento lógico estuviera basado en la formalización fue una propuesta innovadora muy relacionada con el pensamiento simbólico.

1. $at(I, desk) \rightarrow can(go(desk, car, walking))$
2. $at(I, car) \rightarrow can(go(home, airport, driving))$
3. $did(go(desk, car, walking)) \rightarrow at(I, car)$
4. $did(go(home, airport, driving)) \rightarrow at(I, airport)$
5. $canachult(at(I, desk), go(desk, car, walking), at(I, car))$
6. $canachult(at(I, car), go(home, airport, driving), at(I, airport))$
7. $canachult(at(I, desk), prog(go(desk, car, walking), go(home, airport, driving))) \rightarrow at(I, airport)$
8. $do(go(desk, car, walking))$

Figura 7. Secuencia de pasos que debería realizar el Advice Taker según John McCarthy

Otros dos grandes investigadores de la materia fueron *Allen Newell* y *Herbert A. Simon*, creadores de *Logic Theorist*, un programa capaz de resolver problemas lógicos imitando el estilo de pensamiento humano. *Newell* y *Simon* también eran fieles defensores del simbolismo y presentaban en su obra [26] la importancia de los símbolos tanto en la informática como en los procesos cognitivos humanos. Por otra parte, aparece el concepto de “búsqueda heurística”, utilizada para defender los procesos empíricos que derivan en el pensamiento lógico que permite resolver un problema. El objetivo, común a esta corriente, es moldear el funcionamiento de la máquina para que su manera de abordar una tarea se asemeje lo máximo posible al estilo humano.

En este ámbito, la inteligencia artificial fue desarrollándose, sobre todo, como sistemas expertos, sistemas capaces de emular el razonamiento humano de un individuo experto. Estos sistemas hallan la solución a un problema aplicando reglas heurísticas, aplicando un razonamiento basado en casos, es decir, apoyándose en la resolución previa de un problema anterior para adaptarla al actual o aplicando reglas bayesianas.

Característica	Conexionismo	Simbolismo
Unidad básica	Neurona	Símbolos y reglas
Procesamiento	Paralelo y distribuido	Secuencial y basado en reglas

Tareas	Reconocimiento de patrones, aprendizaje automático	Planificación, razonamiento lógico y consciente
--------	--	---

Tabla 1. Algunas diferencias claves entre IA conexionista y simbólica

Aunque el conexionismo y el simbolismo pueden percibirse como corrientes independientes [27], la combinación de ambas visiones puede llegar a ofrecer mejores resultados en el rendimiento de la inteligencia artificial.

Paul Smolensky investigó la cohesión de estos dos enfoques lo que lo llevó a recibir el Premio *Rumelhart* en 2005 [28]. *Smolensky* sugirió que nuestro cerebro no se limita a un único proceso, sino que combina los dos enfoques. Por una parte, el cerebro se comporta de una manera simbólica, razonando de manera lógica, y, por otra parte, también realiza funciones más automáticas e intuitivas, como el reconocimiento de patrones y el aprendizaje implícito. La combinación de ambas corrientes nos permite abarcar un mayor abanico de tareas y nos aporta más capacidades, pudiendo así desempeñar tanto labores complejas como automáticas.

2.2.4 Procesamiento automático del lenguaje en letras de canciones populares

Aunque la segmentación automática por versos y su conteo no es algo demasiado estudiado, en general, el procesamiento de lenguaje a través de distintos modelos siempre ha sido objeto de estudio, siendo orientado de diversas maneras. Este apartado está destinado a presentar investigaciones previas relacionadas con el tema de este proyecto, desde el uso de modelos estadísticos para reconocer palabras en la voz cantada hasta la creación de métodos para dividir por sílabas partituras a capela.

Los *LLMs*, utilizados fundamentalmente en el área del procesamiento de lenguaje natural, llevan una corta pero intensa trayectoria en este campo. El primer modelo, *GPT-1 (Generative Pre-Training Transformer 1)*, surgió alrededor de 2018, y no tardó en mejorar, desembocando en otros modelos, cada vez mejores y más conocidos. Hoy en día se puede apreciar cómo los *LLMs* han evolucionado exponencialmente desde su inicio, y estas mejoras continúan avanzando, sin detenerse, a un ritmo vertiginoso.

Aunque parezca una fecha muy lejana a la actual, en 2010 se publicó un artículo que abordaba el reconocimiento de fonemas y sonidos en canciones tanto monofónicas como polifónicas. Se

pretendía idear un sistema que automatizara esta tarea mediante un modelo oculto de *Márkov* (*HMM*), un tipo de modelo estadístico que mantiene una estrecha relación con el procesamiento del lenguaje. Este tipo de modelo se basa en estados no observables a simple vista, estados ocultos deducibles a partir de observaciones visibles. Ya que una de las primeras aplicaciones, a mediados de la década de los 70, de este tipo de modelos fue el reconocimiento del habla, *Annamaria Mesaros* y *Tuomas Virtanen* vieron la oportunidad de continuar las investigaciones sobre el tema, publicando su artículo sobre el reconocimiento automático de fonemas cantados mediante modelos ocultos de *Márkov* [29]. En su proyecto ya se empiezan a ver resquicios de la automatización inteligente de procesos y tareas cognitivas humanas mediante el uso de modelos de lenguaje y la inteligencia artificial. Este tipo de *IAs* fue una de las bases sobre las que se asientan los sistemas de reconocimiento de lenguaje hablado actuales.

2.3 Modelos extensos de lenguaje (*LLMs*) y su uso en el análisis y segmentación de canciones y poemas

En estos últimos años, los modelos de lenguaje han revolucionado el procesamiento de lenguaje natural. Los *LLMs*, entrenados a partir de grandes fuentes de información, cada vez son mejores a la hora de comprender y generar el lenguaje. Sin embargo, estos modelos están mayoritariamente entrenados con textos no poéticos, lo que dificulta enormemente el análisis y manipulación de estructuras poéticas y líricas. Utilizar los *LLMs* para estas tareas no es trivial, y normalmente desembocan en alucinación y respuestas sin sentido. El desarrollo de estas habilidades por parte de los modelos se sigue investigando hoy en día.

Recientemente, hace apenas un par de años, se realizó un estudio en el que se presentaba *ALBERTI*, un modelo de lenguaje capaz de analizar obras poéticas en diversos idiomas [30].

Con el objetivo de alcanzar su rendimiento, se basó en *BERT* (*Bidirectional Encoder Representations from Transformers*), un modelo de lenguaje desarrollado por *Google* con el objetivo de probar la arquitectura *transformer*. Se caracterizaba por tener la capacidad de comprender una palabra dentro de un contexto, teniendo en cuenta la oración en la que se encontraba.

El procesamiento del lenguaje español siguió investigándose, aprovechando modelos como *BERT* para generar *LLMs* de idiomas específicos, avanzando y obteniendo buenos resultados y herramientas muy útiles en el campo del procesamiento del lenguaje natural [31].

2.3.1 Técnicas y metodologías utilizadas para generar y entrenar a los modelos

En este apartado se tratan las distintas técnicas utilizadas para entrenar a los modelos con el objetivo de comparar las respuestas obtenidas por cada uno de ellos.

2.3.1.1 Uso de plantillas de LangChain

LangChain es una biblioteca de *Python* ideada para interactuar con modelos de lenguaje. Esta herramienta permite crear aplicaciones para manejar el flujo de información entre *LLMs* y fuentes externas, como *APIs* y otros servicios.

Las plantillas (*templates*) proporcionadas por *LangChain* facilitan la interacción con estos modelos, permitiendo una interacción mucho más flexible y personalizada. Además, ofrece una *API* intuitiva de cara al usuario, lo que facilita el uso y aplicación de estas funcionalidades. Estas plantillas permiten crear *prompts* personalizados y reutilizables que serían demasiado complejos si no se utilizara la funcionalidad que ofrece *LangChain*.

2.3.1.2 RAG (Retrieval-Augmented Generation - Generación Aumentada por Recuperación)

La Generación Aumentada por Recuperación (*RAG*) es el proceso que permite optimizar la salida de un *LLM* referenciando una base de conocimiento fiable fuera de sus fuentes de datos de entrenamiento antes de generar una respuesta.

Este método permite resolver algunos problemas, como que el *LLM* no pueda proporcionar la información solicitada al no tener esos datos a su alcance. Además, el modelo podrá citar fuentes de confianza proporcionadas por el usuario, lo que añadirá veracidad a las respuestas. Por otra parte, esta técnica también permite crear modelos personalizados, especializados en una tarea, lo que los convertirá en analizadores mucho más eficientes que los *LLMs* genéricos.

2.3.1.3 Fine-tuning - Ajuste fino

El *fine-tuning* es una técnica que permite entrenar a los modelos de lenguaje para que estos desempeñen tareas concretas, especializándose en ellas [38]. El entrenamiento se realiza utilizando un conjunto de datos de ejemplo para, posteriormente, probar el comportamiento del modelo con información que mantiene el mismo formato que la de entrada, pero distinta, para percibir cómo reacciona el *LLM* frente a casos desconocidos pero familiares.

Esta técnica ofrece muchas ventajas. En primer lugar, es un entrenamiento más rápido, lo que permite ahorrar tiempo si se busca trabajar con modelos más grandes. Por otra parte, mediante ciertas configuraciones, puede reducirse el tiempo de entrenamiento e incluso los recursos empleados [37]. Además, si no se tienen demasiados datos, pueden utilizarse modelos preentrenados y ajustarlos con la información pertinente.

2.3.1.4 Reflection - Reflexión

La reflexión es un proceso utilizado en modelos de lenguaje que busca la autoevaluación y el autoanálisis del modelo con el objetivo de mejorar su respuesta [32]. Esto puede aplicarse de distintas maneras, dependiendo del tipo de sistema o modelo a evaluar. Por ejemplo, si se habla de agentes inteligentes, mediante la reflexión se busca que el agente examine sus propios procesos y cambie su comportamiento en consecuencia, tratando de mejorarlo. En cuanto al procesamiento de lenguaje natural, la reflexión presenta la capacidad de autoajuste de la propia comprensión y generación de texto de un modelo.

2.3.2 Tecnologías y herramientas utilizadas para aplicar las metodologías

2.3.2.1 Ollama

Ollama [33] es una herramienta que permite centralizar, crear y utilizar modelos de lenguaje de manera local. Este programa resuelve muchos de los problemas que surgen al intentar trabajar con *LLMs*.

En primer lugar, aunque puede parecer un inconveniente por el espacio necesario, *Ollama* permite trabajar de manera local con los modelos de lenguaje, lo que evita tener que acceder a cualquier *LLM* de manera remota, las restricciones de peticiones y velocidad (*rate limits*) y otros problemas derivados. Además, al funcionar de manera local, siempre se respetará la privacidad del usuario, ya que los datos utilizados se mantendrán en el equipo, sin necesidad de conectarlo a Internet.

Por otra parte, *Ollama* ofrece la posibilidad de crear modelos personalizados a partir de archivos de configuración (*Modelfile*), dando aún más libertad al usuario. Esto permite generar modelos con funciones específicas que responden con información mucho más cercana a lo que se busca.

Ollama fue fundada por *Michael Chiang* y *Jeffrey Morgan* con el objetivo de acercar los mejores modelos de inteligencia artificial a los usuarios, permitiéndoles acceder a estos *LLMs* desde sus propios equipos, de manera local y segura. Actualmente cuenta con un enorme catálogo de diversos modelos, de todos los tamaños y tipos. Algunos de los más utilizados y conocidos son *Llama*, *Gemma*, *DeepSeek* y *Mistral*.

2.3.2.2 LangChain

LangChain es una biblioteca de código abierto que facilita la creación de aplicaciones que utilizan grandes modelos de lenguaje (*LLMs*). Este marco de software permite integrar y gestionar los componentes necesarios para interactuar con este tipo de modelos de manera más sencilla y eficiente. Además, ayuda a, no sólo establecer una conexión con estos modelos en sí, sino también con otras fuentes de datos, *APIs* y herramientas externas.

LangChain permite diseñar aplicaciones que interactúan con los modelos de manera dinámica y flexible. Además, posibilita la personalización del envío de información a los modelos de lenguaje gracias a la encapsulación de llamadas a estos modelos, plantillas de *prompts*, funcionalidades que permiten la conexión con otras aplicaciones que gestionan *LLMs*... Esto ayuda a generar respuestas mejores y más precisas, ya que pueden integrarse diversas fuentes de datos para alimentar a los modelos con información externa, permitiendo, además, el uso de técnicas más avanzadas, como la reflexión o la *RAG*.

2.3.2.3 Unsloth

Unsloth es una herramienta de código abierto que permite entrenar a modelos de lenguaje mediante *fine-tuning*, adaptándolos así para realizar tareas específicas [34]. Aunque puede utilizarse para entrenar cualquier modelo, *Unsloth* ofrece plantillas para algunos modelos ya seleccionados, como *Llama*, *Gemma*, *Mistral* o *DeepSeek*. Además, existe una gran cantidad de tutoriales para guiar al usuario en el proceso de entrenamiento, explicando desde el comienzo todo el procedimiento; esto lo convierte en una herramienta muy amigable e intuitiva frente a otros *frameworks* más complejos.

Por otra parte, *Unsloth* se encarga de optimizar los recursos y aprovecharlos para mejorar el rendimiento del proceso y el entrenamiento, utilizando la tecnología *LoRA* para ajustar únicamente una pequeña parte del modelo en vez de manejar todo el conjunto de parámetros.

Capítulo 3 - Modelo de análisis de segmentación de canciones

En esta sección se presentan los modelos utilizados para probar las distintas implementaciones propuestas, se habla del conjunto de datos que se analizará con los LLMs entrenados y se discutirá la automatización del conteo por sílabas de los textos poéticos y canciones, lo que ha supuesto un problema, y cómo se ha intentado orientar.

3.1 LLMs utilizados en el estudio

Se han utilizado una serie de modelos para aplicar las metodologías y técnicas propuestas.

Modelo	Parámetros	Tamaño
<i>llama2</i>	6.74B	3.8GB
<i>gemma3</i>	4.3B	3.3GB
<i>stablelm2</i>	12B	7.0GB

Tabla 2. LLMs utilizados en el estudio

Para analizar los distintos comportamientos de estos modelos, se han escogido tres tipos diferentes entre sí, dedicados a diversas tareas, con distintos pesos y parámetros.

3.2 Preparación previa del *dataset* a utilizar

EA-DIGIFOLK cuenta con una gran base de datos constituida por información en torno a la música tradicional folklórica de numerosas regiones. Sin embargo, gran parte de esta información está incompleta o no digitalizada.

Al querer analizar tal masa de contenido con distintos fines nace la necesidad de mantener cierta coherencia y consistencia entre toda la información. Muchas de las letras de las canciones almacenadas no tienen sus versos segmentados.

En este caso, para probar las distintas metodologías y técnicas propuestas, contamos con un *dataset* de 339 canciones correctamente segmentadas. Este *dataset* se ha limpiado con el objetivo de trabajar únicamente con la información estrictamente necesaria, facilitando así el procesamiento de los datos. Las tres columnas que se utilizarán en este proyecto son las siguientes: “Nombre”, “Letra” y “Versos”.

Nombre	Letra	Versos
"A la ruru, niño"	"A la ruru, niño, a la ruru ya; duérmete, mi niño, y duérmete ya."	"A la ruru, niño, a la ruru ya; duérmete, mi niño, y duérmete ya."

Tabla 3. Ejemplo de canción almacenada en el *dataset*

En la columna “Nombre” aparecen los nombres de las canciones almacenadas.

En la columna “Letra” aparecen las letras concatenadas de las canciones, sin segmentar, en una sola línea.

En la columna “Versos” aparecen las letras de las canciones con la segmentación correcta.

3.3 Problema derivado de la automatización del conteo de sílabas

Aunque la información contenida en el *dataset* es, en principio, suficiente, se ha añadido una columna extra donde se almacena el conteo de sílabas de los versos segmentados. Esto se incluyó a la hora de escribir el artículo científico [“Spanish folk music lyrics segmentation with large language models and verse metrics”](#), redactado durante el proceso de investigación y aceptado en el congreso “*Digital Humanities 2025*”. El objetivo del estudio era presentar los problemas que generan los *LLMs* al intentar desempeñar tareas de análisis y segmentación de canciones populares. En el trabajo se comparaban los resultados de los modelos y de los algoritmos tradicionales al segmentar versos y al contar las sílabas de estos. Por esa razón resulta conveniente añadir el conteo de sílabas en este proyecto más amplio, con el objetivo de mejorar los resultados obtenidos en el estudio previo.

Sin embargo, implementar un algoritmo para automatizar el conteo de sílabas en versos no es trivial. Partiendo de que la segmentación de versos indicada en el *dataset* es orientativa y basada

más en la estructura poética que en la musical, la distribución de sílabas varía mucho en función de cómo se interprete la canción.

