



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

— **TELECOM** ESCUELA  
TÉCNICA **VLC** SUPERIOR  
DE INGENIERÍA DE  
TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de  
Telecomunicación

BERTlangua: análisis descriptivo de la obra de Luis García  
Berlanga con Inteligencia Artificial y generación de nuevos  
guiones con modelos LLM

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Ochoa Jiménez, Carlos

Tutor/a: Igual García, Jorge

CURSO ACADÉMICO: 2023/2024



## Resumen

El objetivo de este proyecto es entrenar un modelo de generación de texto (LLM) para generar guiones con el estilo del director de cine español Luis García Berlanga. De esta forma podremos crear nuevos guiones con su estilo e incluso utilizar y acabar las ideas del propio director que no se pudieron llevar a cabo. Paralelamente se realizará un análisis descriptivo de su obra mediante el uso de distintas herramientas y modelos de Inteligencia Artificial, analizando y confirmando los elementos característicos de sus películas. Será mediante la comparación de los resultados obtenidos durante el análisis de su obra junto con los guiones generados por el modelo LLM entrenado que confirmaremos la posibilidad de generar nuevos guiones con el estilo de Luis García Berlanga mediante el uso de Inteligencia Artificial. Este enfoque fusiona la creatividad del director con la capacidad de análisis de la IA, ofreciendo nuevas perspectivas sobre el trabajo del director y explorando aplicaciones innovadoras de esta tecnología en el ámbito cinematográfico.

## Resum

L'objectiu d'aquest projecte és entrenar un model de generació de text (LLM) per generar guions amb l'estil del director de cinema espanyol Luis García Berlanga. D'aquesta manera podem crear nous guions amb el seu estil i fins i tot utilitzar i acabar les idees del propi director que no es van poder dur a terme. Paral·lelament es realitzarà una anàlisi descriptiva de la seva obra mitjançant l'ús de diferents eines i models d'Intel·ligència Artificial, analitzant i confirmant els elements característics de les seves pel·lícules. Serà mitjançant la comparació dels resultats obtinguts durant l'anàlisi de la seva obra juntament amb els guions generats pel model LLM entrenat que confirmarem la possibilitat de generar nous guions amb l'estil de Luis García Berlanga mitjançant l'ús d'Intel·ligència Artificial. Aquest enfocament fusiona la creativitat del director amb la capacitat d'anàlisi de la IA, oferint noves perspectives sobre el treball del director i explorant aplicacions innovadores d'aquesta tecnologia en l'àmbit cinematogràfic.

## Abstract

The objective of this project is to train a large language model (LLM) to generate scripts in the style of Spanish film director Luis García Berlanga. This will allow us to create new scripts in his style and even utilize and complete the director's own ideas that could not be realized. In parallel, a descriptive analysis of his work will be conducted using various AI tools and models, analyzing and confirming the characteristic elements of his films. By comparing the results obtained from the analysis of his work with the scripts generated by the trained LLM, we will confirm the possibility of generating new scripts in the style of Luis García Berlanga using artificial intelligence. This approach merges the director's creativity with the analytical capability of AI, offering new perspectives on the director's work and exploring innovative applications of this technology in the cinematic field.

## RESUMEN EJECUTIVO

La memoria del TFG del GTDM debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la TDM

CONCEPT (ABET)	CONCEPTO (traducción)	¿Cumple? (S/N)	¿Dónde? (páginas)
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1-2
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	1, 45, 55
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	1-2
2. FORMULATE:	2. FORMULAR:	S	1-2
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	3-22, 23-26, 45-53
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	55-70
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	66-72
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	71-72

## Agradecimientos

A mi tutor, Jorge, por acompañarme y guiarme durante todo este proyecto.

A mi familia y amigos, por acompañarme y guiarme durante toda la vida.

# Índice general

## I Memoria

<b>1</b>	<b>Introducción y objetivos</b>	<b>1</b>
1.1	Objetivos . . . . .	1
1.2	Estructura del proyecto . . . . .	2
<b>2</b>	<b>Análisis de la obra de Luis G. Berlanga con IA</b>	<b>3</b>
2.1	Introducción . . . . .	3
2.2	Lo Berlanguiano . . . . .	5
2.3	Análisis de lo ‘berlanguiano’ con Inteligencia Artificial . . . . .	6
2.3.1	Planos secuencia . . . . .	6
2.3.2	Coralidad . . . . .	10
2.3.3	Encuadre . . . . .	14
2.3.4	Vocabulario . . . . .	16
2.3.5	Otras características . . . . .	19
2.3.5.1	Uso de las ubicaciones . . . . .	19
2.3.5.2	Austro-húngaro . . . . .	20
2.3.5.3	Evolución de conceptos . . . . .	20
2.3.5.4	Objetivos fotográficos . . . . .	22
<b>3</b>	<b>Datos: metodología y preprocesado</b>	<b>23</b>
3.1	Digitalización de los guiones . . . . .	23
3.1.1	Extracción de guiones en PDF . . . . .	25
3.1.2	Extracción de guiones con Audio . . . . .	25
3.2	Preprocesado . . . . .	26
<b>4</b>	<b>Introducción al Natural Language Processing (NLP)</b>	<b>29</b>
4.1	Tokenización con BPE . . . . .	29
4.2	ANN . . . . .	30
4.2.1	Neuronas artificiales . . . . .	30
4.2.2	Feedforward Neuronal Networks (FNN) . . . . .	31
4.2.3	Entrenamiento . . . . .	32
4.2.4	RNNs y LSTMs . . . . .	34
4.2.4.1	RNNs . . . . .	34
4.2.4.2	LSTMs . . . . .	35
<b>5</b>	<b>Transformers y LLMs</b>	<b>37</b>
5.1	Arquitectura de los Transformers . . . . .	37

5.1.1	Input	38
5.1.2	Encoder	39
5.1.3	Decoder	41
5.1.4	Output	41
5.2	Estrategias de Muestreo	42
5.3	LLM	42
5.4	Fine-tuning	43
<b>6</b>	<b>Entrenamiento del modelo</b>	<b>45</b>
6.1	Mistral 7B	45
6.1.1	Preparación de la DB	46
6.1.2	Cargar modelo base	47
6.1.3	Tokenización	47
6.1.4	Elección de la longitud máxima	48
6.1.5	LoRA	48
6.1.6	Entrenamiento	49
6.1.7	Testing	50
6.2	LlaMa 3	51
6.2.1	Preparación de la DB	51
6.2.2	Cuantificación	52
6.2.3	Entrenamiento	52
6.3	ChatGPT	53
6.3.1	Entrenamiento	53
<b>7</b>	<b>Resultados</b>	<b>55</b>
7.1	Evolución del proyecto y resolución de problemas	55
7.1.1	Introducción	55
7.1.2	Primeros intentos	55
7.1.3	Introducción de contexto	57
7.2	Análisis de los resultados de Mistral 7B	60
7.3	Análisis de los resultados de Llama3 8B	62
7.4	Análisis de los resultados de ChatGPT 3.5	64
7.5	‘El Autocar’: Análisis y comparación de la obra de BERTlanga	66
7.5.1	Coralidad	66
7.5.2	Encuadre	67
7.5.3	Vocabulario	67
7.5.4	Análisis de conceptos	68
7.5.5	Análisis de ubicaciones	69
7.5.6	Conclusión	70
<b>8</b>	<b>Conclusión final</b>	<b>71</b>
8.1	Presupuesto del proyecto	71
8.2	Próximos objetivos	71
8.3	Conclusión	72
	<b>Bibliografía</b>	<b>73</b>