Aunque la implementación podría basarse en la aplicación de reglas básicas sobre las métricas de poesía en castellano, la decisión que determina si una sinalefa existe o no es ambigua y distinta en función del contexto, época, cultura, pueblo, etc. Por ejemplo, al analizar la obra de Francisco de Quevedo, no siempre se cuentan las sinalefas de la misma manera. Según Jacobo Llamas-Martínez [35]:

“Así, dos de los cuatro empleos de ruido son opinables: las sinalefas entre que hicieron y que hace podrían ser entendidas como dialefas por la presencia de la h, cuyo sonido fricativo fue recogido por tratadistas del siglo xvi y xvii. Las sílabas de los versos serían entonces: con.el.rui.do.que.hi.cie.ron y flor.duen.de.que.ha.ce.rui.do.”

La poesía escrita no deja de ser una representación materializada de un sentimiento ambiguo, de un pensamiento. Aunque existen obras poéticas basadas en lo visual, como los caligramas, y muchas obras en verso pueden disfrutarse, al ser contempladas, por su estructura sonora, de rima y métricas, como los sonetos, la poesía es una creación que se degusta mayoritariamente con el corazón, saboreando las sensaciones que transmite. Esto hace que, al ser recitada, varíe notablemente entre interpretaciones, ya que cada persona recita una misma obra con un tono diferente, acentuando las palabras de manera única y añadiendo los reposos a su gusto.

Con las canciones, que no dejan de ser poemas cantados, también ocurre algo similar. Al igual que con la poesía, las letras de las canciones tienden a redactarse por versos. Esto no deja de ser una abstracción para preservar y facilitar la lectura de estas obras artísticas. Sin embargo, al ser cantadas, los versos se convierten en estructuras intangibles e imaginarias que, en el aire, no tienen por qué mantener su estructura original. Además, analizando las canciones populares desde este punto de vista, aquellas que nunca han sido escritas desde el principio, se puede llegar a la conclusión de que nunca fueron poemas en papel, sino poesía cantada, ambigua y cambiante.

Esto hace del conteo de sílabas algo imposible. En las canciones, las letras se cantan siguiendo la música interpretada. Como hay montones de interpretaciones diferentes, la manera de recitar estos versos varía. Por otra parte, para adecuar las palabras a la música, algunas reglas no se cumplen o se modifican, como, por ejemplo, la división de la sinalefa mencionada con anterioridad.

Por ello, la variación de la distribución de las sílabas en función de la interpretación de la obra imposibilita una automatización precisa y genérica de este conteo.

Con el objetivo de obtener un conteo de sílabas de las canciones de referencia para comparar las salidas de los modelos y poder entrenarlos con una base significativa, se intentó automatizar el proceso de conteo para completar el *dataset* utilizando distintas bibliotecas con el fin de encontrar una solución generalizada que funcionara en todos los casos. Sin embargo, como se ha explicado con anterioridad, encontrar la fórmula para solventar este problema no es trivial; de hecho, con los recursos actuales, es imposible.

Al principio se ideó una solución utilizando *Rantanplan*, una biblioteca de *Python* preparada para analizar estructuras poéticas en español. Esta herramienta realiza un escaneo de la obra deseada y proporciona información como la acentuación en las palabras o el uso de sinalefas. Por desgracia, *Rantanplan* no proporciona un método que permita realizar el conteo por versos o estrofas de un texto poético, por lo que acabó descartándose.

Posteriormente se probó a implementar un algoritmo de conteo muy básico con *Re*, otra biblioteca de *Python* que analiza expresiones regulares en el texto proporcionado. Aunque muchos de los versos se contaron como se esperaba, un gran porcentaje de versos no se analizó correctamente, no teniendo en cuenta sinalefas o no respetando las licencias poéticas basada en la acentuación de las palabras finales en los versos. Esta no era una solución generalizada.

Pyverse fue la siguiente opción. Esta biblioteca sí que proporcionaba métodos específicos que permitían contar automáticamente el número de sílabas de un texto poético. En este caso, las respuestas fueron mucho más fieles. Sin embargo, surgían claros fallos que rompían toda la estructura anterior, por lo que tampoco se utilizó como solución.

Finalmente se probó con *Pyphen*, otra herramienta ideada para dividir en sílabas los textos proporcionados. Ya que este método también cometía muchos errores, se acabó descartando por completo como el resto.

Para comprender mejor los problemas derivados de estos algoritmos implementados, se ha ideado la *Tabla 4*, donde se presentan las respuestas de las distintas bibliotecas utilizadas para realizar el conteo de versos de una canción en concreto. La canción a analizar ha sido la siguiente:

"A la rorro, rorro,

y a la rorrórró;
 duérmase, niñoito
 de mi corazón.
 A la rorro, niño,
 y a la rorrórró;
 duérmete, bien mío,
 que ya amaneció."

Analizando esta canción como se analizaría un poema, aplicando las licencias poéticas pertinentes, todos los versos deberían tener seis sílabas, teniendo en cuenta las sinalefas y añadiendo una sílaba extra cada vez que la última palabra del verso es aguda. Sin embargo, en los conteos realizados por las tres bibliotecas se puede apreciar que esto no se cumple. Aunque algunas de las respuestas se acercan a la salida esperada, ninguna respeta completamente la estructura original.

re	pyverse	pyphen
[6, 5, 6, 6, 6, 5, 6, 6, 6, 5, 6, 5]	[6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7]	[6, 5, 6, 6, 5, 6, 6, 6, 6, 5, 6, 6]

Tabla 4. Distintos conteos de versos realizados por cada biblioteca de Python

Ya que automatizar este conteo suponía un problema sin una solución que funcionara para absolutamente todos los casos presentados, se decidió dejar de lado esta propuesta. Para facilitar el proceso y seguir teniendo información a analizar, se realizó el conteo a mano de un pequeño conjunto de canciones para poder tener un *dataset* de referencia a la hora de entrenar los modelos y comparar los resultados obtenidos, como puede verse en la.

Nombre	Letra	Versos	Conteo
"A la rurru, niño"	"A la rurru, niño, a la rurru ya; duérmete, mi niño, y duérmete ya."	"A la rurru, niño, a la rurru ya; duérmete, mi niño, y duérmete ya."	"[6, 6, 6, 6]"

Tabla 5. Ejemplo de formato y contenido final del dataset a utilizar

Capítulo 4 - Segmentación por versos de letras de canciones populares usando *LLMs*

Automatizar la división de las canciones por versos es uno de los principales temas a investigar en este proyecto. Realizar esta segmentación mediante *LLMs* no es algo trivial, ya que estos no están específicamente entrenados para desempeñar tal función.

Por lo tanto, lo que se pretende con este apartado es visualizar cómo varía el comportamiento de una serie de modelos en función de su entrenamiento, del *prompt* utilizado, de las canciones de entrada, etc.

Aunque no existe una solución generalista que segmente bien todas las letras solicitadas, es posible acercarse a alguna implementación que permita realizar una correcta división por versos sobre algunas canciones. Para conseguirlo, se aplicarán distintas metodologías y se analizará la evolución de las respuestas de los distintos modelos utilizados.

4.1 Segmentador usando un *prompt* exhaustivo y sin instrucciones de formato

En esta primera versión se pretende analizar la salida proporcionada por los modelos realizando peticiones con un simple *prompt*. Para ello se han creado modelos específicos personalizados mediante un *Modelfile* con el objetivo de obtener respuestas más específicas sobre la segmentación. Este archivo contiene información relativa al *LLM* que se quiere generar; en este caso, se ha utilizado el siguiente *prompt* con temperatura 1:

Prompt: "You are a song verses analyst. Your answers must always have a JSON format only. You must answer in a JSON format with "Verses" and "Syllables" as keys. Answer using the only two keys given. Use each key only once. The "Verses" key must segment the given song in verses. The answer for the "Verses" key must be the song segmented in verses. In the "Verses" key write only the verses separated by line breaks. The "Syllables" key must give information about how many syllables are in each verse given in the "Verses" key answer. In the "Syllables" key answer with an array with the number of syllables of every segmented verse. Underscores mean syllables that count as one, that is, a synalepha. You will only respond with a JSON object with the keys "Verses" and "Syllables". Do not provide explanations."

Para probar esta versión, las peticiones se realizarán utilizando una canción en concreto:

Canción utilizada: "El gato negro"

Letra concatenada: "Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Segmentación correcta:

"Me dices que era un gato

el que entró por tu ventana.

En la vida he visto yo,

serrano de mi querer,

un gato negro y con sotana."

Conteo de sílabas de referencia: [9, 9, 8, 7, 10]

Como se ha ido mencionando a lo largo del proyecto, el conteo de sílabas en versos de canciones nunca puede percibirse sin música, ya que la distribución de golpes de voz depende mayoritariamente de las melodías interpretadas. Esto hace del análisis una tarea difícil. Sin embargo, para establecer un estándar y poder calcular de manera eficaz el error cometido por el modelo, el conteo de sílabas establecido que va a utilizarse en las comparaciones para esta canción en específico es el siguiente: [9, 9, 8, 7, 10]. Estas métricas se han calculado contando cada sílaba de cada verso sin tener en cuenta las asíntotas ni las reglas que añaden o sustraen sílabas en función de la acentuación de la última palabra. De esta manera se puede convertir una salida de texto en una característica numérica, permitiendo así calcular el error cometido más fácilmente.

Por supuesto, a la hora de analizar el conteo de sílabas realizado por el *LLM*, aunque el error cometido se calcule de la misma manera, se tendrán en cuenta las cifras proporcionadas para determinar si estas tienen o no sentido. Ya que este conteo no es universal y puede realizarse de distintas maneras dependiendo de qué se tenga en cuenta (las melodías de la canción, las reglas poéticas, la adición de sinalefas o sinéresis...), se analizarán estas respuestas de manera particular, ya que alguna puede llegar a tener sentido en su contexto.

Para sacar una muestra de soluciones de cada modelo, se realizarán veinte peticiones con cada *LLM* y se analizarán y discutirán las respuestas obtenidas.

4.1.1 Modelo segmentador específico creado a partir de Llama2

En esta sección se presentarán los resultados obtenidos a partir de la creación del modelo segmentador usando como *LLM* base *Llama2*.

Versos	Sílabas
"Me dices que era un gato El que entró por tu ventana. En la vida he visto yo, Serrano de mi querer, Un gato negro y con sotana."	[14, 9, 8, 5, 7]
"Me dices que era un gato El que entró por tu ventana En la vida he visto yo, Serrano de mi querer, Un gato negro y con sotana"	[10, 7, 5, 6, 8]
"Me dices que era un gato El que entró por tu ventana. En la vida he visto yo, Serrano de mi querer, Un gato negro y con sotana."	[10, 7, 5, 8, 9]
"Me dices que era un gato el que entró por tu ventana. En la vida he vio yo, serrano de mi querer, un gato negro y con sotana."	[14, 10, 7, 5]
"Me dices que era un gato El que entró por tu ventana. En la vida he visto yo, Serrano de mi querer, Un gato negro y con sotana."	[10, 7, 5, 8, 5]

Tabla 6. Cinco respuestas del modelo segmentador de Llama2

Al hacer varias consultas al modelo con la canción indicada se pueden observar infinidad de detalles importantes que pueden analizarse por separado y que le dan más visibilidad al presente problema de la segmentación automática. En primer lugar, aunque muchas de las salidas coincidan en la figura, el modelo ha realizado segmentaciones diversas. En este caso pueden apreciarse tres diferentes. Para visualizar mejor la variación a la hora de segmentar por versos, se ha representado la información obtenida mediante la *Tabla 7*. En ella se puede ver cuál va siendo las distintas segmentaciones devueltas a lo largo de las peticiones realizadas.

Tipo de segmentación	Versos
A	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, Un gato negro y con sotana."
B	"Me dices que era un gato el que entró por tu ventana. En la vida he vio yo, serrano de mi querer, un gato negro y con sotana."
C	"Me dices que era un gato el que entró por tu ventana., En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Tabla 7. Segmentaciones distintas devueltas por el modelo de Llama2

En este caso, tras realizar veinte peticiones, las respuestas del modelo pueden clasificarse en tres tipos en función de cómo segmentó la letra por versos. Cada vez que proporcionaba una salida, el *LLM* usaba una de las tres segmentaciones mencionadas.

Como se puede observar en la *Figura 8*, el primer tipo de respuesta, que se repitió diez veces en las salidas generadas, coincide con la segmentación establecida como correcta. El eje X es el tipo de iteración distinta devuelta por el modelo, mientras que el eje Y es el número de veces que se ha repetido dicha iteración en las respuestas del *LLM*.

La moda, en este caso, es la segmentación correcta. La abundancia de la respuesta correcta en las salidas del modelo no implica que este haya aplicado un proceso racional para determinar cuál

es la segmentación que tiene más sentido. La otra mitad de las salidas son incorrectas y muchas de ellas están distribuidas hacia las últimas iteraciones, lo que quiere decir que cada petición es independiente y no existe un aprendizaje entre ellas.

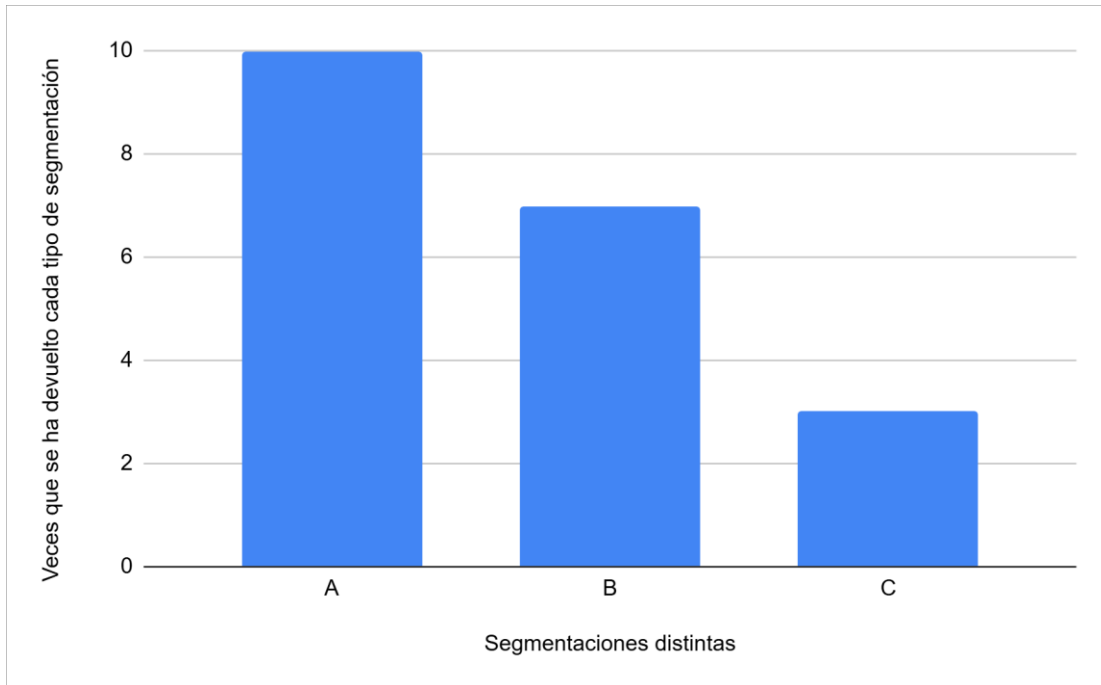


Figura 8. Número de veces que se ha repetido cada segmentación en las respuestas de Llama2

Además de proporcionar variedad en sus respuestas, el modelo, en ocasiones, varía la letra original, eliminando caracteres e incluso palabras enteras. Aunque esto ocurre de vez en cuando, se han agrupado las salidas en función de la segmentación por versos sin tener en cuenta estos fallos por parte del *LLM*.

Como se puede apreciar, los conteos de sílabas no mantienen demasiada coherencia con las segmentaciones proporcionadas por el modelo.

En catorce de las veinte respuestas obtenidas, el número de versos segmentados coincidía con el número de versos contados. Sin embargo, esto no quiere decir que el *LLM* haya contado exitosamente el número de versos en los que ha dividido la letra de la canción. Además, aunque muchas de las salidas no son desorbitadas y mantienen valores no muy grandes que se acercan a los esperados, las reglas que podrían aplicarse a algunos versos para que tengan sentido no se aplican en otros del mismo ejemplo, no manteniendo así la coherencia entre conteos.

Versos	Sílabas
"Me dices que era un gato El que entró por tu ventana. En la vida he visto yo, Serrano de mi querer, Un gato negro y con sotana."	[10, 7, 5, 8, 9]

Tabla 8. Ejemplo de conteo sin sentido por el segmentador de Llama2

Por ejemplo, en este caso, representado en la *Tabla 8*, aunque pueda hallarse la manera de que el conteo de las sílabas por verso segmentado de algún verso coincida con el proporcionado por el modelo, si se aplican esas mismas reglas establecidas en otros versos, el conteo no tiene por qué coincidir. En este caso, para que el último verso, "*Un gato negro y con sotana.*", coincida con el conteo proporcionado por el modelo, debería considerarse la sinalefa "*negro_y*". Sin embargo, es imposible conseguir que "*En la vida he visto yo*" contenga únicamente cinco sílabas.

El conteo de sílabas por verso de toda la canción proporcionado en cada iteración ha variado mucho, siendo mínimo el número de veces que se han dado repeticiones en las respuestas, como puede apreciarse en la *Figura 9*. El modelo ha ofrecido hasta diecisiete conteos distintos entre todas las salidas proporcionadas. En la *Tabla 9* pueden verse los cinco primeros conteos devueltos por el *LLM*, todos distintos entre sí. Muchos de ellos no mantienen ningún tipo de coherencia con la segmentación realizada ni con el resto de conteos presentados.

Tipo de conteo	Conteos de sílabas
A	[10, 8, 7, 5]
B	[10, 8, 5, 6, 7, 8]
C	[14, 9, 8, 5, 7]
D	[10, 7, 5, 6, 8]
E	[10, 7, 5, 8, 9]

Tabla 9. Algunos conteos distintos devueltos por el segmentador de Llama2

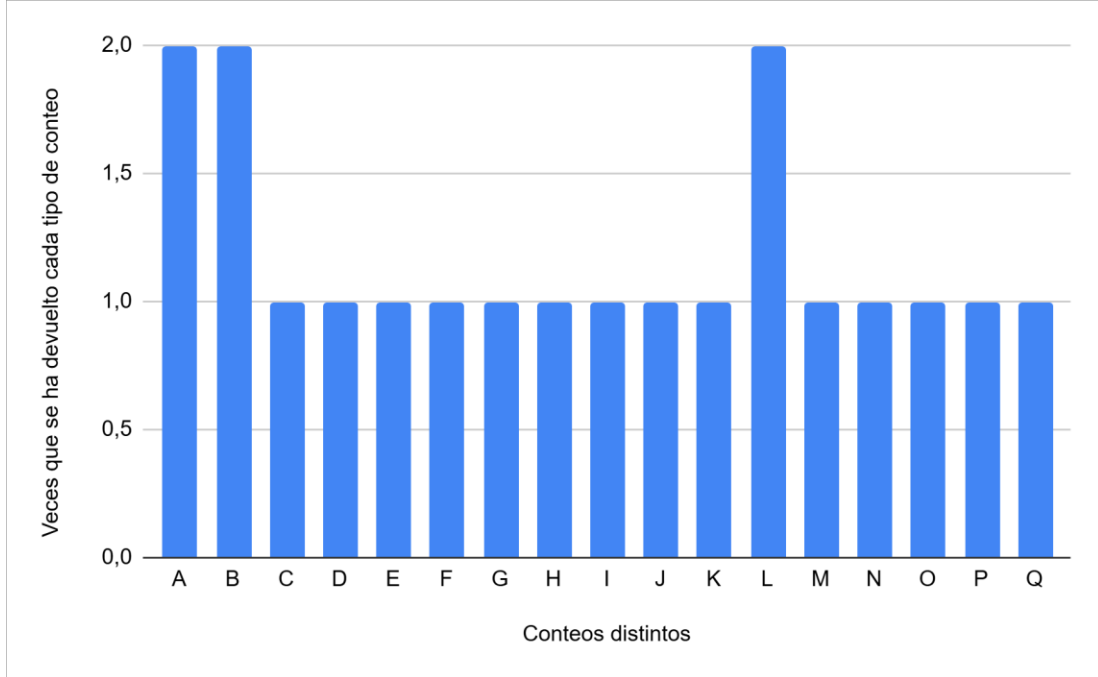


Figura 9. Número de veces que se ha repetido cada conteo en las respuestas de Llama2

En cuanto al formato de salida, el modelo siempre lo proporciona como *JSON*. Sin embargo, la separación por versos varía: en ocasiones divide los versos con “\n”, con saltos de línea, envolviendo cada verso con comillas, etc. Para facilitar el análisis de la segmentación sólo se ha analizado cómo han sido distribuidos los versos en cuanto a contenido.

Este es un ejemplo de formato literal de salida del modelo:

```
{
```

```
  "Verses":
```

```
    "Me dices que era un gato \n el que entró por tu ventana. \nEn la vida he visto yo, \nserrano de mi querer, \ngato negro y con sotana."
```

```
    "Syllables": [10, 8, 7, 5]
```

```
}
```

```
{
```

```
  "Verses":
```

"Me dices que era un gato \nEl que entró por tu ventana. \nEn la vida he visto yo, \nSerrano de mi querer, \nUn gato negro y con sotana."

"Syllables": [10, 8, 5, 6, 7, 8]

}

4.1.2 Modelo segmentador específico creado a partir de Gemma3

En esta sección se presentarán los resultados obtenidos a partir de la creación del modelo segmentador usando como *LLM* base *Gemma3*.

Versos	Sílabas
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, Un gato negro y con sotana."	[10, 8, 7]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[10, 8, 6]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, Un gato negro y con sotana."	[10, 8, 8]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, Un gato negro y con sotana."	[10, 8, 7]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, Un gato negro y con sotana."	[10, 8, 7]

Tabla 10. Cinco respuestas del modelo segmentador de Gemma3

En el caso de *Gemma3*, como puede comprobarse en la *Tabla 10*, las salidas son mucho más regulares. Tanto las segmentaciones como el conteo de versos se mantienen en la misma línea, teniendo una mínima o nula variación. El modelo ha devuelto exactamente la misma segmentación de versos en todas las iteraciones; esta no es la segmentación esperada.

En este caso, el modelo no ha variado la letra. En ocasiones ha cambiado el uso de mayúsculas y minúsculas al comienzo de algunos versos, pero se ha considerado como algo no influyente en el análisis.

Los conteos devueltos mantienen una gran semejanza entre sí, y se han repetido mucho, existiendo, en veinte iteraciones, sólo cuatro tipos de conteo diferente, como puede apreciarse en la *Tabla 11*. Puede comprobarse cómo todos los casos comienzan por diez y casi todos continúan con un ocho, exceptuando uno que contiene un siete. La longitud de los cuatro casos coincide con el número de versos del único tipo de segmentación calculado: tres. Aunque pueda parecer un comportamiento lógico por parte del modelo por las similitudes entre todos los casos, el conteo calculado por el *LLM* queda un poco lejos de los posibles conteos que pueden tener los versos. Además, aunque siempre ha devuelto el mismo tipo de segmentación en todos los casos, el conteo ha variado; lo que, si tuviera sentido, podría considerarse como la aplicación de ciertas reglas de métrica poética. Por ejemplo, ya que el conteo del último verso varía muy ligeramente en los cuatro casos, esto podría interpretarse como la existencia de sinalefas en unas iteraciones y no en otras. Sin embargo, si esto ocurriera de verdad, el conteo de los otros dos versos también debería afectado tras aplicar la regla. Esto demuestra que los conteos realizados no tienen mucho sentido.

La moda de conteos, como puede comprobarse en la *Figura 10*, es el tipo de conteo C, repetido ocho veces a lo largo de todas las iteraciones.

Tipo de conteo	Conteos de sílabas
A	[10, 8, 7]
B	[10, 8, 6]
C	[10, 8, 8]
D	[10, 7, 7]

Tabla 11. Todos los conteos distintos devueltos por el segmentador de Gemma3

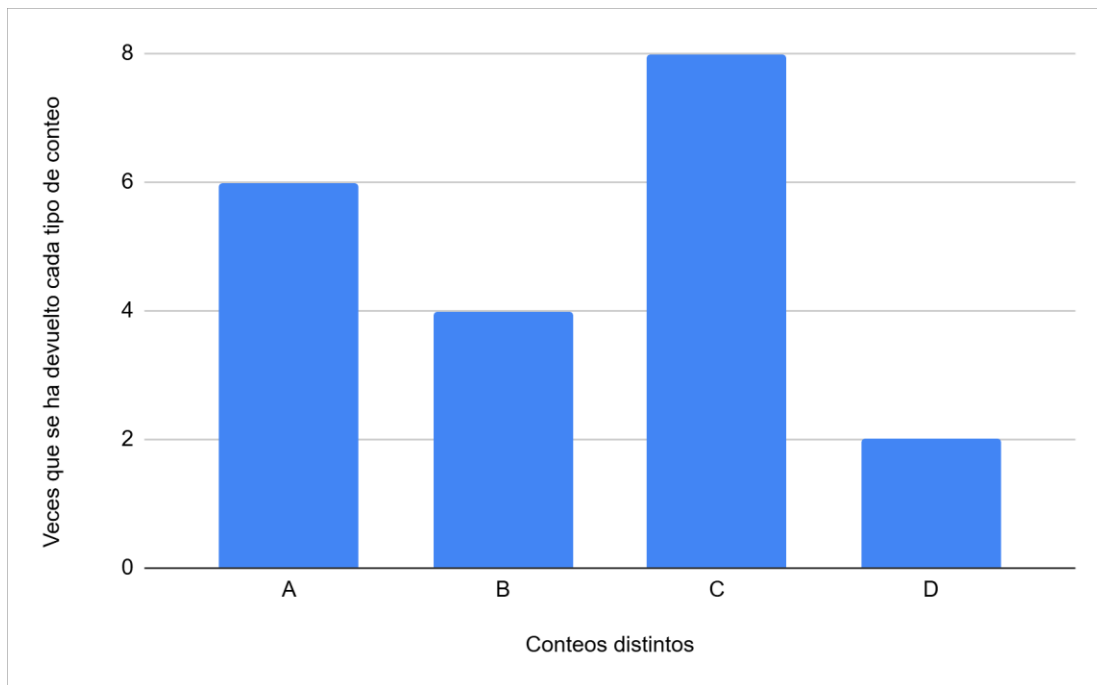


Figura 10. Número de veces que se ha repetido cada conteo en las respuestas de Gemma3

En cuanto al formato de salida, el modelo siempre ha proporcionado su respuesta en formato *JSON*. Sin embargo, la separación de los versos se ha indicado de una manera distinta en cada petición, como ocurrió en el ejemplo anterior.

Este es un ejemplo de formato literal de salida del modelo:

```
{
  "Verses": "Me dices que era un gato el que entró por tu ventana. \nEn la vida he visto yo, serrano
de mi querer, \nUn gato negro y con sotana.",
  "Syllables": [10, 8, 7]
}
{
  "Verses": [
    "Me dices que era un gato el que entró por tu ventana.",
    "En la vida he visto yo, serrano de mi querer, ",

```

"un gato negro y con sotana."

],

"Syllables": [10, 8, 6]

}

4.1.3 Modelo segmentador específico creado a partir de StableLM2

En esta sección se presentarán los resultados obtenidos a partir de la creación del modelo segmentador usando como *LLM* base *StableLM2*.

Versos	Sílabas
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[10, 9, 15]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[10, 23, 17]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[12, 9, 14]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[8, 12, 13, 7, 9, 6]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[10, 9, 12, 14, 7, 8]

Tabla 12. Cinco respuestas del modelo segmentador de StableLM2

Tras realizar las veinte peticiones a este modelo, se pueden observar detalles en común con los ejemplos anteriores y diferencias claras. En primer lugar, han aparecido cuatro formas distintas de segmentar. Por primera vez en esta implementación, el *LLM* ha cortado algunos de los versos,

separándolo entre dos líneas. Esto es especialmente visible en el tercer tipo de segmentación, presente en la *Tabla 13*, aunque también aparece en el segundo. Además, como ocurrió con Llama2, el modelo ha cometido algunos fallos al tratar de mantener la letra original. Este error puede comprobarse en el tercer tipo de segmentación proporcionado por el *LLM*.

Por otra parte, se puede comprobar cómo el modelo ha sido capaz de alcanzar la segmentación correcta, aunque sólo se ha devuelto una vez, como se puede ver en la *Figura 11*.

Tipo de segmentación	Versos
A	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."
B	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."
C	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, an gato negro y con sotana."
D	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Tabla 13. Segmentaciones distintas devueltas por el modelo de StableLM2

En este caso, el tipo de segmentación más usada por el *LLM* ha sido el primero. Esta segmentación fue, además, la única devuelta por *Gemma3*. El resto de los tipos sólo han sido devueltos una sola vez; entre ellos, la segmentación esperada.

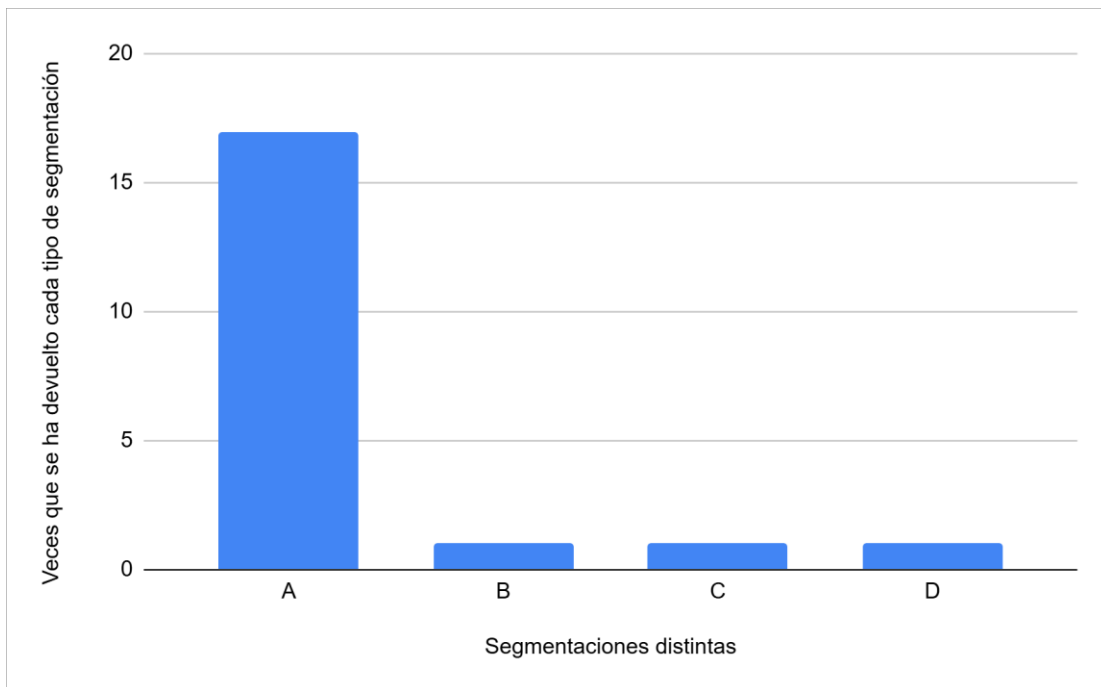


Figura 11. Número de veces que se ha repetido cada segmentación en las respuestas de StableLM2

El modelo ha proporcionado tras cada petición un conteo de sílabas distinto, como se puede comprobar en la *Figura 12*; exceptuando la sexta respuesta, que coincide con la primera. Además, aunque algunas de las respuestas puedan parecerse entre sí, como se puede ver en la primera y cuarta respuesta en la *Tabla 14*, otras no tienen nada que ver. Como puede verse en las tablas de salidas proporcionadas, la tercera petición, a la que se respondió con la segmentación correcta, de seis versos, se adjuntó un conteo de únicamente tres versos.

Tipo de conteo	Conteos de sílabas
A	[10, 9, 15]
B	[10, 23, 17]
C	[12, 9, 14]
D	[8, 12, 13, 7, 9, 6]
E	[10, 9, 12, 14, 7, 8]

Tabla 14. Algunos conteos distintos devueltos por el segmentador de StableLM2

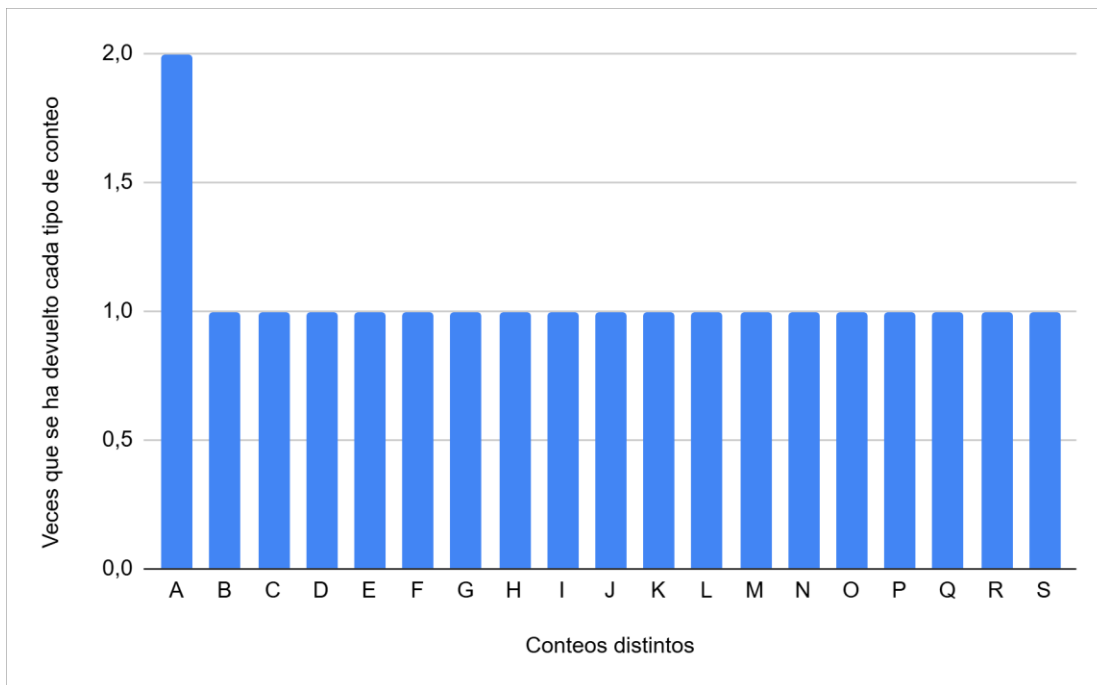


Figura 12. Número de veces que se ha repetido cada conteo en las respuestas de StableLM2

A la hora de proporcionar una respuesta, el modelo no se ha limitado a devolver la información en formato *JSON*, sino que ha acompañado la segmentación y el conteo de versos por una muy breve justificación de su toma de decisiones. Esta es una de sus respuestas:

To fulfill your request, I will segment the given song into verses and count the syllables in each verse. Here's the response in JSON format with "Verses" and "Syllables" keys:

```
``json
{
  "Verses": [
    "Me dices que era un gato el que entró por tu ventana.",
    "En la vida he visto yo, serrano de mi querer,",
    "un gato negro y con sotana."
  ],
  "Syllables": [10, 8, 13]
```

}

...