**II Anexos**

**A Códigos** 77

**B Guiones** 91



# Índice de figuras

2.1	Portada película ‘El Verdugo (1963)’ . . . . .	4
2.2	Portada película ‘Plácido (1961)’ . . . . .	6
2.3	Cortes de secuencia en la película ‘La Escopeta Nacional (1978)’ . . . . .	7
2.4	Gráfica duración promedio escenas de las películas de Berlanga . . . . .	8
2.5	Gráfica 10 planos secuencia más largos de Berlanga . . . . .	9
2.6	Gráfica comparación duración media de secuencias . . . . .	9
2.7	Histograma duración media de secuencias . . . . .	10
2.8	Código detección de escenas con scenedetect en python . . . . .	11
2.9	Comparación media de personajes con 10 líneas de diálogo . . . . .	12
2.10	Histograma de número de personajes de 1220 guiones distintos. . . . .	12
2.11	Frame de la película ‘Todos a la Cárcel (1993)’ con los personajes detectados por el modelo YOLOv8 . . . . .	13
2.12	Media de personajes por escena en cada película . . . . .	14
2.13	Recuento de planos usados en los guiones digitalizados de Luis G. Berlanga . . . . .	15
2.14	Porcentaje de planos usados en los guiones digitalizados . . . . .	15
2.15	Resultados análisis de planos con CineScale . . . . .	16
2.16	Nube de palabras de cada las películas . . . . .	17
2.17	Nube de palabras de los diálogos de todas las películas . . . . .	18
2.18	Nube de palabras de diálogos y descripciones de todas las películas . . . . .	18
2.19	Porcentaje escenas en interior/exterior por película . . . . .	19
2.20	Porcentaje escenas de día/noche por película . . . . .	19
2.21	Evolución de conceptos en las películas de Berlanga . . . . .	21
3.1	Captura de la página web BerlangaFilmMuseum . . . . .	23
3.2	Estado de los guiones mecanografiados de ‘Esa Pareja Feliz’ y ‘Calabuch’ . . . . .	25
3.3	Subtitulando la película ‘Plácido’ mediante el software Subtitle Edit . . . . .	26
3.4	Ejemplo CSV de ‘Esa Pareja Feliz’ . . . . .	27
3.5	Ejemplo CSV de ‘Plácido’ . . . . .	28
4.1	Proceso de una neurona en una red neuronal [22] . . . . .	30
4.2	Sigmoid Function . . . . .	31
4.3	Tanh Function . . . . .	31
4.4	ReLU Function . . . . .	31
4.5	Estructura de una FNN [22] . . . . .	32
4.6	Comparación estructuras RNN y FNN [26] . . . . .	34
4.7	Comparación entre una neurona básica y una neurona de una red LSTM [22] . . . . .	35
5.1	Arquitectura Transformers [27] . . . . .	38

5.2	Estructura de una <i>single-head attention</i> [22]	40
5.3	Estructura de una <i>multi-head attention</i> [22]	40
5.4	Ejemplo de Finetuning [28]	43
6.1	Eficiencia Mistra 7B con respecto a los modelos Llama en diferentes benchmarks [29]	46
6.2	Distribución longitud de los inputs_ ids del Dataset	48
6.3	Captura del resultado del entrenamiento de Mistral7B	50
6.4	Herramienta 'Text Generation Web UI' para LLMs	50
6.5	Comparación Benchmarks de Llama3 con otros modelos [31]	51
6.6	Distribución longitud de los inputs_ ids del Dataset	52
6.7	Comparativa entre ChatGPT 3.5 y ChatGPT4 en diferentes benchmarks [32]	53
6.8	Proceso de Finetuning con ChatGPT 3.5	54
6.9	Resultado Finetuning con ChatGPT 3.5	54
7.1	Porcentaje de planos usados en el guion 'El Autocar'	66
7.2	Porcentaje de planos usados en el guion 'El Autocar'	67
7.3	WordCloud guion 'El Autocar'	67
7.4	Evolución de conceptos con 'El Autocar (1957)'	68
7.5	Porcentaje escenas en interior/exterior por película con 'EL AUTOCAR'	69
7.6	Porcentaje escenas de día/noche por película con 'EL AUTOCAR'	69
A.1	Código detección de escenas con scenedetector en python	77
A.2	Código detección de peronas con YOLOv8	78
A.3	Código detector de planos usados en cada frame	79
A.4	Código generación de Wordclouds	80
A.5	Código conversión archivo .docx con estilos a .csv estructurado	81
A.6	Código conversión de audio a texto mediante Whisper	82
A.7	Finetuning Mistral7B. 1º Paso: Cargar modelo cuantificado	82
A.8	Finetuning Mistral7B. 2º Paso: Cargar tokenizador	83
A.9	Finetuning Mistral7B. 3º Paso: Selección de longitud máxima	84
A.10	Finetuning Mistral7B. 4º Paso: Aplicamos LoRA	84
A.11	Finetuning Mistral7B. 5º Paso: Entrenamiento final	85
A.12	Finetuning Llama 3. 1º Paso: Cargar modelo cuantificado	86
A.13	Finetuning Llama 3. 2º Paso: Cargar tokenizador	87
A.14	Finetuning Llama 3. 3º Paso: Aplicamos LoRA	88
A.15	Finetuning Llama 3. 4º Paso: Entrenamiento final	89

# Índice de tablas

2.1	Tabla de palabras escogidas para cada categoría social . . . . .	21
3.1	Tabla de Películas y Guiones . . . . .	24
4.1	Comparación de funciones de activación [22] . . . . .	31
7.1	Entrenamiento de Llama2 con texto plano . . . . .	56
7.2	Entrenamiento de Llama2 con tokens especiales . . . . .	57
7.3	Entrenamiento con contexto de la escena anterior . . . . .	58
7.4	Escena generada con Mistral7B entrenado . . . . .	60
7.5	Escena generada con Llama8B entrenado . . . . .	62
7.6	Escena generada con ChatGPT 3.5 entrenado . . . . .	65
B.1	Escena original Bienvenido Mr. Marshall. Extracto obtenido de BerlangaFilmMuseum . . . . .	91



# Listado de acrónimos empleados

**ANN** Artificial Neural Network.

**DB** Base de datos.

**FNN** Feedforward Neuronal Networks.

**GPU** Unidad de Procesamiento Gráfico.

**IA** Inteligencia Artificial.

**LLM** Large Language Model.

**NLP** Natural Language Processing.

**OS** Open Source.

**PG** Plano General.

**PGC** Plano General Corto.

**PGL** Plano General Largo.

**PM** Plano Medio.

**PMC** Plano Medio Corto.

**PML** Plano Medio Largo.

**PT** Plano Total.

**Parte I**

**Memoria**



# Capítulo 1

## Introducción y objetivos

En la actualidad, la Inteligencia Artificial (IA) ya abarca prácticamente todas las disciplinas, transformándolas y permitiendo nuevas visiones innovadoras. Con la aparición de modelos avanzados como ChatGPT, Dall-e o YOLO, se han puesto al descubierto todo el abanico de oportunidades que pueden ofrecer todos estos modelos de Inteligencia Artificial.