The song is segmented into three verses: the first verse has 10 syllables, the second verse has 8 syllables, and the third verse contains 13 syllables. The counting assumes normal pronunciation rules; underscores are not considered as they usually aren't counted separately (as a synalepha).

Según su argumento, el modelo es consciente de que, efectivamente, ha segmentado la canción en tres versos y por eso el tamaño de la lista de versos a los que se ha aplicado el cálculo del conteo de sílabas es también tres. Sin embargo, este conteo no tiene sentido ni guarda coherencia con los versos proporcionados.

Este es un ejemplo de formato literal de salida del modelo:

*To fulfill your request, I'll segment the given song into verses and count the syllables in each verse for you. Here's the JSON object with "Verses" and "Syllables" as keys: `***json`*

```
{
```

```
"Verses": [
```

```
],
```

```
"Me dices que era un gato el que entró por tu ventana.",
```

```
"En la vida he visto yo, serrano de mi querer,"
```

```
"un gato negro y con sotana."
```

```
"Syllables": [10, 9, 15]
```

In this JSON object, the verses are separated by line breaks and listed under the "Verses" key. The number of syllables for each verse is provided as an array under the "Syllables" key.

Verse 1: Me dices que era un gato el que entró por tu ventana. (10 syllables)

Verse 2: En la vida he visto yo, serrano de mi querer (9 syllables)

Verse 3: un gato negro y con sotana. (15 syllables)

4.1.4 Conclusiones generales y soluciones propuestas ante los resultados obtenidos

Tras realizar las pruebas indicadas con los modelos, se ha podido comprobar que cada uno tiene sus particularidades y su forma de llegar a una solución. Las características de las respuestas de los *LLMs* se han agrupado en la *Tabla 15* con el objetivo de discutir las semejanzas y diferencias entre las salidas de los modelos evaluados.

Estos resultados se han obtenido tras realizar veinte peticiones a cada modelo. Para poder analizar el comportamiento de los *LLMs* se ha tomado esta medida como estándar.

LLM base	Llama2	Gemma3	StableLM2
Tipos distintos de segmentación	3	1	4
Tipos distintos de conteo	17	4	19
Veces que ha devuelto la segmentación esperada	10 (50%)	0 (0%)	1 (5%)
Veces que ha devuelto un conteo posible	0 (0%)	0 (0%)	0 (0%)

Tabla 15. Características de las salidas de los modelos usando un prompt exhaustivo

Durante el proceso de desarrollo del estudio, se ha enviado y aceptado un artículo (*long paper*) financiado por el proyecto europeo *EA-DIGIFOLK* al congreso científico *Digital Humanities 2025*, celebrado en Lisboa del 14 al 18 de julio de 2025. El artículo se titula “*Spanish folk music lyrics segmentation with large language models and verse metrics*”, y presenta el problema que surge ante la necesidad de automatizar la digitalización y análisis del material que conforma la base de datos de canciones populares del proyecto. Aunque este corpus también contiene material audiovisual, este artículo se centra en el análisis de los textos.

Para arrojar un poco de luz sobre el problema propuesto, se llevó a cabo una investigación que buscaba comparar los resultados obtenidos por algoritmos tradicionales y modelos de lenguaje a la hora de segmentar canciones y hacer un recuento de sus versos. Para tener una referencia certera de las respuestas correctas esperadas, las salidas de estos algoritmos y modelos se compararon con la segmentación y conteo recogidos a mano por dos expertos.

Los resultados permitieron concluir que los *LLMs* tendían a alucinar tras comprobar que el error cometido por estos modelos era mayor que el observado en los algoritmos tradicionales.

Aunque estos algoritmos acertaran más en general, seguían cometiendo fallos. No es posible encontrar una solución definitiva y genérica para todas las canciones, pero investigar más este campo aclaró el problema presente y propuso un medio para acercarse algo más a los resultados esperados.

Este artículo fue presentado al congreso científico mencionado con anterioridad y aceptado para su presentación en julio de este año en dicho congreso.

El artículo completo se incluye al final de esta memoria como [anexo](#).

4.2 Segmentador usando plantillas de LangChain, centrado en la segmentación de versos y no su conteo de sílabas

La solución propuesta para solventar los problemas surgidos con la primera versión de los modelos se basa en el uso de *LangChain* para realizar las peticiones a los modelos.

Para aplicar este método se han utilizado los modelos de segmentador personalizados, creados a partir del *Modelfile* para el segmentador generado usando un prompt exhaustivo y sin instrucciones de formato.

El procedimiento ha sido el siguiente:

- Se ha utilizado un *PromptTemplate* de *LangChain*, una plantilla simple de inserción de texto en un *prompt* para realizar una petición a un modelo de lenguaje, para crear un prompt base que será utilizado con posterioridad, siendo incluido en un *FewShotPromptTemplate*, un *prompt* un poco más completo que incluirá, además, ejemplos de cómo debería ser el output para los inputs utilizados e indicaciones insertadas en la petición al modelo. El *prompt* base sin ejemplos es el siguiente:

"""

Input: {input}

Output: {output}

"""

- Se han tomado dos ejemplos resueltos para que el modelo tenga como referencia cómo debería devolver un *output* en función del *input* recibido. En este caso, el *input* es la

letra concatenada de la canción, sin formato. En esta versión, el objetivo es acercarse más a una correcta segmentación de los versos; el conteo de sílabas por verso se mejorará en versiones posteriores, por lo que el *output* debería ser la letra dividida por versos, estando cada verso separado por un salto de línea (“\n”) para tener un formato genérico en todas las respuestas de todos los modelos. El *prompt* con ejemplos es el siguiente:

```
[
  {
    "input": "El que quiera saber de qué color son las penas siente plaza de soldado y auséntese de su tierra",
    "output": "El que quiera saber\nde qué color son las penas\nsiente plaza de soldado\ny auséntese de su tierra"
  },
  {
    "input": "Ni la fuente más risueña ni el canario más sonoro ni la tórtola en su breña cantarán como yo lloro gotas de sangre por ella",
    "output": "Ni la fuente más risueña\nni el canario más sonoro\nni la tórtola en su breña\ncantarán como yo lloro\ngotas de sangre por ella"
  }
]
```

- Se ha implementado un *FewShotPromptTemplate* donde se incluye la plantilla base creada con anterioridad, los ejemplos resueltos de canciones, un prefijo y sufijo para el *prompt* y las variables de *input* correspondientes. El prefijo añadido al *prompt* es el siguiente:

""You are a lyrics segmenter. Your task is to take unformatted song lyrics and divide them into meaningful verses.

*Each verse should be separated by ****one newline****.*

Keep the original wording — do not invent lyrics.

Here are some examples:""

suffix="""Now segment the following song:

Input:

{input}

Output (verses separated by one newline):""

- Se ha inicializado el *LLM* utilizando la librería que proporciona *LangChain* para interactuar con *Ollama* y sus modelos. En la creación del objeto *OllamaLLM* se indica el nombre del modelo al que enviar la petición (en este caso, la segunda versión del modelo personalizado utilizando de base *Llama2*) y el formato en el que se quiere recibir la respuesta (en este caso, en formato *JSON*).
- Se han unificado las plantillas utilizadas e incluido la canción a analizar en el *input* final. El *input* con la canción a analizar es el siguiente:

Canción utilizada: "El gato negro"

Letra concatenada: "Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Segmentación correcta:

*"Me dices que era un gato
el que entró por tu ventana.
En la vida he visto yo,
serrano de mi querer,
un gato negro y con sotana."*

4.2.1 Modelo segmentador específico creado a partir de Llama2

En esta sección se presentarán los resultados obtenidos a partir de la creación del modelo segmentador usando como *LLM* base *Llama2*.

Versos	Sílabas
"Me dices que era un gato el que entró por tu ventana"	[14, 10]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer un gato negro y con sotana"	[4, 7, 5, 6, 7]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[14, 9, 5, 8, 6, 7]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, sereno de mi querer, gato negro y con sotana."	[14, 8, 5, 7, 9, 6, 7, 5]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[18, 17]

Tabla 16. Cinco respuestas del modelo segmentador de Llama2

Como se puede apreciar en la *Tabla 17*, el modelo segmentador ha devuelto tres tipos de segmentaciones diferentes, entre ellas, la segmentación correcta esperada. El segundo tipo obtenido puede considerarse como un error, ya que el *LLM* sólo ha devuelto los dos primeros versos de la canción.

Tipo de segmentación	Versos
A	"Me dices que era un gato El que entró por tu ventana. En la vida he visto yo,

	Serrano de mi querer, Un gato negro y con sotana."
B	"Me dices que era un gato el que entró por tu ventana"
C	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Tabla 17. Segmentaciones distintas devueltas por el modelo de Llama2

Como se puede apreciar en la *Figura 13*, la segmentación que más se ha repetido, ocupando casi todas las peticiones, ha sido la segmentación correcta. Los otros dos tipos de segmentación se han devuelto una sola vez. Con estos resultados se puede concluir que el rendimiento del modelo mejora si se le ofrecen ejemplos resueltos como referencia.

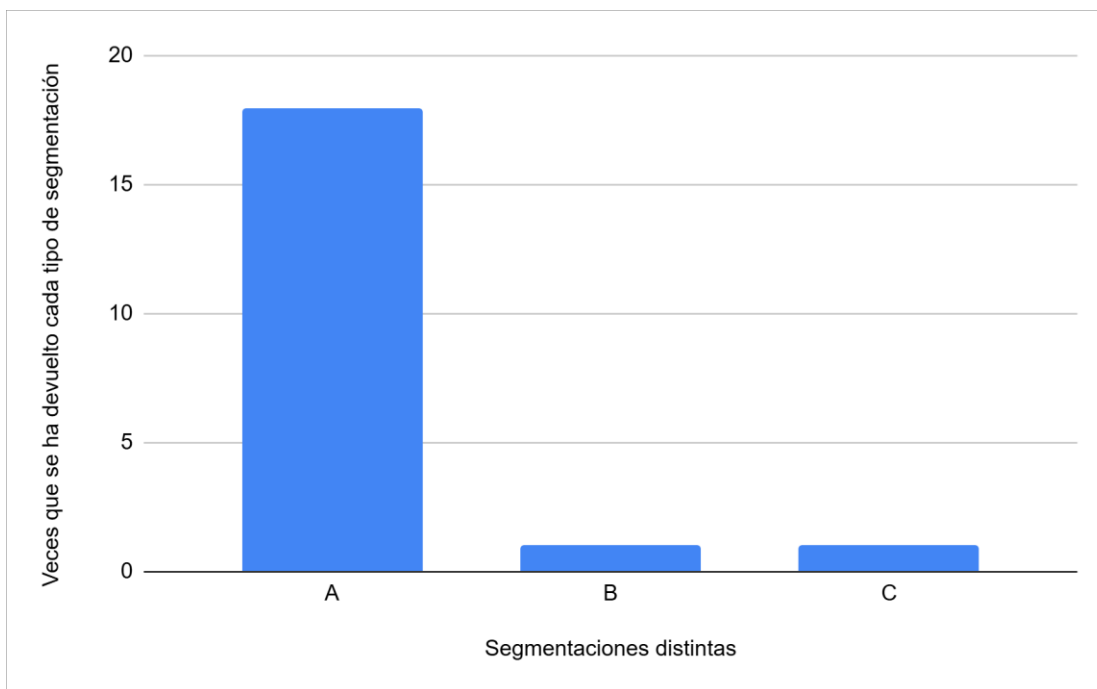


Figura 13. Número de veces que se ha repetido cada segmentación en las respuestas de Llama2

Aunque el conteo de sílabas no se ha añadido en el *prompt* de entrada, se ha analizado para compararlo con posterioridad con otra implementación más centrada en el conteo. Como se puede comprobar en la *Figura 14*, el modelo ha devuelto un conteo de sílabas diferente en cada iteración.

Además, cada respuesta proporcionada difiere mucho del resto, pudiendo considerarse el razonamiento del modelo como algo casi aleatorio.

Tipo de conteo	Conteos de sílabas
A	[10, 9, 8, 7, 6]
B	[10, 8, 6, 5, 4]
C	[8, 6, 9, 7, 10]
D	[7, 5, 4, 8, 6]
E	[7, 8, 5, 7, 8, 6]

Tabla 18. Algunos conteos distintos devueltos por el segmentador de Llama2

En la *Tabla 18* puede comprobarse cómo las cifras devueltas no son tan desorbitadas. Aunque ningún conteo llega a ser completamente aplicable a las segmentaciones devueltas, los valores giran en torno a números de sílabas que podrían encajar con alguna división de versos realizada por el modelo.

Todas las segmentaciones realizadas por el *LLM* están compuestas por cinco versos, exceptuando dos de ellas, de dos y tres versos. Sin embargo, sólo doce de veinte conteos calculados por el modelo están compuestos por cinco valores. Esto demuestra cierta aleatoriedad en las respuestas proporcionadas.

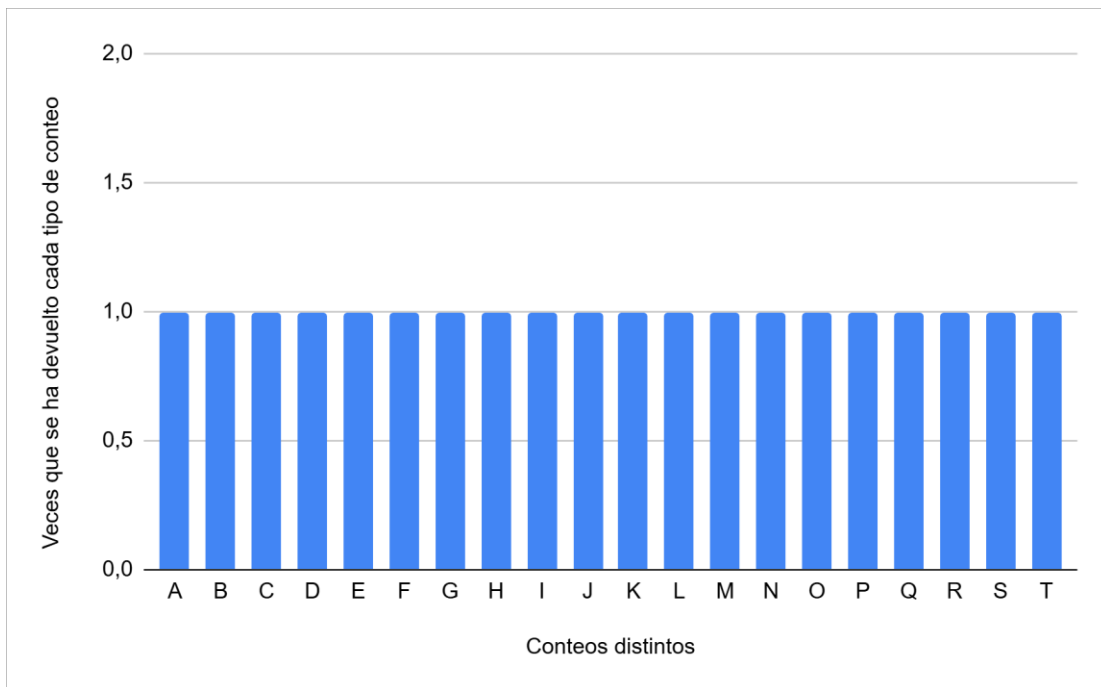


Figura 14. Número de veces que se ha repetido cada conteo en las respuestas de Llama2

El modelo ha proporcionado sus respuestas en formato *JSON*. Sin embargo, ha habido variaciones en la manera de indicar la separación entre versos. Estas han sido dos de las salidas del modelo:

```
{
  "Verses": "\nMe dices que era un gato \nel que entró por tu ventana.\nEn la vida he visto yo,
\nserrano de mi querer,\nun gato negro y con sotana.",
  "Syllables": [7, 8, 5, 7, 8, 6]
}
```

```
{
  "Verses": [
    "Me dices que era un gato",
    "el que entró por tu ventana."
  ]
}
```

"En la vida he visto yo,"

"serrano de mi querer,"

"un gato negro y con sotana."

],

"Syllables": [10, 7, 6, 8, 8]

}

4.2.2 Modelo segmentador específico creado a partir de Gemma3

En esta sección se presentarán los resultados obtenidos a partir de la creación del modelo segmentador usando como *LLM* base *Gemma3*.

Versos	Sílabas
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[4, 4, 4, 4, 4, 4]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[4, 4, 4, 4, 4, 4]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[4, 4, 4, 4, 4, 4]
"Me dices que era un gato el que entró por tu ventana.	[6, 6, 6, 6]

En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[4, 4, 2, 2, 2, 2]

Tabla 19. Segmentaciones distintas devueltas por el modelo de Gemma3

En este caso, tras realizar veinte peticiones, se ha comprobado que el modelo ha respondido con la misma segmentación ante todas. Además, esta segmentación ha resultado ser la salida correcta y esperada, como puede verse en la *Tabla 20*.

Tipo de segmentación	Versos
A	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Tabla 20. Segmentaciones distintas devueltas por el modelo de Gemma3

Aunque el *LLM* haya acertado con todas sus respuestas, las salidas de conteo de sílabas no han tenido tanto sentido. En primer lugar, ha devuelto hasta siete tipos distintos de conteo para la canción dividida en la misma estructura por versos. Además, aunque esta estructura esté dividida en cinco versos, los dos únicos conteos que respetan este esquema son los de tipo C y G, representados en la *Tabla 21*. Estas dos respuestas apenas se repiten, al contrario que la primera, compuesta por seis elementos de la misma longitud. Esta última salida es la que aparece con más frecuencia a lo largo de las iteraciones, como puede verse en la *Tabla 19*.

Tipo de conteo	Conteos de sílabas
A	[4, 4, 4, 4, 4, 4]
B	[4, 4, 4, 4, 4, 4, 4]

C	[6, 6, 4, 4, 4]
D	[4, 4, 2, 3, 2, 3]
E	[6, 6, 6, 6]
F	[4, 4, 2, 2, 2, 2]
G	[4, 4, 2, 2, 2]

Tabla 21. Todos los conteos distintos devueltos por el segmentador de Gemma3

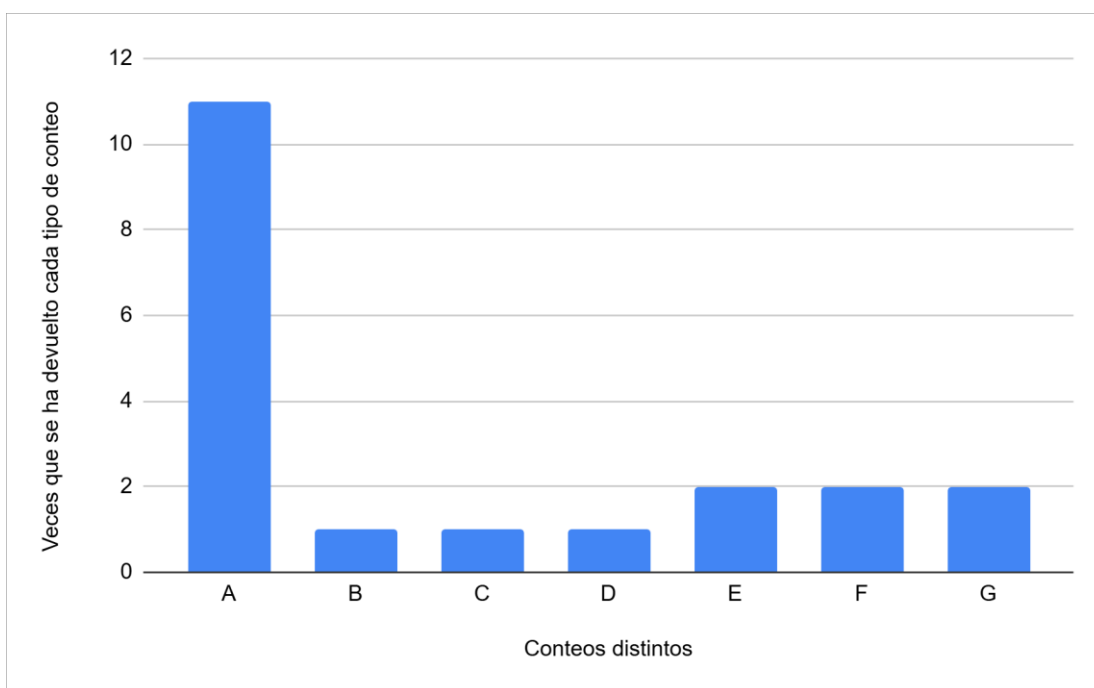


Figura 15. Número de veces que se ha repetido cada conteo en las respuestas de Gemma3

En cuanto al formato de salida, el modelo siempre ha proporcionado su respuesta en formato *JSON*. Además, la separación de los versos se ha indicado de la misma manera en cada petición, divididos por “\n”. Este es un ejemplo del formato de salida de las respuestas del *LLM*:

```
``json
{
  "Verses": "Me dices que era un gato\nel que entró por tu ventana.\nEn la vida he visto yo,\nserrano de mi querer,\nun gato negro y con sotana.",
  "Syllables": [4, 4, 4, 4, 4, 4]
```

```

}
...
``json
{
  "Verses": "Me dices que era un gato\n el que entró por tu ventana.\nEn la vida he visto yo,\n serrano de mi querer,\n un gato negro y con sotana.",
  "Syllables": [4, 4, 2, 2, 2]
}

```

4.2.3 Modelo segmentador específico creado a partir de StableLM2

En esta sección se presentarán los resultados obtenidos a partir de la creación del modelo segmentador usando como *LLM* base *StableLM2*.

Versos	Sílabas
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[9, 8, 14, 13]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo serrano de mi querer, un gato negro y con sotana."	[7, 14, 9, 8, 10]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[14, 9, 13, 7, 8, 6, 10, 12, 9, 5, 7, 8, 4, 13]

"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana"	[7, 9, 13, 12, 6, 8]
"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."	[8, 9, 15, 12]

Tabla 22. Segmentaciones distintas devueltas por el modelo de StableLM2

En este caso, el modelo ha devuelto cuatro segmentaciones diferentes, presentadas en la *Tabla 23*. En algunas de las divisiones los versos han sido fraccionados por la mitad, mientras que en otras se han unido múltiples versos en uno. Sin embargo, el *LLM* también ha devuelto la segmentación correcta, en este caso, de tipo B. Además, esta salida es la que más se ha repetido a lo largo de las iteraciones, apareciendo hasta ocho veces, como puede comprobarse en la *Figura 16*. La segunda segmentación más devuelta es la A, bastante similar a la correcta excepto por la unión de los dos penúltimos versos en una sola línea.

Tipo de segmentación	Versos
A	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."
B	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo serrano de mi querer, un gato negro y con sotana."
C	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo,

	serrano de mi querer, un gato negro y con sotana."
D	"Me dices que era un gato el que entró por tu ventana. En la vida he visto yo, serrano de mi querer, un gato negro y con sotana."

Tabla 23. Segmentaciones distintas devueltas por el modelo de StableLM2

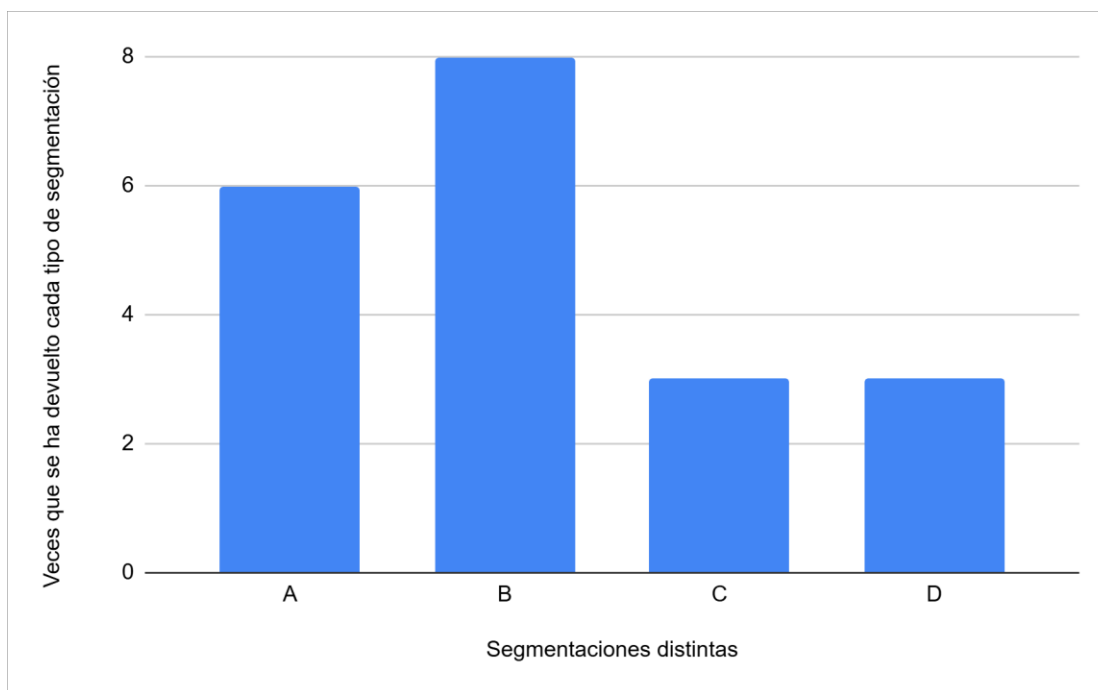


Figura 16. Número de veces que se ha repetido cada segmentación en las respuestas de StableLM2

En cuanto a los conteos calculados por el modelo, en cada iteración se ha devuelto una respuesta distinta, en algunos casos, sin sentido, como el tipo C, representado en la *Tabla 24*. Además, los conteos con cuatro, cinco o seis elementos, aunque podrían parecer algo más cercanos a una solución viable, presentan valores desorbitados que no se ajustan a las segmentaciones proporcionadas, sobre todo las últimas cifras de cada respuesta.

Tipo de conteo	Conteos de sílabas
A	[9, 8, 14, 13]
B	[7, 14, 9, 8, 10]
C	[14, 9, 13, 7, 8, 6, 10, 12, 9, 5, 7, 8, 4, 13]

D	[7, 9, 13, 12, 6, 8]
E	[8, 9, 15, 12]

Tabla 24. Algunos conteos distintos devueltos por el segmentador de StableLM2

En cada salida el modelo ha respondido aportando un conteo de sílabas distinto. Además, como se puede ver en la Figura 17, ante siete de las peticiones, el LLM no ha devuelto un conteo, sino que se ha limitado a calcular la segmentación por versos. Esto tiene sentido, ya que en los ejemplos proporcionados al modelo sólo se tenía en cuenta la división por versos de la canción, no de su conteo por sílabas.

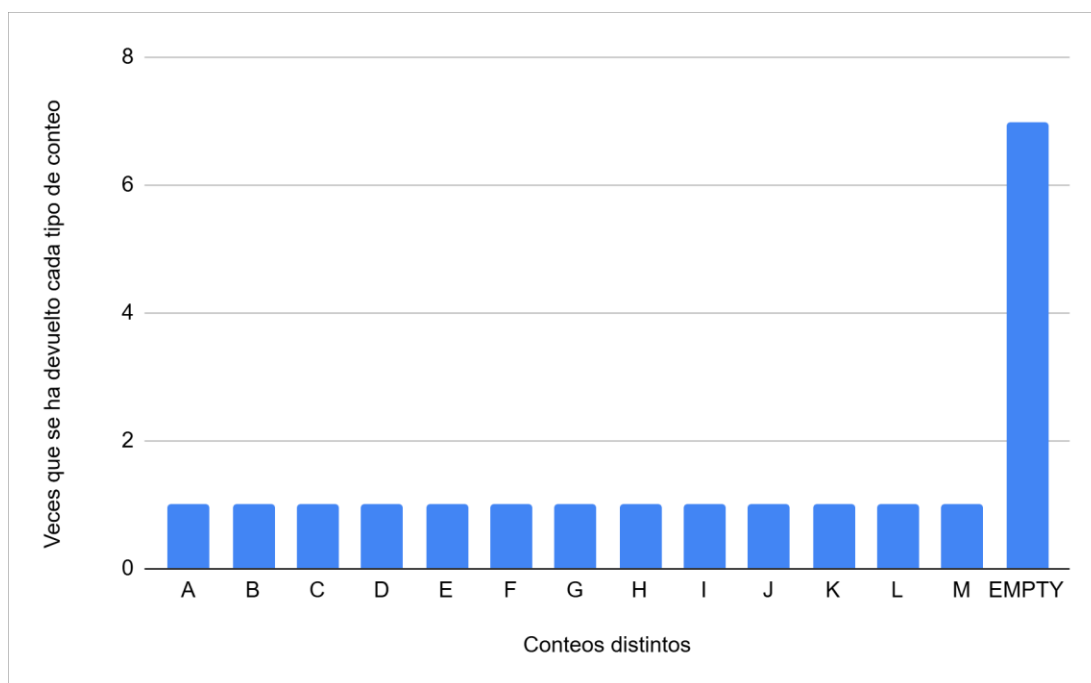


Figura 17. Número de veces que se ha repetido cada conteo en las respuestas de StableLM2

El formato de salida del modelo varía en función de la petición. Algunas salidas sólo están compuestas por la respuesta completa en formato JSON, mientras que otras contienen también la propia justificación del modelo, y algunas no incluyen el conteo de sílabas. Esto último no tiene mucha importancia en este caso ya que el objetivo principal de esta implementación era la mejora de la segmentación. Estas son algunas salidas de ejemplo proporcionadas por el modelo:

```
{
  "Verses": [
```

"Me dices que era un gato el que entró por tu ventana.",

"En la vida he visto yo, serrano de mi querer,"

"un gato negro y con sotana."

],

"Syllables": [16, 20, 24]

}

{"Verses": [

"Me dices que era un gato",

"el que entró por tu ventana.",

"En la vida he visto yo, serrano de mi querer,"

"un gato negro y con sotana."

]}]

{*"Verses": "Me dices que era un gato\nel que entró por tu ventana.\nEn la vida he visto yo,\nserrano de mi querer,\nun gato negro y con sotana."*}

"Syllables": [20, 13, 19, 17, 18, 15, 14]}

Explanation: The input song is segmented into verses with each verse separated by one newline. The "Verses" key contains the formatted lyrics.

To count syllables, we separate each word at underscores (synalepha) and count the number of vowel sounds followed by a consonant sound or punctuation mark:

- Me dices que era: 20 syllables

- un gato el que: 5 syllables

- entró por tu ventana: 13 syllables

- *En la vida he visto yo, serrano de mi querer, un gato negro y con sotana. : 34 syllables*

The "Syllables" key provides the array of syllable counts for each verse segment.

4.2.4 Conclusiones generales y soluciones propuestas ante los resultados obtenidos

Tras realizar las pruebas planificadas con los modelos se han extraído una serie de datos que varían con respecto a la versión anterior de la implementación, mejorando en algunos casos y empeorando en otros. Al estar algo más orientado a la segmentación, los resultados del estudio relacionados con la división por versos de la canción han sido mucho mejores, obteniendo mayoritariamente la respuesta correcta por parte del modelo. Sin embargo, como el conteo de sílabas no se ha tenido tan en cuenta en este apartado, las salidas no han sido muy válidas.

Las características de las respuestas de los *LLMs* se han agrupado en la *Tabla 15* con el objetivo de discutir las semejanzas y diferencias entre las salidas de los modelos evaluados.

Estos resultados se han obtenido tras realizar veinte peticiones a cada modelo. Para poder analizar el comportamiento de los *LLMs* se ha tomado esta medida como estándar.

LLM base	Llama2	Gemma3	StableLM2
Tipos distintos de segmentación	3	1	4
Tipos distintos de conteo	20	7	13
Veces que ha devuelto la segmentación esperada	18 (90%)	20 (100%)	8 (40%)
Veces que ha devuelto un conteo posible	0 (0%)	0 (0%)	0 (0%)

Tabla 25. Características de las salidas de los modelos usando plantillas de LangChain

4.3 Segmentador utilizando *fine-tuning* con *LoRA*, centrado en la segmentación de versos y no su conteo de sílabas

Otra de las soluciones propuestas para mejorar la salida de los modelos es el entrenamiento previo de los *LLMs* con un porcentaje de los datos con los que se trabaja para, posteriormente, probar su funcionamiento con los ejemplos restantes.

Para llevar a cabo este método se ha aplicado *fine-tuning* a los modelos base con los que se está trabajando. Estos modelos no han sido creados a partir de un *prompt* o un *Modelfile*, sino que se les proporcionará el *prompt* justo antes de entrenarlos.

La herramienta utilizada en este caso para entrenar a los modelos es *Unsloth*. Además, en este caso no se busca una salida con formato *JSON*. Se pretende comprobar únicamente la segmentación realizada por el modelo entrenado.

Para entrenar a los modelos, se ha implementado una sencilla aplicación que divide los datos en un grupo de entrenamiento y un grupo de prueba en función del porcentaje proporcional que debe ocupar cada grupo. Aunque una vez divididas todas las canciones en dos *datasets* diferentes para facilitar el proceso de *fine-tuning*, el apartado final de prueba no se incluirá de manera completa en esta sección, sino que se el estudio se realizará con cinco canciones seleccionadas de distinta longitud y forma, para comparar los resultados en función de la obra.