En concreto, los modelos de generación de texto, conocidos como Large Language Model (LLM), se encuentran en una carrera competitiva entre las principales empresas tecnológicas del momento para conseguir el modelo mas potente y eficaz. Los más conocidos son ChatGPT, de OpenAI; Llama, de META; GEMINI, de Google; CLAUDE de Anthropic o Mistral, de Mistral AI.

La relación entre estos modelos de Inteligencia Artificial y el cine en la actualidad es cada vez más estrecha. Los avances en IA están revolucionando la industria cinematográfica en diversas áreas, desde la creación de guiones y la generación de efectos visuales hasta la mejora en la calidad de la postproducción. Esta integración de IA en el cine no solo optimiza procesos, sino que también abre nuevas fronteras para la experimentación artística y narrativa.

En este contexto surge la idea de realizar este proyecto, cuyo objetivo es combinar las capacidades de los modelos de Inteligencia Artificial más avanzados para analizar la obra cinematográfica de un director de cine e intentar replicarla o imitarla. En concreto, del director de cine valenciano: Luis García Berlanga.

### 1.1. Objetivos

El presente proyecto tiene como objetivo entrenar un modelo de generación de texto (LLM) que sea capaz de producir nuevos guiones con el inconfundible estilo del renombrado director de cine español, Luis García Berlanga. De esta forma, buscamos no solo generar guiones inéditos que reflejen la esencia de su estilo, sino también dar vida a aquellas ideas que Berlanga dejó escritas pero no pudo llevar a cabo debido a las restricciones de la censura franquista o a limitaciones presupuestarias.

Paralelamente se realizará un análisis exhaustivo con IA de toda su obra para identificar las características más importantes. De esta manera, una vez hayamos generado el nuevo guion con nuestro modelo LLM entrenado, podremos comprobar si realmente el modelo ha sido capaz de aprender las características de este director.



Este enfoque tiene el objetivo de fusionar la creatividad del director con la capacidad de análisis de la Inteligencia Artificial. Al hacerlo, no solo rendimos homenaje al director y su obra, sino que también exploramos nuevas aplicaciones de todas estas nuevas tecnologías en el ámbito cinematográfico.

## **1.2. Estructura del proyecto**

Este proyecto está estructurado de la siguiente forma. En el Capítulo 2 se contextualizará sobre la figura de Luis García Berlanga y la importancia de su obra. Además se realizará un análisis con Inteligencia Artificial de toda su obra y se comprobará la relevancia de su característico estilo.

En el Capítulo 3 se describirá la metodología realizada para el preprocesamiento de los guiones, es decir, como se obtuvieron todos los guiones con los que ha sido entrenado el modelo de Inteligencia Artificial.

En el Capítulo 4 se realizará una introducción al Natural Language Processing (NLP), abordando temas como la tokenización o la estructura de distintas redes neuronales como FNN, RNNs o LSTMs. En el Capítulo 5 se explorará el funcionamiento de los Transformers y su proceso de fine-tuning.

En el Capítulo 5 se explicará paso a paso cómo se llevó a cabo el entrenamiento de cada uno de los modelos, y en el Capítulo 6 se describirá primeramente todo el proceso de prompting durante el entrenamiento y se analizarán todos los resultados.

Por último, en el Capítulo 7 se describirá el presupuesto del proyecto junto con las perspectivas futuras del mismo y una conclusión final.

## Capítulo 2

# Análisis de la obra de Luis G. Berlanga con IA

### 2.1. Introducción

Luis García Berlanga (1921 - 2010) fue un renombrado director de cine y guionista nacido en Valencia. Fue un director emblemático de la España de posguerra y se le considera uno de los artistas valencianos más famosos del mundo [1] y el mejor director de comedia europeo, mientras que en España se le considera creador de las películas más importantes del cine español. Su cine se caracterizó por una aguda crítica social y una gran capacidad para retratar la hipocresía y las contradicciones de la sociedad española de la época [2].

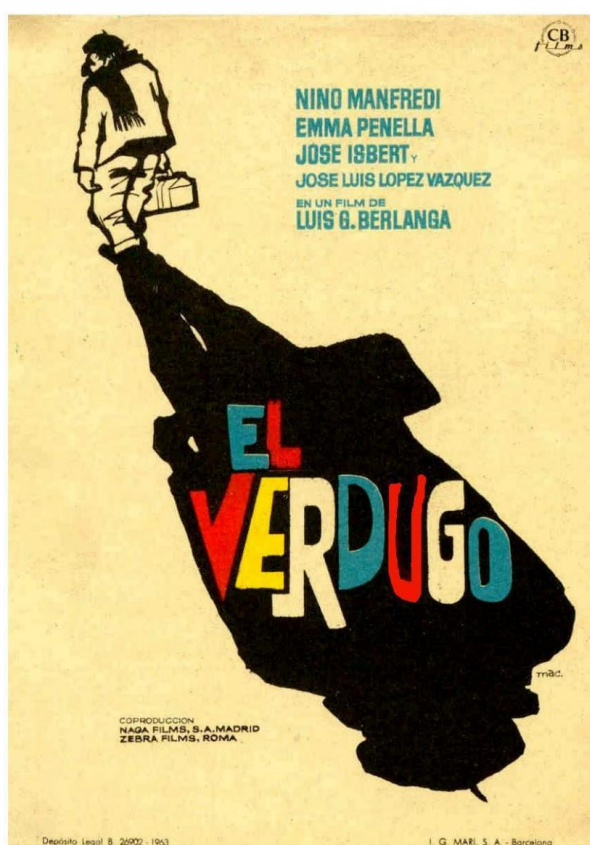
Durante la época de la dictadura franquista, Berlanga tuvo que enfrentarse repetidamente a la censura impuesta a sus guiones. Sin embargo, esto no le impidió mantener un estilo propio muy marcado y conseguir sortear esta censura mediante el uso de la ironía y una representación vívida de la España mundana [3].

Su filmografía se compone de diecisiete largometrajes:

- **Esa Pareja Feliz** (1951)
- **Bienvenido, Míster Marshall** (1953)
- **Novio a la vista** (1954)
- **Calabuch** (1956)
- **Los jueves, milagro** (1957)
- **Plácido** (1961)
- **El verdugo** (1963)
- **La boutique** (1967)
- **¡Vivan los novios!** (1970)

- **Tamaño Natural** (1974)
- **La escopeta nacional** (1978)
- **Patrimonio nacional** (1981)
- **Nacional III** (1982)
- **La vaquilla** (1985)
- **Moros y Cristianos** (1987)
- **Todos a la cárcel** (1993)
- **París-Tombuctú** (1999)

Indudablemente, ‘Bienvenido, Mister Marshall (1953)’ destaca como la película más reconocida entre el público español. La película ‘Plácido (1961)’ le valió una nominación al premio Oscar a la mejor película de habla no inglesa. Finalmente, es con su obra ‘El verdugo (1963)’ (Imagen 2.1) donde muestra la máxima expresión de su talento cinematográfico, siendo posiblemente su obra más aclamada internacionalmente.



**Figura 2.1: Portada película ‘El Verdugo (1963)’**

## 2.2. Lo Berlanguiano

No podemos intentar replicar o imitar el estilo de un director de cine sin antes comprender su obra y su estilo. No es hasta que hayamos comprendido estos elementos fundamentales que no podremos seguir desarrollando el trabajo, para finalmente poder comparar los resultados y comprobar si realmente hemos conseguido el objetivo buscado. Por tanto, preguntémosnos primero: ¿Qué es lo berlanguiano?

En 2020, la Real Academia Española introdujo en el diccionario la palabra ‘Berlanguiano’, la cual definió como: ‘*Berlanguiano. Que tiene rasgos característicos de la obra de Luis García Berlanga.*’. Una definición vacua que no llega a explicar realmente lo que significa este concepto.

El estilo de Berlanga, o lo Berlanguiano, es toda aquella escena (cinematográfica o de la vida real) grotesca e inverosímil pero real. Es una visión personal del cineasta entre el absurdo y lo grotesco, entre la inocencia de unos y la mezquindad de otros, entre el irreverente ácrata y el despota corrupto. Es toda aquella situación coral esperpéntica donde se revelan los ideales morales de los involucrados [4].

Todas sus películas son retratos de una España que se pretendía única, relatos tiernos y crueles a la vez, comedias a primera vista y tragedias en el fondo: tragicomedias en suma, con una gran influencia del movimiento neorrealista italiano de la época, con referencias a Roberto Rossellini o Federico Fellini, muy caracterizados por combinar el humor, la ternura y la crítica social [4].

En el universo berlanguiano lo colectivo precede al individuo, los personajes se ven obligados a actuar en las distintas situaciones, puesto que viven en un mundo mezquino, sometidos a los grandes poderes: clero, burguesía y estado [5].

En definitiva, su estilo responde a un fracaso colectivo. Es alrededor del pobre que se vertebra lo berlanguiano y su capacidad de hacer humor en la desgracia. Es dentro de esta coralidad que se narra la tragedia individual de un personaje en sus afanes por conseguir algo que está fuera de su alcance. El fracaso al que se ve abocado el protagonista no representa únicamente el suyo, sino también el de la sociedad en su conjunto. Y es en el final de sus películas donde todas estas tragedias desembocan en un lugar común, dando lugar a la ‘*mascletá*’ final

*Todas mis películas son la crónica de un fracaso. En ellas siempre he tratado el tema de alguien que quiere conseguir algo y nunca lo logra.*

Luis G. Berlanga [2]

Para quien no sea conocedor de su obra no hace falta más que leer la sinopsis de alguna de sus películas para ver reflejado su estilo. Por ejemplo, la película Bienvenido Mr Marshall: ‘*El Alcalde de un pueblecito castellano llamado Villar del Río recibe la visita del Delegado General, quien le anuncia la inmediata llegada de una delegación del gobierno de los EE.UU como parte de un Plan de Recuperación Europea. Las fuerzas vivas deciden organizar a los americanos una calurosa acogida con la ayuda de una cantante folclórico, Carmen Vargas, cambiando la fisonomía del austero municipio castellano, convirtiéndolo en un típico pueblo andaluz, colorido y alegre*’ [6].

También podemos observar este estilo en la sinopsis de Plácido (Imagen 2.2): ‘*En una pequeña ciudad provinciana, a unas burguesas ociosas se les ocurre la idea de organizar una campaña navideña cuyo lema es: ‘Siente a un pobre a su mesa’. Se trata de que los más necesitados compartan la cena de Nochebuena con familias acomodadas y disfruten del calor y el afecto que no tienen.*

*Plácido ha sido contratado para participar con su motocarro en la cabalgata, pero surge un problema que le impide centrarse en su trabajo: ese mismo día vence la primera letra del vehículo, que es su único medio de vida” [7].*



**Figura 2.2: Portada película ‘Plácido (1961)’**

Querría destacar, por último, que es injusto darle el mérito de este estilo tan característico únicamente a Berlanga. Durante toda su carrera los guiones los escribió conjuntamente con muchas otras personas. Su primeros guiones (hasta ‘Novio a la Vista (1954)’ fueron escritos y dirigidos junto con Juan Antonio Bardem. Desde ‘Plácido (1961)’ hasta ‘Moros y Cristianos (1987)’ escribió los guiones conjuntamente con el novelista y guionista Rafael Azcona. También destacan otros colaboradores como: Edgar Neville, Miguel Mihura, José Luis Colina o Jorge Berlanga. Es gracias al conjunto de las visiones de todos estos guionistas, y en especial la de Azcona, que se alcanzó este estilo tan reconocible.

En resumen, hay tres elementos cinematográficos que distinguen indudablemente a Berlanga: los planos secuencia, la coralidad y su uso de la cámara. Vamos a realizar un análisis detallado de estos elementos junto a algunas otras características que también considero importantes.

## **2.3. Análisis de lo ‘berlanguiano’ con Inteligencia Artificial**

### **2.3.1. Planos secuencia**

Sin duda alguna, los planos secuencia son una de las señas de identidad más reconocibles de Berlanga y es el rasgo de estilo que más admiración suscita. Todos los actores confluyen en estos caóticos planos secuencias de forma muy natural y ordenada (aún a pesar del caos que estas suponen). Es debido a la coralidad de sus películas que necesita de este recurso, pues si se le quiere dar un multiprotagonismo a todos los personajes con un mismo porcentaje de participación, el plano-contraplano cuadra mal [2].

Un plano secuencia es un plano que incorpora movimientos de cámara dinámicos con encuadres de toma larga, muy a menudo abarcando la acción de varias escenas que, de otro modo, se cubrirían con varios planos separados [8].

Tras un arduo proceso de digitalización de las películas gracias a la ayuda de la Biblioteca de la UPV, de la cual he podido extraer la gran mayoría de sus películas, se ha podido analizar la longitud de todas las secuencias de cada una de las películas.