Nombre	Letra	Versos
Las cáscaras de huevo	Que rueden, que rueden, las cáscaras de huevo: las lavanderas hacen así, las planchadoras hacen así, las barrenderas hacen así	Que rueden, que rueden, las cáscaras de huevo: las lavanderas hacen así, las planchadoras hacen así, las barrenderas hacen así
Spanish Petenera (Laura Alexandrine Smith, 1889)	Déjame memoria triste; no me estás atormentando; se la quise o no la quise, niña de mi corazón, no me estás recordando.	Déjame memoria triste; no me estás atormentando; se la quise o no la quise, niña de mi corazón, no me estás recordando.
Repertorio de música española para canto con acompañamiento y guitarra, Núm. 17- Malagueña. Aire popular andaluz (Tomás Damas, 1871)	Y al aura el gentil capullo busca al imán el acero la fuente el que tiene sed y mi corazón al tuyo Y con esto me despido adiós público del alma al que le guste mi canto mil gracias da la gitana,	Y al aura el gentil capullo busca al imán el acero la fuente el que tiene sed y mi corazón al tuyo Y con esto me despido adiós público del alma al que le guste mi canto mil gracias da la gitana
La pájara pinta	Estaba la pájara pinta sentadita en el verde limón, con el pico recoge la hoja, con las alas recoge la flor. ¡Ay, sí! ¿Cuándo la veré yo? ¡Ay, sí! ¿Cuándo la veré yo? Me arrodillo a los pies de mi madre, fiel y constante, dame una mano, dame la otra, dame un besito que sea de tu boca.	Estaba la pájara pinta sentadita en el verde limón, con el pico recoge la hoja, con las alas recoge la flor. ¡Ay, sí! ¿Cuándo la veré yo? ¡Ay, sí! ¿Cuándo la veré yo? Me arrodillo a los pies de mi madre, fiel y constante, dame una mano, dame la otra, dame un besito

		que sea de tu boca.
Arrullo mestizo de México	Duérmete, mi niño con todo y tambache, tu madre la zorra, tu padre el tlacuache. Duérmete, niñita, que ahí viene el viejo, a llevarte viene con todo y pellejo. Duérmete, niñito, que ahí viene el coyote, a llevarte viene y a comerte al monte. Duérmete, mi niño, que estás en cajón; tu madre la zorra, tu padre el tejón. Duérmete, niñito, no venga el caucón, te quite la vida y a mí el corazón.	Duérmete, mi niño con todo y tambache, tu madre la zorra, tu padre el tlacuache. Duérmete, niñita, que ahí viene el viejo, a llevarte viene con todo y pellejo. Duérmete, niñito, que ahí viene el coyote, a llevarte viene y a comerte al monte. Duérmete, mi niño, que estás en cajón; tu madre la zorra, tu padre el tejón. Duérmete, niñito, no venga el caucón, te quite la vida y a mí el corazón.

Tabla 26. Canciones utilizadas para probar los modelos entrenados

4.3.1 Llama2

Se ha entrenado al *LLM Llama2* para crear un modelo segmentador específico siguiendo el procedimiento explicado. A la hora de entrenar al modelo, se ha hecho un seguimiento del error cometido por el modelo en cada iteración. Para resumir el resultado de este seguimiento, se han sintetizado treinta ejemplos en la *Tabla 27* que muestran el proceso y variación del error a lo largo de todo el entrenamiento. Se puede apreciar una clara mejora desde el inicio hasta el final de la ejecución.

Step	Training Loss	Step	Training Loss	Step	Training Loss
1	1.410700	20	1.212300	51	0.883800
2	1.699500	21	0.949500	52	0.882000
3	1.113300	22	1.022000	53	0.708100
4	1.540600	23	1.002600	54	0.696300
5	1.532600	24	1.009000	55	0.829800
6	1.612600	25	0.942100	56	0.697000
7	1.592800	26	1.125200	57	0.893700
8	1.499600	27	1.130200	58	0.768100
9	1.075600	28	0.819600	59	0.628800
10	0.982500	29	0.882200	60	1.051400

Tabla 27. Seguimiento resumido del error de entrenamiento del modelo Llama2

Como se ha indicado al comienzo de la sección, para probar el rendimiento del modelo entrenado se han utilizado las letras de cinco canciones diferentes que varían en longitud y formato.

Se ha realizado únicamente una petición por cada canción para obtener una respuesta tentativa de cada tipo de obra.

Canción utilizada para la cuarta petición: "Las cáscaras de huevo"

La respuesta del modelo a la petición ha sido la siguiente:

"Que rueden, que rueden,

las cáscaras de huevo:

las lavanderas hacen así,

las planchadoras hacen así,

las barrenderas hacen así"

Como se puede observar, la segmentación realizada por parte del modelo se corresponde con la esperada. Puede que, al ser una obra de cortos versos y longitud, el *LLM* no haya tenido mucha dificultad para encontrar el patrón o patrones que definen la estructura de esta canción. Por ello se realizan más peticiones con el resto de canciones para comprobar si este comportamiento esperado se mantiene en el resto de ejemplos o no.

Canción utilizada para la segunda petición: "Spanish Petenera (Laura Alexandrine Smith, 1889)"

La respuesta del modelo a la petición ha sido la siguiente:

"Déjame memoria triste;

no me estás atormentando;

se la quise o no la quise,

niña de mi corazón,

no me estás recordando."

En este caso, la segmentación realizada por el modelo también es correcta. Esto puede deberse a que la longitud de esta canción es igual que la anterior. Además de acertar con el número de versos, estos están divididos correctamente, sin haber saltos separando palabras que deben ir en una misma línea.

Canción utilizada para la tercera petición: "Repertorio de música española para canto con acompañamiento y guitarra, Núm. 17- Malagueña. Aire popular andaluz (Tomás Damas, 1871)"

La respuesta del modelo a la petición ha sido la siguiente:

*"Y al aura el gentil capullo
busca al imán el acero
la fuente el que tiene sed
y mi corazón al tuyo
Y con esto me despido
adiós público del alma
al que le guste mi canto
mil gracias da la gitana,"*

Como en el primer y segundo caso, esta canción también ha sido segmentada correctamente. Esta canción es más larga que las anteriores, lo que podría haber generado problemas en la salida del modelo. Sin embargo, la respuesta obtenida se corresponde con la esperada.

Canción utilizada para la cuarta petición: "La pájara pinta"

La respuesta del modelo a la petición ha sido la siguiente:

*"Estaba la pájara pinta
sentadita en el verde limón,
con el pico recoge la hoja,
con las alas recoge la flor. ¡Ay, sí! ¿Cuándo la veré yo?
¡Ay, sí! ¿Cuándo la veré yo? Me arrodillo a los pies de mi madre,
fiel y constante, dame una mano,
dame la otra, dame un besito que sea de tu boca."*

En este caso, el modelo ha tenido algunos fallos al segmentar la canción, seguramente debido a la longitud de la obra, algo mayor que las anteriores. Sin embargo, estos errores no son demasiado

graves. El *LLM* no ha cortado ningún verso por la mitad. Al equivocarse ha unido dos versos en uno, pero ha mantenido el formato de ambos, aunque los haya escrito juntos, sin romper el verso extra, respetando el final natural del verso.

Canción utilizada para la cuarta petición: "Arrullo mestizo de México"

La respuesta del modelo a la petición ha sido la siguiente:

*"Duérmete, mi niño con todo
y tambache, tu madre la zorra,
tu padre el tlacuache.*

*Duérmete, niñita, que ahí
viene el viejo, a llevarte
viene con todo y pellejo.*

*Duérmete, niñito, que ahí
viene el coyote, a llevarte
viene y a comerte al monte.*

*Duérmete, mi niño, que estás
en cajón; tu madre la zorra,
tu padre el tejón.*

*Duérmete, niñito, no venga
el caucón, te quite la vida
y a mí el corazón."*

En este caso, la segmentación realizada por el modelo es peor. Algunos versos se han cortado por la mitad, no respetando así su estructura original. Mientras que la obra original está formada por veinte versos, esta tiene únicamente quince, habiendo unido versos en uno y añadido palabras a versos a los que no pertenecen. Este ejemplo es significativamente mayor que los anteriores, lo que ha podido desembocar en la mala división por versos por parte del *LLM*.

4.3.2 Conclusiones y soluciones propuestas a partir de los resultados de Llama2

Aunque las respuestas son meramente tentativas, se puede interpretar y predecir el comportamiento que tendrá el modelo ante un mayor número de peticiones.

Ante los resultados obtenidos al probar la segmentación del modelo entrenado con canciones de distinta longitud, estrofa y estructura, se puede observar cómo el rendimiento del *LLM* empeora cuanto más larga es la letra de la obra. El error es nulo cuando se analizan canciones cortas y, a medida que aumenta el tamaño de estas, aumenta: la longitud varía, los versos se cortan por la mitad y algunas palabras acaban en versos distintos, etc.

Aunque no ha sido posible realizar más pruebas con modelos entrenados con *fine-tuning* debido a problemas derivados de la limitación de recursos, los resultados obtenidos indican que probar con otros modelos entrenados de la misma manera podrían resultar en la obtención de una salida prometedora.

Capítulo 5 - Servicio web para facilitar la interacción del usuario con las funcionalidades implementadas

Con el objetivo de permitir que el usuario acceda a todas las funcionalidades implementadas de una manera cómoda e intuitiva, se ha preparado un servicio web desde el cual se pueden realizar peticiones a los modelos si estos se encuentran localmente donde se presente el servicio. En la *Figura 18* aparece la interfaz del servicio web, donde se presentan todas las opciones de interacción con los modelos. Para que las interacciones funcionen correctamente, el usuario debe tener los modelos de manera local y configurar sus nombres como se indica en el *script* que determina la estructura de la página (o viceversa).

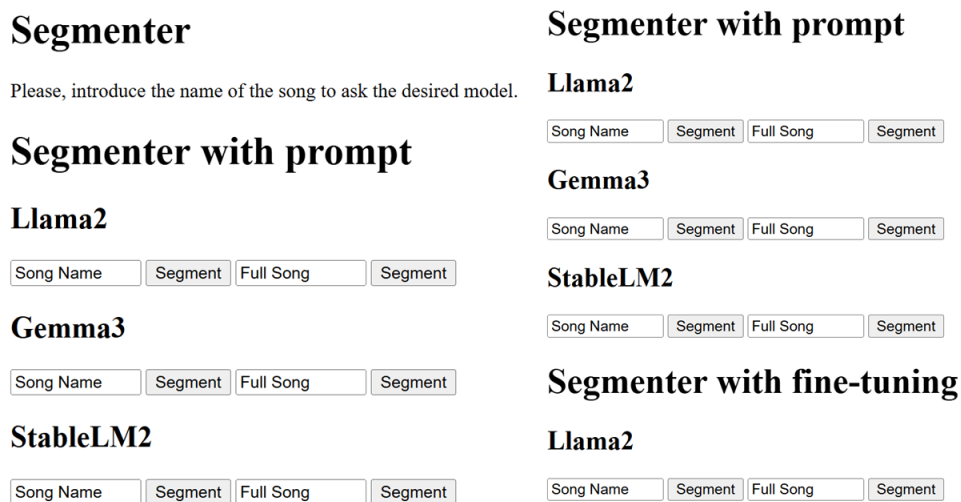


Figura 18. Capturas de la interfaz del servicio web

El usuario puede interactuar con la interfaz para solicitar que los modelos especificados segmenten las canciones deseadas. Para llevar esto a cabo se ofrecen dos posibilidades: introducir el título de la canción si esta aparece en el *dataset* o introducir la letra completa de la canción para que sea analizada sin necesidad de que esta aparezca en los ficheros del proyecto. En la *Figura 19* se muestra un ejemplo de uso en el que, tras escribir el título de una canción perteneciente al *dataset* en la caja de texto, se realiza una petición al modelo, en este caso *Llama2*, para que este segmente la letra de la obra indicada.

Segmenter

Please, introduce the name of the song to ask the desired model.

Segmenter with prompt

Llama2

la calandria Segment Full Song Segment

```
{  
  "Verses": ["En una jaula de oro colgada en un balcón, estaba unacalandria, lloraba su pasión.", "Y a la ru, y a la rurrurrá: duérmeta chiquito y duérmeta ya."],  
  "Syllables": [14, 7]  
}
```

Figura 19. Ejemplo de uso del servicio web solicitando una canción del dataset

En la Figura 20 puede apreciarse el segundo ejemplo de uso, en el que se envía la letra completa de una canción, sin necesidad de que esta esté registrada en el *dataset*, y el modelo recibe la petición para segmentarla por versos.

Segmenter

Please, introduce the name of the song to ask the desired model.

Segmenter with prompt

Llama2

Song Name Segment En una jaula de Segment

```
{  
  "Verses": "  
  En una jaula de  
  oro colgada en un balcón,  
  estaba unacalandria,  
  lloraba su pasión.  
  Y a la ru, y a la rurrurrá:  
  duérmete chiquito y duérmete ya."  
  "Syllables": [10, 8, 5, 7, 6, 5]  
}
```

Figura 20. Ejemplo de uso del servicio web enviando la letra completa de la canción

Capítulo 6 - Discusión y análisis de limitaciones

Como se ha podido comprobar, el análisis y segmentación de canciones populares realizado por modelos de lenguaje no es un procedimiento trivial. Estos modelos no han sido entrenados por lenguaje poético, lo que hace que muchos de estos *LLMs* ni siquiera lleguen a comprender el concepto de “sílabas” en la práctica.

El estudio que se ha realizado para este proyecto se ha basado más en la anchura que en la profundidad. Para realizar las pruebas pertinentes con las soluciones propuestas implementadas y los modelos de lenguaje escogidos, se ha decidido experimentar mayoritariamente con una sola obra, comprobando y comparando así los distintos comportamientos demostrados por los modelos. Por limitaciones de recursos no se ha podido cumplir con todos los objetivos propuestos al principio. Por ello, las pruebas realizadas son tentativas, ideadas para representar las mejoras que podrían demostrarse con más canciones y metodologías aplicadas.

En cuanto a los resultados obtenidos de la segmentación por versos de los distintos modelos, se puede ver que, aunque a simple vista el número de segmentaciones distintas devueltas por los *LLMs* coinciden de una implementación a otra, en la versión mejorada con plantillas de *LangChain* las segmentaciones que no son la esperada tienden a ser errores o simplemente no se repiten tanto como la respuesta correcta. En esta última implementación, la salida que ha predominado ha sido la correcta, mientras que el resto de tipos ha aparecido como mucho una sola vez en el caso de *Llama2* y *Gemma3*. Ya que *StableLM2* está más orientado a ser previamente entrenado, aunque ha mejorado su rendimiento de una implementación a otra, este cambio no ha sido tan significativo como en el resto de modelos: *StableLM2* ha pasado a devolver la segmentación correcta de un 5% a un 40% de las veces, mientras que *Llama2* ha pasado de un 50% a un 90% y *Gemma3* de un 0% a un 100%.

Ya que el estudio ha estado mayoritariamente orientado a la segmentación de las canciones por versos y no su conteo por sílabas, los resultados de estas pruebas no han sido tan favorables como los obtenidos en el ámbito de la estructura dividida de los poemas. La tarea más compleja a realizar en este proyecto por un modelo es el conteo por sílabas, ya que no comprenden este concepto. Como se puede comprobar en las salidas de los *LLMs*, ninguno de estos ha sido capaz de realizar un conteo válido sobre las canciones proporcionadas.

Los datos obtenidos en el entrenamiento mediante *fine-tuning* han demostrado que con un entrenamiento con un corpus algo pequeño puede ayudar a segmentar obras pequeñas y medianas de la manera correcta. El modelo comienza a fallar cuando se realizan peticiones con canciones más largas y de mayor complejidad lírica.

Aunque en este proyecto se propone una solución basada en el uso y entrenamiento de modelos de lenguaje, utilizar algoritmos tradicionales para segmentar textos líricos también puede ser una opción favorable si se tienen en cuenta las variables necesarias.

Capítulo 7 - Conclusiones y trabajo futuro

La automatización de tareas de análisis de textos poéticos con modelos de lenguaje es una labor complicada y sin una solución genérica aparentemente visible. Sin embargo, tras los análisis realizados y la aplicación y combinación de diversas metodologías y técnicas, se observa que es posible acercarse a una solución, en muchas ocasiones fiel a la esperada, que permite segmentar las canciones en versos, como si de poemas se trataran.

Aunque no se han hecho pruebas con varias canciones, los resultados obtenidos tras realizar peticiones con una sola canción y observar las salidas favorables proporcionadas por los modelos, se puede concluir que es posible que los modelos de lenguaje se acerquen a las segmentaciones esperadas con el entrenamiento correcto.

Ante los resultados obtenidos de las segmentaciones realizadas por los modelos, puede concluirse que la implementación con plantillas de *LangChain* ha generado mejores respuestas por parte de los *LLMs*. Además, las mejoras más significativas pueden observarse en los modelos construidos teniendo como base *Llama2* y *Gemma3*. Ya que *StableLM2* es un modelo preparado para ser entrenado previamente, una posible solución a implementar sería la aplicación de la prueba de *fine-tuning* a este modelo y comparar los resultados con los obtenidos en este estudio.