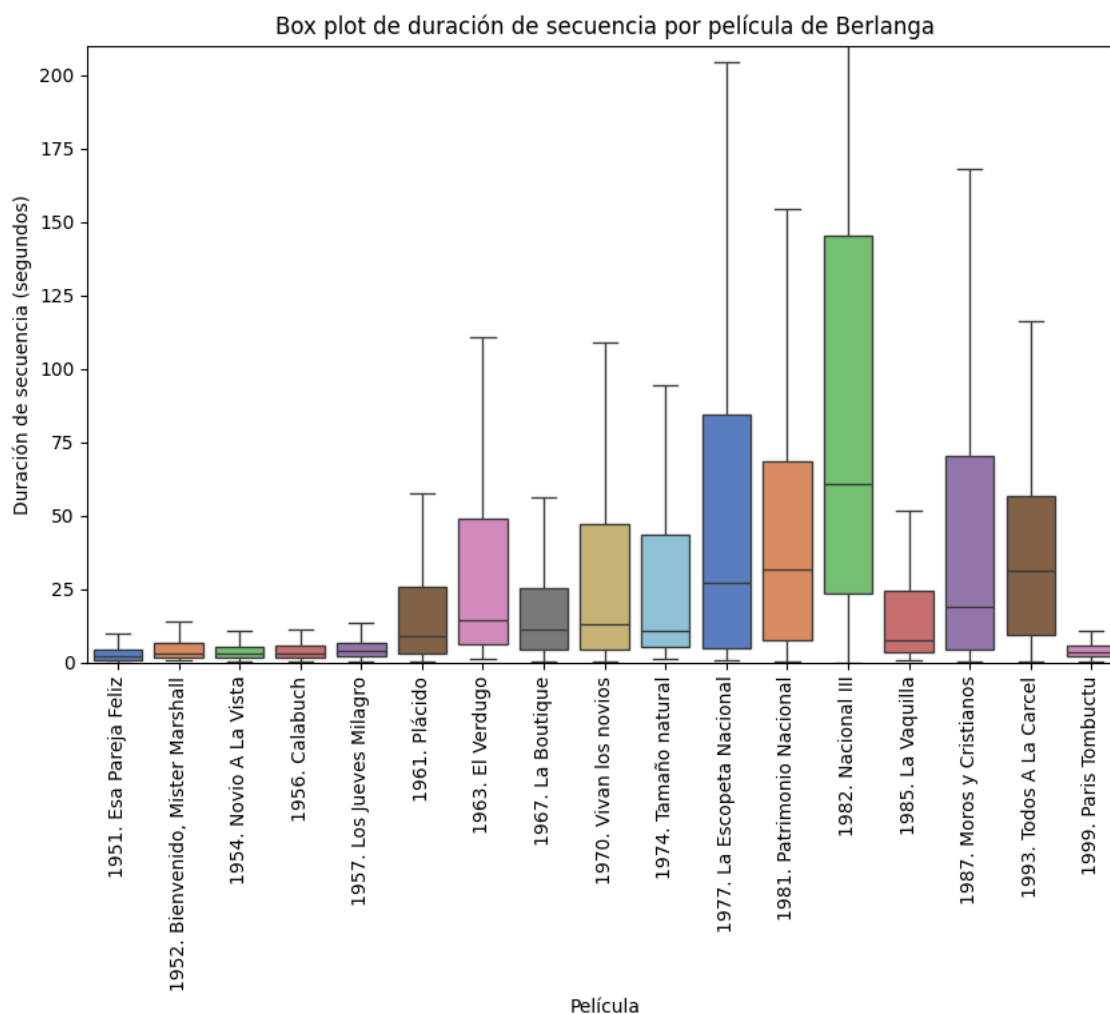
Para ello he hecho uso del paquete de python `scenedetect` [9]. Este permite obtener el *timestap* exacto en el que se produce un corte en la película, es decir, un cambio de escena. Mediante el modo `AdaptativeDetector` comparamos la diferencia entre cada fotograma y sus fotogramas adyacentes con un *score ratio* que al ser excedido detecta un cambio de escena. En la imagen 2.3 se muestra los 5 primeros cortes de la películas 'La Escopeta Nacional (1978)'.



Figura 2.3: Cortes de secuencia en la película 'La Escopeta Nacional (1978)'

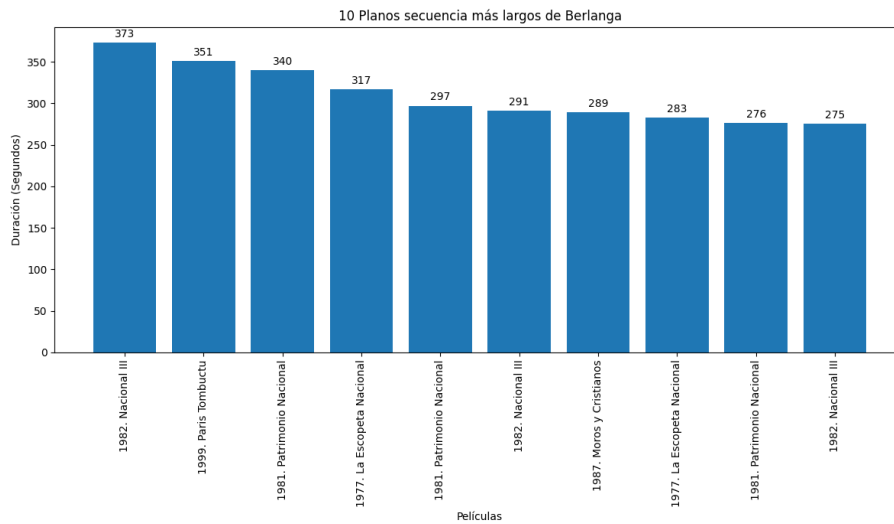
Algunas películas son muy antiguas y en blanco y negro, por lo que los *timestap* no son perfectos. Es por ello que se ha necesitado realizar un revisión manual de los resultados, añadiendo nuevos cortes y eliminando cortes detectados erróneamente. El código de este apartado se puede consultar en el anexo A.1.

Una vez ya tenemos todos los resultados podemos juntarlos y analizarlos. En la imagen 2.4 se muestra el tiempo promedio de duración de las escenas. En esta gráfica se puede observar la evolución del propio director. En sus primeras películas el plano secuencia no sería característico; no sería hasta ‘Plácido (1961)’ que Berlanga comenzaría a usar este recurso de forma recurrente en todas sus películas, siendo ‘Nacional III (1982)’ la que más duración tendrían sus secuencias.



**Figura 2.4: Gráfica duración promedio escenas de las películas de Berlanga**

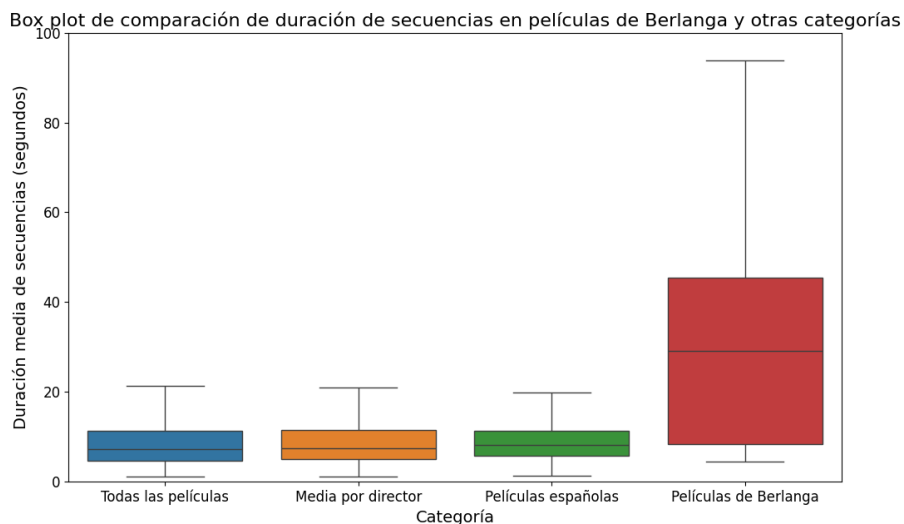
Podemos aprovechar esta información para intentar descubrir cuales fueron los planos secuencias más largos de Berlanga. En la imagen 2.5 observamos que los resultados obtenidos concuerdan con la imagen anterior (2.4), pues son precisamente las películas que más uso hicieron de los planos secuencia en las que aparecen las secuencias más largas. Como primer lugar destaca la escena de ‘Nacional III (1982)’ con una duración de 373 segundos (6 minutos y 12 segundos).