Como el proyecto se ha orientado más a la segmentación por versos, los resultados obtenidos de los conteos por sílabas no han mejorado. En ningún caso se ha devuelto un conteo válido; casi todas las salidas estaban compuestas por cifras desproporcionadas. Además, en la mayoría de casos los modelos han respondido con un conteo distinto ante cada petición, lo que demuestra que estos números se colocaban de manera arbitraria y sin ningún procedimiento con sentido.

Por otra parte, los resultados obtenidos en el entrenamiento con *fine-tuning* indican que, con un buen entrenamiento y datos útiles, los modelos pueden acercarse aún más a una solución robusta y fiel a la original.

Para estudiar soluciones orientadas a la segmentación con modelos de lenguaje y algoritmos tradicionales, se ha trabajado en la redacción del artículo científico "*Spanish folk music lyrics segmentation with large language models and verse metrics*", adjunto al final de esta memoria. En este estudio se comparan los resultados obtenidos por los dos métodos y la cuantificación del error

cometido. Comparando los datos extraídos con las respuestas esperadas, analizadas a mano por profesionales, se concluyó que estos métodos fallaban, sobre todo, los modelos de lenguaje.

Como se ha podido comprobar a lo largo de este proyecto, automatizar el conteo de sílabas, tanto con *LLMs* como con algoritmos tradicionales, sigue siendo un misterio. Nadie tiene la certeza de que una medición de versos sea correcta, aunque esta se muestre como una estructura regular y perfecta. Aunque se tiende a decir que la música es matemática, la interpretación y la emoción no entienden de números.

Ante la imposibilidad de realizar un conteo perfecto de sílabas dentro de una canción, ya que no es algo posible, se propone hacer, además, un análisis musical técnico que permita alcanzar una solución un poco más cercana y real a la pronunciación y división de palabras dentro de una obra artística.

Aunque todos los *LLMs* se han personalizado y moldeado con el objetivo de conseguir respuestas más fieles y con sentido, todas las pruebas se han realizado tomando como base modelos de lenguaje ya existentes. Construir un *LLM* específico dedicado a la segmentación de canciones populares mejoraría infinitamente la calidad de las divisiones por versos. Se propone la implementación desde cero de este modelo para obtener mejores salidas. Esto puede desembocar en la creación de sistemas expertos capaces de segmentar a la perfección y contar correctamente el número de sílabas por verso segmentado.

Este proyecto se ha centrado en el análisis y procesamiento de texto, pero el legado de las canciones populares se transmite de forma oral. Idear un procedimiento para recoger audio y poder manejarlo con ese formato es otro camino a seguir para ampliar el proyecto, proporcionando así aún más herramientas para analizar y difundir la cultura musical y popular.

Introduction

Music is one of the most widespread artistic expressions across the planet. Humanity has been creating and manifesting art from the beginning, evolving and generating new movements, styles, and techniques. This art form has not only changed throughout our history, but each culture and place has adopted their own characteristic styles, with unique traits and inspirations from multiple origins.

After centuries of musical artistic creation, the amount of existing content is immeasurable. This enormous amount of art continues to grow at an exponential rate, as tools for the composition and production of music are increasingly accessible, creating diversity in the universal collection of existing works.

Although nowadays keeping audiovisual content safe is within everyone's reach, the possibility of preserving artistic works for future generations has not always existed. This poses a current problem.

When it comes to collecting and analyzing music, the resources that are necessary to carry out this process are not always available. Folk music is a diverse and personal artistic expression passed down from generation to generation. Each region and culture maintains its own framework, although many of these styles borrow certain traits from other cultures and peoples, which adds richness and diversity to their works. The main challenge that arises when trying to gather information on folk songs is that they are mostly transmitted orally. This means that much of the sung content is lost or partially preserved because it is not written down. Furthermore, although each culture is unique, many songs are passed on from one person to another, leading to variations in the interpretation of these works, even if the foundation remains the same.

The European project *EA-DIGIFOLK* is being developed in this context. Its main objective is to preserve and share European and Ibero-American popular folk culture, particularly in the musical field. *EA-DIGIFOLK* seeks to create a platform that facilitates the dissemination of this culture in educational and public settings. Furthermore, and more closely related to this project, another of the main objectives is the musical and ethnomusicological analysis of the collected audiovisual and written material.

The need to develop this research project stems from an interest in preserving the popular culture that has traveled so long between generations, providing us with information about people, traditions, and places. This involves digitizing artistic content and analyzing it to complete partial or lost information as best as possible. As we find ourselves in an era of countless technological advances, especially in the field of artificial intelligence, which is continually growing exponentially, natural language processing and the automation of analytical tasks find in *LLMs* a key tool to improve and facilitate research in these fields.

Artificial intelligence has never been more important than it is today. The advances made in recent years have allowed for increasingly precise and accessible tools and models to be used for a wide variety of tasks in diverse fields. Its ability to process large volumes of data and identify patterns, among many other functionalities, has allowed for the creation of applications that reshape the way we interact with information and the world around us.

The two main uses of artificial intelligence mentioned in the previous paragraph are precisely two of the skills needed to perform analysis and predictions to preserve and retrieve information about popular culture.

On the other hand, traditional algorithms remain important. With proper study and implementation, results close to those expected can be achieved.

It is possible to approach an appropriate solution using traditional algorithms. However, these results never provide a perfect answer for all the cases analyzed. Each song has a different structure, and, as mentioned above, variations may even arise from the oral transmission of these works. This greatly complicates the process of segmenting lyrics; knowing how to divide them perfectly using a traditional algorithm is not trivial.

Therefore, this project is largely focused on the research and development of specialized *LLMs* capable of analyzing, segmenting, and comparing lyrics of popular songs. Using these models, the results obtained do not always come close to those expected. Furthermore, there are numerous techniques and methodologies that can be used to leverage language models. During this process, research will be conducted to find the optimal implementation that best meets the desired solution.

This research is a small extension of the European *EA-DIGIFOLK* project, whose main objective is to create an open digital platform to collect, analyze, and disseminate folk music from

European and Ibero-American cultures, thus facilitating public access to multimedia resources such as sheet music and recordings, promoting the understanding of folk music from ethnomusicological, musical, and educational perspectives.

Motivation

In this context, the specific problem of analyzing popular song lyrics arises. Automating this task is not trivial, neither using traditional algorithms nor with language models.

To begin with, as in poetry, many rules devised as standard, such as the use of synalepha or syneresis [1], vary depending on the context, the metrics of the other verses, or the type of stanza used in each case. This makes it impossible to generalize that would provide a correct solution in each of the cases analyzed, since the application or non-application of these poetic liberties depends greatly on the context, the period of creation, the author, the structure of the work, regional accents, pronunciation, etc.

Furthermore, popular song lyrics are sung. This means that the verses and syllables do not follow a specific structure determined by grammatical rules, but rather adapt to the melodies and rhythms. Sometimes, each played note may be accompanied by a different, non-repeated syllable, while in other cases, the same syllable may be pronounced repeatedly throughout different rhythmic figures as an artistic device. When writing the lyrics of a song, these variations in the oral transmission of the work are not reflected in the transcription. Traditional algorithms and *LLMs* cannot "hear" the music of these works, or at least not without training or going through a specific process that prepares them to do so.

A specific problem with *LLMs* is that they are generally trained on everyday language, not on poetic or song texts. This type of material has special structures that differ greatly from other types of textual content. Therefore, these models are not fully prepared from the ground up to tackle verse analysis and segmentation tasks and provide correct or valid results in all cases.

This is the motivation behind the project, which aims to analyze various ways to automate the segmentation of popular songs by verse and their syllable counting using traditional algorithms and various techniques applied to *LLMs*. Finding an exact solution is not possible with the current

proposal, but analyzing the results obtained in this study will allow us to address the problem in a different way, with better resources and much greater knowledge.

Goals

General goals

The main goal of this project is to investigate and implement different ways of managing and training *LLMs* to study and compare their behavior and performance. As discussed in the introduction, automating certain analysis tasks such as song segmentation by verse is not trivial, especially when using large language models. This study seeks to implement a series of analysis and segmentation systems for popular songs based on the lyrics of unverified works. The main objective of this proposal is to find the optimal implementation that best meets the expected results.

Given the existence of different techniques, tools, and methodologies, another goal of this project is to compare the results obtained from *LLMs* and traditional algorithms. This will not only allow us to assess which approach is best suited to the segmentation and analysis of song lyrics, but will also allow us to better understand the limitations and capabilities of each method.

Specific goals

- Implement the optimal technique to segment popular song lyrics in the way that most closely matches the original.
- Implement the optimal technique to analyze and compare popular song lyrics with the most accurate results.
- The techniques that will be applied to compare the different results obtained are the following:
 - o Simple prompt: Model a prompt that allows the creation of specific models dedicated to performing the aforementioned tasks.
 - o LangChain templates: Improve the created models using templates prepared to make direct requests to local models specialized in the tasks.

- RAG: Provide the models with additional information related to the request sent by the user so that the *LLM* can base its response on it.
 - Fine-tuning: Train the *LLMs* with the provided database with the goal of obtaining responses that more closely resemble the training information.
 - Reflection: Make the models reflect, thus being aware of their response, giving them room to improve the output and make it more similar to the expected information.
- Compare the results obtained by language models and traditional algorithms.
 - Provide users with an accessible, understandable, and intuitive service that allows them to analyze, segment, and compare the songs in the database.

Work plan

To achieve all the proposed goals, they have been divided into a series of specific tasks, make the research and implementation process much easier:

- Implementation of custom models based on existing models and specific prompts to analyze, segment, and compare song lyrics.
- Use of custom models to analyze, segment, and compare song lyrics with only a simple prompt.
- Use of custom models to analyze, segment, and compare song lyrics using LangChain templates.
- Use of custom models to analyze, segment, and compare song lyrics using *RAG* with song lyrics from the database.
- Use of custom models to analyze, segment, and compare song lyrics using fine-tuning.
- Use of custom models to analyze, segment, and compare song lyrics using reflection.
- Comparison of the results obtained by language models and traditional algorithms.
- Implementation of a web service that integrates all the implemented functionalities to facilitate user access and use.

Task planning

This section shows the Gantt chart in which the different tasks to be carried out throughout the course of the project have been distributed, based on a system of objectives defined by meetings with the tutors and members of the *EA-DIGIFOLK* project:



Figura 21. Planning represented by a Gantt chart

Throughout the year, unforeseen events and events have arisen. This Gantt chart has been updated to reflect the reality and the final organization and completion of tasks within the stipulated timeframe.

During the research process, at least two meetings were held each month with the tutors and, occasionally, with part of the *EA-DIGIFOLK* project team. The objective of these meetings was to monitor the work developed so far, resolve any emerging questions, and contribute further ideas to the project proposal.

In December, we had the opportunity to write a scientific article within the European *EA-DIGIFOLK* project related to the problems presented in this research. The topic and development of this publication can be found in [Apéndice A](#).

Work on this European project has been carried out in collaboration with other institutions and companies such as the University of Salamanca, the University of Coimbra, the National Autonomous University of Mexico, and PTNera Consulting, a company primarily focused on developing projects related to scientific, cultural, social, and educational research and dissemination.

One of the main challenges that arose during development was the need to change the underlying technology used until then due to certain problems.

Furthermore, although at the beginning of the project, implementing RAG as a technique to be applied to the proposed models was proposed, it was ultimately decided not to include it due to the complexity of the process and the lack of results.

Conclusions and future work

Automating poetic text analysis tasks with language models is complicated and has no apparent generic solution. However, after analyzing the data and combining various methodologies and techniques, it is observed that it is possible to approach a solution, often close to the expected one, that allows the songs to be segmented into verses, as if they were poems.

Although tests with multiple songs have not been performed, the results obtained after executing requests with a single song and observing the favorable outputs provided by the models lead us to conclude that the language models can approach the expected segmentations with proper training.

Given the results obtained from the segmentations performed by the models, we can conclude that the implementation with *LangChain* templates has generated better responses from the *LLMs*. Furthermore, the most significant improvements can be observed in the models built using *Llama2* and *Gemma3*. Since *StableLM2* should be pre-trained before being used, a possible solution would be to *fine-tune* this model first and then compare the results with those obtained in this study.

Since the project focused more on verse segmentation, the results obtained from the syllable counts did not improve. No valid count was returned by the models; almost all outputs were composed of numbers that made no sense. Furthermore, in most cases, the models responded with a different count for each request, demonstrating that these numbers were set arbitrarily and without any meaningful procedure.

Furthermore, the results obtained after *fine-tuning* the model indicate that, with good training and useful data, the models can get a closer solution to the original.

To study segmentation solutions using language models and traditional algorithms, the team worked on the scientific article "*Spanish folk music lyrics segmentation with large language models and verse metrics*", attached at the end of this report. This study compares the results obtained by the two methods and quantifies the error rate. Comparing the extracted data with the expected responses, analyzed by hand by professionals, it was concluded that these methods failed, especially the language models.

As has been demonstrated throughout this project, automating syllable counting, both with *LLMs* and traditional algorithms, remains a mystery. No one is certain that a verse measurement is correct, even if it appears to have a regular and perfect structure. Although it is often said that music is mathematics, performance and emotion can't be expressed with numbers.

Given the impossibility of performing a perfect syllable count of a song, a technical musical analysis can be done to achieve a more realistic solution to the pronunciation and word division.

Although all *LLMs* have been customized to achieve more accurate and meaningful responses, all tests were based on existing language models. Building a specific *LLM* dedicated to segmenting popular songs would improve the quality of verse divisions. This model could be made from scratch to obtain better outputs. This could lead to the creation of expert systems capable of perfect segmentation and correctly counting the number of syllables per segmented verse.

This project has focused on text analysis and processing, but the legacy of popular songs is transmitted orally. Developing a method for collecting audio and processing it in that format is another way of expanding the project, providing even more tools for analyzing and sharing musical and popular culture.

BIBLIOGRAFÍA

- [1] A. A. Feltstrom y U. de Nankín, «HERRAMIENTA PARA EL DESARROLLO DE LA EXPRESIÓN ORAL EN E/LE: LÍMITES Y EXCEPCIONES DE LA TENDENCIA ANTIHIÁTICA DEL ESPAÑOL».
- [2] H. Touvron et al., «Llama 2: Open Foundation and Fine-Tuned Chat Models», 19 de julio de 2023, arXiv: arXiv:2307.09288. doi: 10.48550/arXiv.2307.09288.
- [3] G. Team et al., «Gemma 2: Improving Open Language Models at a Practical Size», 2 de octubre de 2024, arXiv: arXiv:2408.00118. doi: 10.48550/arXiv.2408.00118.
- [4] M. Bellagente et al., «Stable LM 2 1.6B Technical Report», 27 de febrero de 2024, arXiv: arXiv:2402.17834. doi: 10.48550/arXiv.2402.17834.
- [5] D. C. José, MÉTRICA ESPAÑOLA. Editorial UNED, 2014.
- [6] A. M. Turing, «Computing Machinery and Intelligence», en Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer, R. Epstein, G. Roberts, y G. Beber, Eds., Dordrecht: Springer Netherlands, 2009, pp. 23-65. doi: 10.1007/978-1-4020-6710-5_3.
- [7] F. Rosenblatt, «The perceptron: A probabilistic model for information storage and organization in the brain.», Psychol. Rev., vol. 65, n.º 6, pp. 386-408, 1958, doi: 10.1037/h0042519.
- [8] M. Minsky y S. A. Papert, Perceptrons: An Introduction to Computational Geometry. The MIT Press, 2017. doi: 10.7551/mitpress/11301.001.0001.
- [9] H. L. Dreyfus, What Computers Can't Do: The Limits of Artificial Intelligence. Harper & Row, 1972.
- [10] D. E. Rumelhart, G. E. Hinton, y R. J. Williams, «Learning representations by back-propagating errors», Nature, vol. 323, n.º 6088, pp. 533-536, 1986, doi: 10.1038/323533a0.
- [11] L. Arnold, S. Rebecchi, S. Chevallier, y H. Paugam-Moisy, «An Introduction to Deep Learning», presentado en The European Symposium on Artificial Neural Networks, abr. 2011. Accedido: 24 de mayo de 2025. [En línea]. Disponible en: <https://www.semanticscholar.org/paper/An-Introduction-to-Deep-Learning-Arnold-Rebecchi/7fc7bb4eec2f27b8f0a0c7fb2a4112c7a7a7abed>
- [12] A. C. Vásquez, J. P. Quispe, y A. M. Huayna, «Procesamiento de lenguaje natural», Rev. Investig. Sist. E Informática, vol. 6, n.º 2, pp. 45-54, 2009.
- [13] J. F. Kolen y S. C. Kremer, «Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies», en A Field Guide to Dynamical Recurrent Networks, IEEE, 2001, pp. 237-243. doi: 10.1109/9780470544037.ch14.
- [14] S. Hochreiter y J. Schmidhuber, «Long Short-Term Memory», Neural Comput., vol. 9, n.º 8, pp. 1735-1780, nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [15] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, y J. Schmidhuber, «A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks», en 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China: IEEE, oct. 2006, pp. 543-548. doi: 10.1109/IROS.2006.282190.

- [16] D. Eck y J. Schmidhuber, «Learning the Long-Term Structure of the Blues», en *Artificial Neural Networks — ICANN 2002*, vol. 2415, J. R. Dorronsoro, Ed., en *Lecture Notes in Computer Science*, vol. 2415. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 284-289. doi: 10.1007/3-540-46084-5_47.
- [17] A. Vaswani et al., «Attention Is All You Need», 2 de agosto de 2023, arXiv: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.
- [18] I. Sutskever, O. Vinyals, y Q. V. Le, «Sequence to Sequence Learning with Neural Networks», 14 de diciembre de 2014, arXiv: arXiv:1409.3215. doi: 10.48550/arXiv.1409.3215.
- [19] K. Cho et al., «Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation», en *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, y W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, oct. 2014, pp. 1724-1734. doi: 10.3115/v1/D14-1179.
- [20] L. Chen et al., «Decision Transformer: Reinforcement Learning via Sequence Modeling», 24 de junio de 2021, arXiv: arXiv:2106.01345. doi: 10.48550/arXiv.2106.01345.
- [21] S. R. y Cajal, *Textura del sistema nervioso del hombre y de los vertebrados: estudios sobre el plan estructural y composición histológica de los centros nerviosos adicionados de consideraciones fisiológicas fundadas en los nuevos descubrimientos*. N. Moya, 1899.
- [22] D.o Hebb, *The Organization Of Behavior*. 1949. Accedido: 13 de mayo de 2025. [En línea]. Disponible en: <http://archive.org/details/in.ernet.dli.2015.226341>
- [23] Marvin Lee Minsky, *The society of mind*. Simon and Schuster, 1986. Accedido: 24 de mayo de 2025. [En línea]. Disponible en: <http://archive.org/details/societyofmind00marv>
- [24] R. McNaughton y S. Papert, *Counter-free automata*. Cambridge, Mass., M.I.T. Press, 1971. Accedido: 24 de mayo de 2025. [En línea]. Disponible en: http://archive.org/details/CounterFre_00_McNa
- [25] J. McCarthy, «Programs with Common Sense», Massachusetts Institute of Technology, USA, Technical Report, nov. 1960.
- [26] A. Newell y H. A. Simon, «Computer science as empirical inquiry: symbols and search», *Commun. ACM*, vol. 19, n.º 3, pp. 113-126, mar. 1976, doi: 10.1145/360018.360022.
- [27] L. Amoroso, M. Bruno, y M. Dominino, «Algunas diferencias entre modelos simbólicos y conexionistas», presentado en XIV Jornadas de Investigación y Tercer Encuentro de Investigadores en Psicología del Mercosur, Facultad de Psicología - Universidad de Buenos Aires, 2007. Accedido: 13 de mayo de 2025. [En línea]. Disponible en: <https://www.academica.org/000-073/88>
- [28] P. Smolensky, «Connectionist AI, symbolic AI, and the brain», *Artif. Intell. Rev.*, vol. 1, n.º 2, pp. 95-109, 1987, doi: 10.1007/BF00130011.
- [29] A. Mesaros y T. Virtanen, «Automatic Recognition of Lyrics in Singing», *EURASIP J. Audio Speech Music Process.*, vol. 2010, n.º 1, Art. n.º 1, dic. 2010, doi: 10.1155/2010/546047.
- [30] J. de la Rosa, Á. P. Pozo, S. Ros, y E. González-Blanco, «ALBERTI, a Multilingual Domain Specific Language Model for Poetry Analysis», 3 de julio de 2023, arXiv: arXiv:2307.01387. doi: 10.48550/arXiv.2307.01387.

- [31] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, y J. Pérez, «Spanish Pre-trained BERT Model and Evaluation Data», 6 de agosto de 2023, arXiv: arXiv:2308.02976. doi: 10.48550/arXiv.2308.02976.
- [32] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, y S. Yao, «Reflexion: Language Agents with Verbal Reinforcement Learning», 10 de octubre de 2023, arXiv: arXiv:2303.11366. doi: 10.48550/arXiv.2303.11366.
- [33] «AI with No Internet Connection». [En línea]. Disponible en: <https://www.codemag.com/Article/2411031/AI-with-No-Internet-Connection>
- [34] A. Shrikhande, «Fine-Tuning LLMs for Domain-Specific Tasks using Unsloth», ADaSci. [En línea]. Disponible en: <https://adasci.org/fine-tuning-llms-for-domain-specific-tasks-using-unsloth/>
- [35] J. Llamas-Martínez, «Teoría y práctica en la poesía de Quevedo: la “compositio” fonética y silábica», La Perinola, vol. 24, pp. 49-69, abr. 2020, doi: 10.15581/017.24.49-69.
- [36] J. Schmidhuber, D. Wierstra, y F. J. Gomez, «Evolino: Hybrid neuroevolution/optimal linear search for sequence prediction», en Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI), 2005. Accedido: 20 de mayo de 2025. [En línea]. Disponible en: <https://archive.air.in.tum.de/Main/Publications/Schmidhuber2005a.pdf>
- [37] E. J. Hu et al., «LoRA: Low-Rank Adaptation of Large Language Models», presentado en International Conference on Learning Representations, oct. 2021. Accedido: 25 de mayo de 2025. [En línea]. Disponible en: <https://openreview.net/forum?id=nZeVKeeFYf9>
- [38] M. Liu, G. Farina, y A. Ozdaglar, «UFT: Unifying Supervised and Reinforcement Fine-Tuning», 22 de mayo de 2025, arXiv: arXiv:2505.16984. doi: 10.48550/arXiv.2505.16984.

APÉNDICES

Apéndice A - Spanish folk music lyrics segmentation with large language models and verse metrics

Durante el desarrollo de esta investigación, desde el proyecto europeo *EA-DIGIFOLK* se organizó, en diciembre, la redacción de un artículo científico con el objetivo de enviarlo al congreso *Digital Humanities 2025*, celebrado en Lisboa en el mes de julio. Este artículo, “*Apéndice A - Spanish folk music lyrics segmentation with large language models and verse metrics*”, fue finalmente aceptado por el congreso y puede encontrarse adjunto al final de esta memoria.

Spanish folk music lyrics segmentation with large language models and verse metrics

María Saez Carrasco¹, Alejandro Romero Hernández¹, Carlos León¹, Lénica Reyes Zúñiga², José Miguel Hernández Jaramillo² and Hugo Gonçalo Oliveira³

¹Dept. of Computer Science and Artificial Intelligence, Universidad Complutense de Madrid, Spain

²PTNera Consulting, Spain

³CISUC/LASI, Dept. Informatics Engineering, University of Coimbra, Portugal

Introduction

The comparative analysis of music, particularly within oral tradition repertoires, presents a significant challenge in the technological realm. Despite numerous efforts, the outcomes have often fallen short of expectations set by (ethno)musicologists. Ethnomusicologist Miguel A. García and others have highlighted risks associated with studies in digital humanities that prioritize technological innovation over qualitative results, a phenomenon known as “technodeterminism”, and other ethical issues (García, 2016; Holzapfel et al., 2018). The core challenge is to align mathematical measures of musical similarity with the culturally and contextually significant aspects valued by (ethno)musicologists, as many algorithms fail to capture these nuances, rendering them ineffective for (ethno)musicological purposes.

In response to these challenges, some propose an innovative approach based on their experience with the semi-automatic comparison of large musical corpora from both oral and academic traditions (Hernández Jaramillo, 2015; L. R. Z. y J. M. H. Jaramillo, s. f.; Zúñiga, 2011, 2015). This approach emphasizes segmenting musical pieces and comparing these segments rather than conducting blind searches across long note sequences in entire pieces. For music from sung oral traditions, it was found that segmenting by verses is particularly effective when songs contain poetic text. This enables the comparison of the music associated with each verse to that of others (J. M. H. Jaramillo & Zúñiga, 2018) and allows for a more precise and contextualized comparison of the music associated with each verse.

However, this presents a challenge. To analyze large musical corpora encoded in formats like XML or MEI, it is crucial to clearly identify where each verse begins and ends. When these files include the sung syllable for each note, the initial challenge is to determine the boundaries of each verse. For this, one has to develop a system that segments not only by considering the melody of the verse but also its metric structure.

Additionally, the existing materials are hard to analyse because they are heterogeneous, and were collected through different systems and during different periods. The automatic digitalization of the materials is lossy. Many involve optical character recognition processes of sometimes hand-written lyrics).

Automatic text analysis has recently seen a revolution with the widespread use of large language models (LLMs). LLMs are relatively recent machine-learning models trained on vast amounts of textual corpora. They have been used in a wide range of language oriented tasks (Törnberg,2023; Rathje, 2024) with outstanding results in some of them. Based on the *Transformer* architecture (Vaswani et al., 2017), these models have outperformed previous approaches in a wide range of tasks. Additionally, their use for text analysis tasks has been democratized with open models. Their utilization is simple because simple prompting and zero-shot learning implies predictions without supervision or annotated data. Yet, while LLMs typically produce human-like output, it is well known that they tend to *hallucinate*, or give outputs that may seem structurally correct, but deviate from a reasonable response.

Still, the application of LLMs for segmenting song lyrics by verses is so far unexplored. In this paper, we report on a comparison of purely algorithmic lyrics segmentation with the result obtained by prompting open source LLMs.

Comparative study of verse segmentation

As previously introduced, this study is focused on analysing the available datasets of Spanish lyrics of folk songs, including songs from Spain and Mexico, acquired from diverse sources: scanned documents with automatic text recognition and digital versions. The corpus includes songs catalogued during the XVII, XIX and XX centuries. The corpus includes musical information, but it was discarded for this study. The available digital resources are not fully curated and have missing information and errors. Given the size of the corpus and particularities of the content, fixing them by hand is both costly and time consuming because, in many cases, there is no clear golden standard. Folk songs are usually transmitted orally and the way in which they are sung does not correspond to the classic verse analysis. Besides, the texts do not conform to normative Spanish, including highly colloquial, hard to automatically parse constructions (“Y hágase p’acá”) and non-existing words (“a la rorrórró”; “larán larán larito”).

In this paper, we report on a two-fold study: a metric-based algorithm and a method based on prompting LLMs. The former is based on searching for possible options within a symmetric division of the verses. Given a text and following the general syllable-counting rules, we explore all possible segmentations (from 5-syllable to 16-syllable verses), discarding those that do not comply with the established rules. For this, we divide text into verses of N syllables and check if they violate any rules. This gives us a list of viable versifications, which we then filter based on the number of rhymes, selecting the one with the highest number. The results have been compared against a golden standard analysis carried out by experts.

We have compared the algorithmic solution against the LLM *Mistral Large* (Jiang et al., 2023). Among the freely available LLMs, the performance of Mistral competes with OpenAI’s models, which are closed (OpenAI, 2024). For this, each song was fed to the model with the prompt in Figure 1.

You are someone who uses segmentation to divide a song into verses. Add no comment about the reflective process. Write only the verses separated by line breaks. Output a JSON and just a JSON. The JSON has only two fields: 'Verses', with the segmented song

as a single string separated by line breaks; and another field called 'Syllables', an array with the number of syllables of every segmented verse. Must be integer numbers, not strings. Underscores mean syllables that count as one, that is, a synalepha. This is the song you have to parse: "{SONG LYRICS}".

Figure 1: Prompt used for verse analysis ({SONG LYRICS} would be replaced by the content of the song on each case).

The analysis included 121 songs. Each song was analysed by the algorithm and through the LLM, outputting the verses and their computed length in syllables. Additionally, the automated lyrics segmentation was compared against a hand-made division carried out by two experts.

Results and analysis

For each song, the analysis yielded a specific verse division. Each of them was compared against a golden standard of hand-made analysis by the experts.

The algorithm yielded an average difference of 1.76 syllables per verse (stdev=2.81, min=0, max=11). That is, on average, the algorithm had 1.76 wrong syllables. It is important to note that the deviation from the correct version is accumulative: once verse N is wrongly identified, most remaining verses do not match the correct version. Thus, the high values do not represent the sample well, as they are produced by linguistic structures that are very hard to recognize.

The LLM version shows an average deviation of 2.99 syllables per verse (stdev=3.00, min=0, max=15). In some cases, the verse is correctly recognized but the counting is wrong, so it could be fixed by hand or by counting the output syllables with other approaches. For the sake of comparability, the original output is provided.

Results show that LLMs have a clear tendency to *hallucinate* (Huang et al., 2024). This was expected, but adjusting the prompt to achieve less incoherent results has not led to reliable results. While not perfect, the algorithm performs better in general.

Spanish poetry metrics include synalephas, but sung lyrics can use it or not, therefore bypassing its application. Some lyrics in the dataset include underscores to indicate a synalepha but others do not. The algorithm considers this information, but the analysis through the LLM does not address it specifically, and given the complexity of the model, it is challenging to quantify the impact. In any case, the used dataset currently includes 258 underscores out of 2814 words (less than 1 per 10 words), so it is assumed that the actual qualitative impact does not greatly impact the conclusions.

Conclusions

Comparative study of folk music lyrics is a challenging task. Particularly, analysing available corpora implies a number of challenges, because the data is imperfect. We have reported on a comparative automatic study of Spanish folk songs lyrics, using LLM versus an algorithm for computing the metrics. Acquired data reveals the inherent limitations of automatic lyrics segmentation using LLMs, and the way in which data is stored makes it impossible to

achieve reliable performance with classic metric counting methods. Results lead us to conclude that straightforward methods are not sufficient to analyse existing databases, and hybrid approaches, combining different techniques and possibly human-based curation are needed.

The pace of evolution of LLMs makes it difficult to predict robust improvements for specific tasks, including verse identification. However, it is not possible to discard qualitatively better results with more developed models, in particular with fine-tuning methods.

Acknowledgements

This work is funded by the “EA-DIGIFOLK: An European and Ibero-American approach for the digital collection, analysis and dissemination of folk music” project (grant agreement no. 101086338), funded by the European Commission.

Bibliography

- Rathje, S., Mirea, D. M., Sucholutsky, I., Marjeh, R., Robertson, C. E., & Van Bavel, J. J. (2024). GPT is an effective tool for multilingual psychological text analysis. *Proceedings of the National Academy of Sciences*, 121(34), e2308950121.
- Törnberg, P. (2023). How to use llms for text analysis. arXiv preprint arXiv:2307.13106.
- García, M. A. (2016). Some Questions Related to Musicology and its Methods. *El oído pensante* 4 (1). https://www.academia.edu/23874308/Some_Questions_Related_to_Musicology_and_its_Methods
- Hernández Jaramillo, J. M. (2015). *Automatización computacional del análisis paradigmático musical. Su aplicación a la música del Flamenco* [Http://purl.org/dc/dcmitype/Text, Universidad de Sevilla]. <https://dialnet.unirioja.es/servlet/tesis?codigo=145667>
- Holzappel, A., Sturm, B., & Coeckelbergh, M. (2018). Ethical Dimensions of Music Information Retrieval Technology. *Transactions of the International Society for Music Information Retrieval*, 1(1), 44-55.
- Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2024). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* <https://doi.org/10.1145/3703155>
- Jaramillo, J. M. H., & Zúñiga, L. R. (2018). El soncito del churripampí. Un acercamiento a las prácticas musicales de las clases subalternas del periodo colonial. *Músicas coloniales a debate. Procesos de intercambios euroamericanos*. https://www.academia.edu/39767848/El_soncito_del_churripampí_Un_acercamiento_a_las_pr%C3%A1cticas_musicales_de_las_clases_subalternas_del_perodo_colonial
- Jaramillo, L. R. Z. y J. M. H. (s. f.). *De jarabes, puntos, zapateos y guajiras. Un sistema musical de transformaciones (Siglos XVIII-XXI)*. Lénica Reyes Zúñiga y José Miguel Hernández Jaramillo. Recuperado 6 de diciembre de 2024, de <https://www.gerinel.org/index.php/es/publicaciones/tesis/de-jarabes-puntos-zapateos-y-guajiras-un-sistema-musical-de-transformaciones-siglos-xviii-xxi>
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). *Mistral 7B* (No. arXiv:2310.06825). arXiv. <https://doi.org/10.48550/arXiv.2310.06825>
- OpenAI. (2024). *ChatGPT* [Software]. <https://chat.openai.com/chat>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. ukasz, & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing*

Systems, 30.

https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Zúñiga, L. R. (2011). *La petenera en México. Hacia un sistema de transformaciones (Tesis de maestría)*.

https://www.academia.edu/14464918/La_petenera_en_M%C3%A9xico_Hacia_un_sistema_de_transformaciones_Tesis_de_maestr%C3%ADa

Zúñiga, L. R. (2015). *Las malagueñas del siglo XIX en España y México. Historia y sistema musical (Tesis doctoral, etnomusicología)*.

https://www.academia.edu/17690517/Las_malague%C3%B1as_del_siglo_XIX_en_Espa%C3%B1a_y_M%C3%A9xico_Historia_y_sistema_musical_Tesis_doctoral_etnomusicolog%C3%ADa

—