**Figura 2.5: Gráfica 10 planos secuencia más largos de Berlanga**

Realmente este no sería su plano secuencia más largo, pues en 2002 realizó un cortometraje en plano secuencia con una duración de 603 segundos (10 min y 3 segundos): ‘El sueño de la maestra’ [3]. Pero para este estudio solo estamos teniendo en cuenta sus largometrajes, dejando de lado cualquier cortometraje o serie de televisión.

Podemos comparar estos resultados con el resto de directores de cine para ver como de importante es esta característica en Berlanga. Para ello se ha usado la base de datos de Cinemetrics, una base de datos con 22.840 películas en las que se incluye la información de la duración media de las secuencias [10]. En la gráfica 2.6 se puede observar la comparación de la duración media de secuencias entre: las películas de Berlanga, todas las películas de la base de datos, el conjunto de películas de cada director y únicamente las películas españolas de la base de datos.



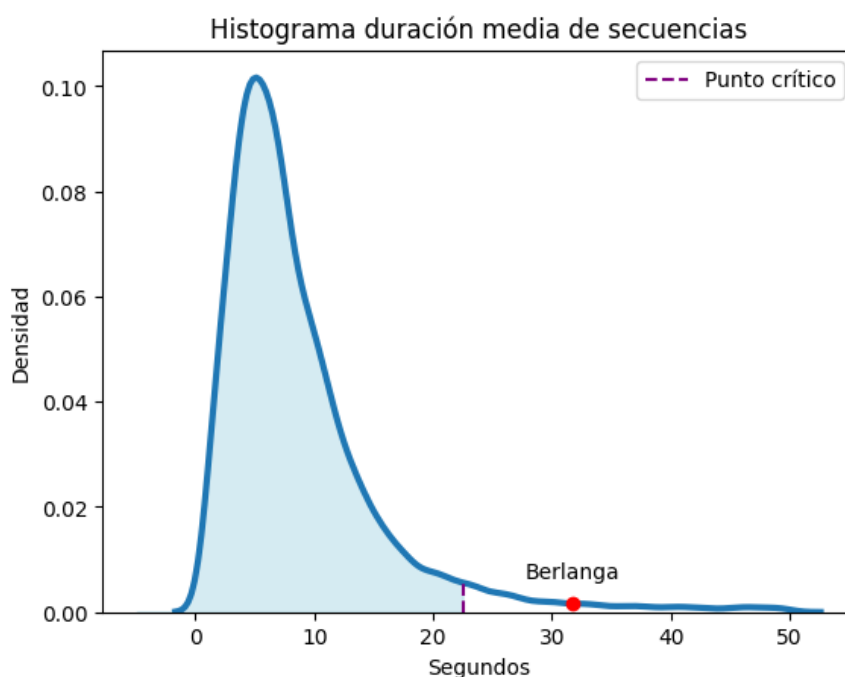
**Figura 2.6: Gráfica comparación duración media de secuencias**



Se puede apreciar la grandísima diferencia que hay entre Berlanga y el resto de comparados. Podemos hacer uso de una función de distribución para comprobar si efectivamente esta característica de Berlanga es realmente una anomalía. Para ello haremos uso de la ecuación 2.1, la cual marca el punto crítico de la función de distribución.

$$\text{Punto crítico} = \mu + 2\sigma \quad (2.1)$$

En la siguiente imagen (2.7) se muestra el histograma de la duración media de todas las secuencias de la base de datos nombrada previamente, junto con el punto crítico y el lugar que ocupa el resultado concreto de Berlanga.



**Figura 2.7: Histograma duración media de secuencias**

De esta forma podemos concluir que la longitud de las secuencias es algo característico de Luis G. Berlanga en comparación con una gran cantidad de directores de cine.

### 2.3.2. Coralidad

Salvo en ciertas excepciones, como ‘Tamaño natural’ o ‘La Boutique’, todas las películas del director valenciano se caracterizaron por no tener un único protagonista y estar plagado de personajes que aportaban todos los estratos de la sociedad. Ni la película de ‘Plácido’, que lleva el nombre de uno de los personajes se libró de esta característica.

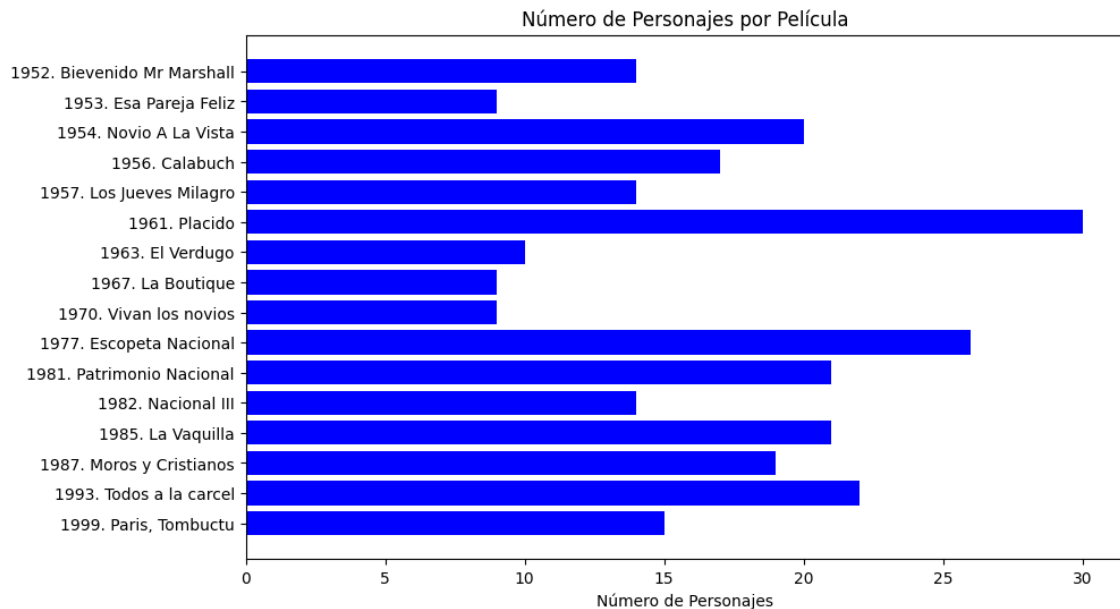
Su estilo se caracterizó por tener muy presentes a todos esos personajes de la España de postguerra. Sus películas tienen siempre personajes que representan al pobre inocente, al pícaro fracasado, al político corrupto, al patetismo del ya irrelevante aristócrata, al desubicado cura retrógrado o a las aprovechadas fuerzas vivas del pueblo. La presencia de tantos personajes en escena supuso una

técnica cinematográfica que el director valenciano supo demostrar de manera sobresaliente; de forma caóticamente ordenada. Berlanga quiso salir del drama personal del yo, que tanto abundaba en el cine de la época (y el actual), para entrar en el conjunto, en lo plural [3].

Como veremos en el apartado 3.1, se han conseguido digitalizar muchos de los guiones de las películas de Berlanga con los cuales podemos intentar demostrar esta característica.

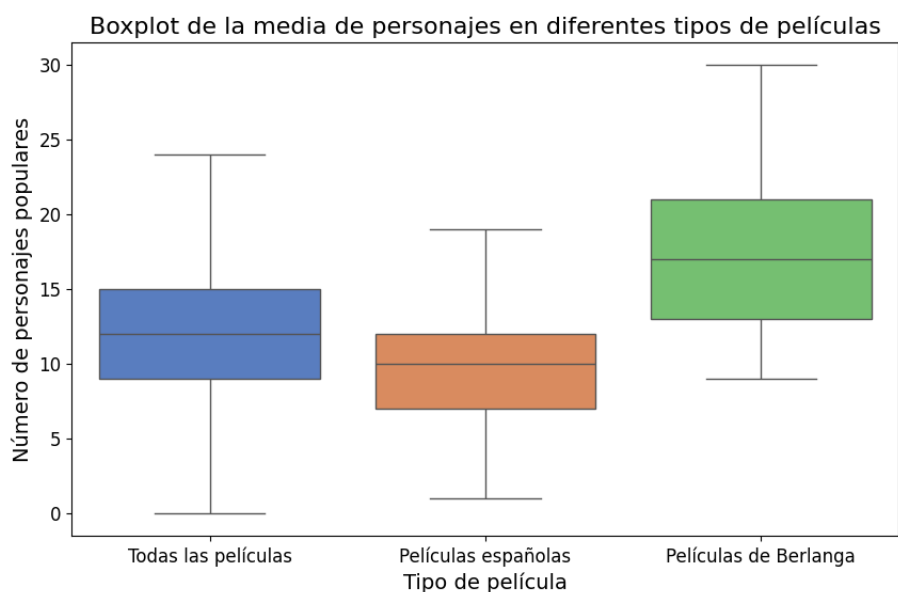
Vamos a considerar que para que un personaje se le considere importante en una secuencia tiene que, por lo menos, tener 10 líneas de diálogo; siendo 10 un valor arbitrario escogido por mi para realizar este análisis. De esta forma estamos quitando todos esos personajes extras que solo realizan pequeñas intervenciones.

Si realizamos un conteo en los guiones de todos los personajes que hablan más de 10 veces obtenemos la gráfica 2.8. En esta se observa como es en la película de Plácido donde más personajes encontramos, seguido por 'La Escopeta Nacional (1977)' y 'Todos a la Cárcel (1993)'.



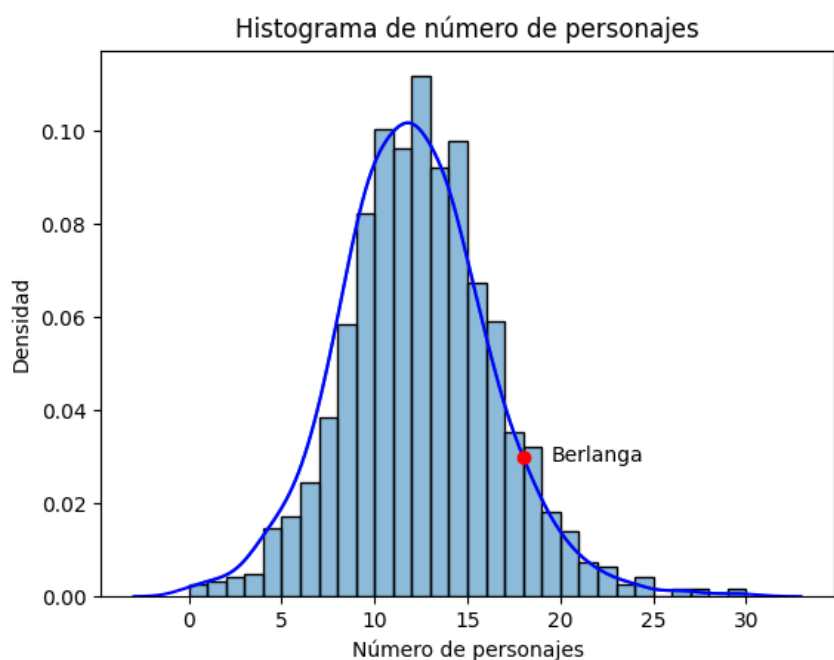
**Figura 2.8: Código detección de escenas con scenedetect en python**

Por otro lado, para poder comparar esta característica con otros directores, he conseguido obtener 1220 guiones de películas gracias a la web [www.aprendercine.com](http://www.aprendercine.com). Tras un largo proceso de conversión de formato para que todas estén en el mismo formato que los guiones de Berlanga, se ha podido comparar el número de personajes. En la gráfica 2.9 se puede observar la comparación de la media del número de personajes entre: las películas de Berlanga, todas las películas de la base de datos y solamente las películas españolas de la base de datos.



**Figura 2.9: Comparación media de personajes con 10 líneas de diálogo**

En esta gráfica se observa como las medias están medianamente igualadas, aunque las películas de berlanga siguen destacando. Si se realiza una vez más una función de distribución podremos analizar más detalladamente esta característica. Véase la imagen 2.10 en la cual se representa un histograma con el caso anterior de todas las películas y se indica el punto exacto donde se encuentra Berlanga.



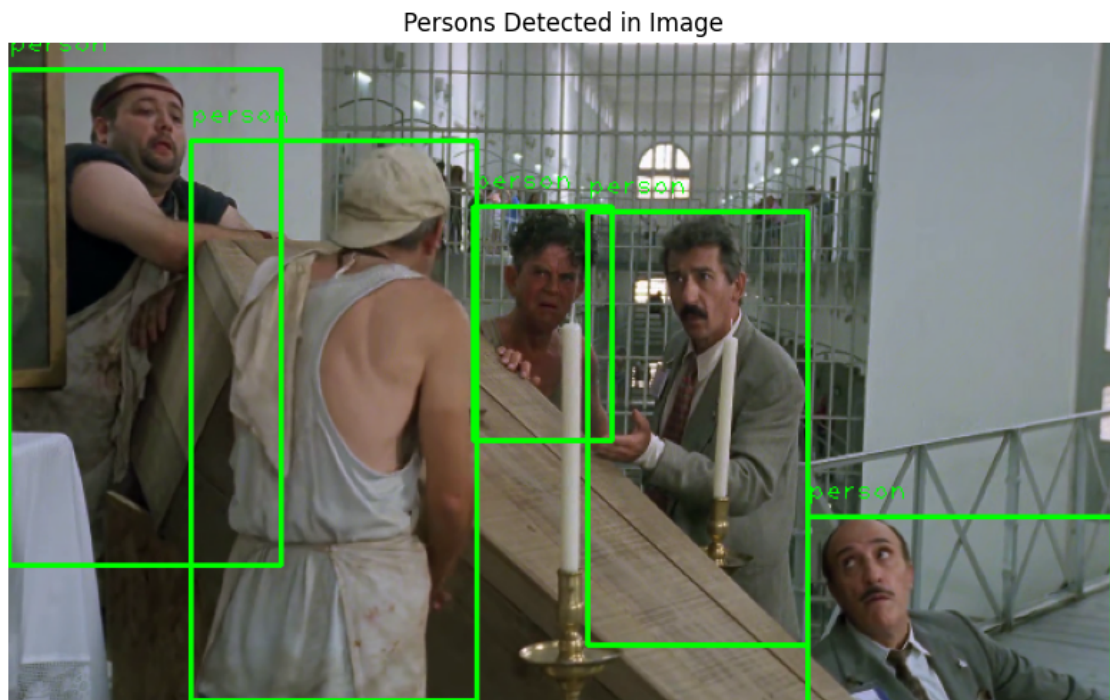
**Figura 2.10: Histograma de número de personajes de 1220 guiones distintos.**

Es sin duda sorprendente que a partir de los datos obtenidos con los 1220 guiones se haya obtenido una gaussiana casi perfecta y de la cual se pueden sacar muchas conclusiones interesantes. En lo que respecta a nuestro estudio actual, observaremos que el valor de Berlanga se encuentra bastante a la derecha. Si calculamos el punto crítico tal y como se hizo en la ecuación 2.1 obtendríamos un valor cercano a 20, por lo que en este caso el valor de Berlanga sí que entra dentro de la distribución y no podríamos considerar que se trata de un valor realmente atípico. Eso no quita que sea un número elevado para lo estándar y que se deba tener en cuenta.

Otro método que podemos utilizar para comprobar esta característica es mediante un detector de IA. Aprovechando que en el apartado anterior conseguimos obtener los cortes entre escenas podemos calcular la media de personajes por escena y, posteriormente, la media de personajes por escena por película.

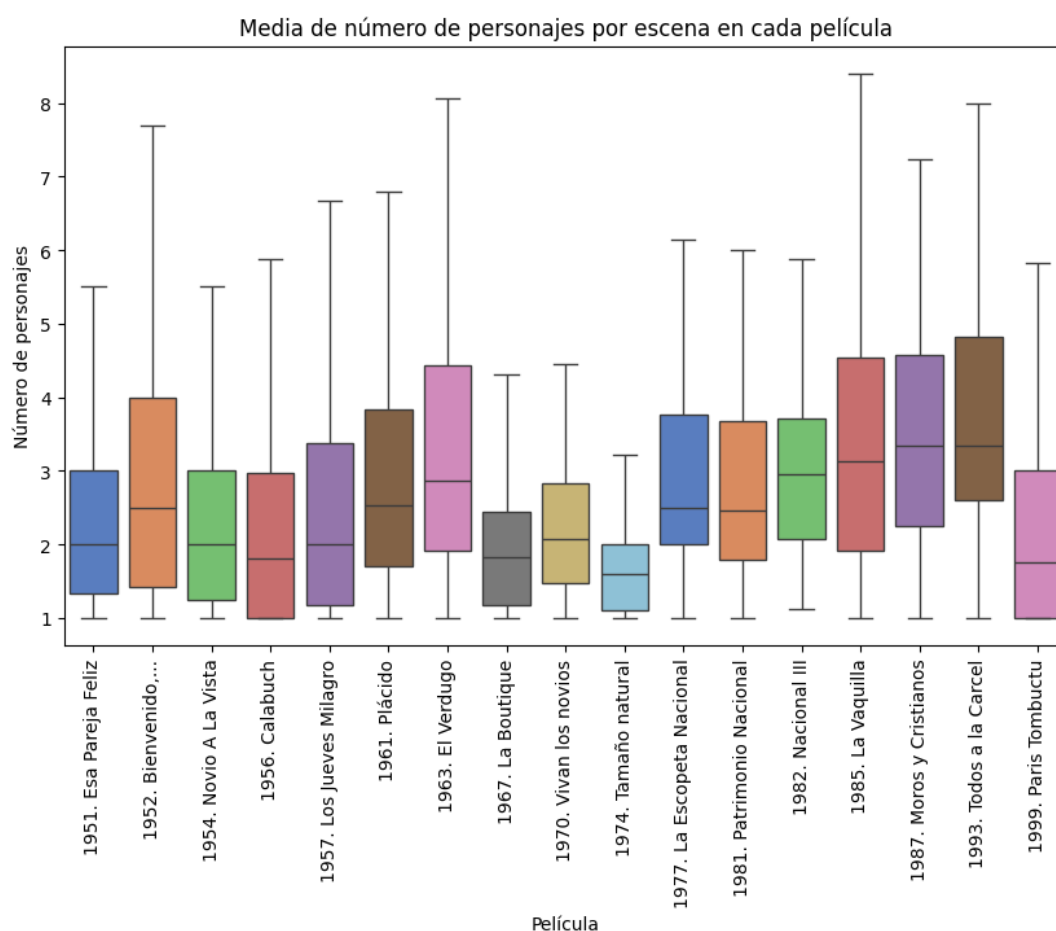
Para ello se utilizó primeramente el paquete de python `face_recognition` [11]. Sin embargo encontré muchos problemas, pues caras lejanas y personajes de espaldas no los detectaba. Por ello decidí cambiar la lógica y en vez de detectar caras, detectar personas. Para esta tarea se utilizó el modelo YOLOv8 [12], un modelo de detección de objetos capaz de detectar cuando hay una persona en escena. El código de este proceso se puede encontrar en el anexo A.2.

En la imagen 2.11 se muestra un ejemplo de un frame de la película 'Todos a la Cárcel (1993)' en el que se puede apreciar cómo el modelo ha sido capaz de detectar correctamente a diferentes personas aún a pesar de estar de espaldas o tapadas por objetos.



**Figura 2.11:** Frame de la película 'Todos a la Cárcel (1993)' con los personajes detectados por el modelo YOLOv8

En la siguiente imagen (2.12) se puede observar la media de personajes por escena en cada película. Se apreciará que, aunque la media sean valores bajos, hay muchos casos con escenas que contienen gran número de personajes. Además hay que tener en cuenta que en cualquier película encontraremos muchas escenas sin personajes en plano y con un único actor en pantalla. En el caso de Berlanga, se consigue que en algunas películas haya de media más de 3 personas en pantalla de manera constante.



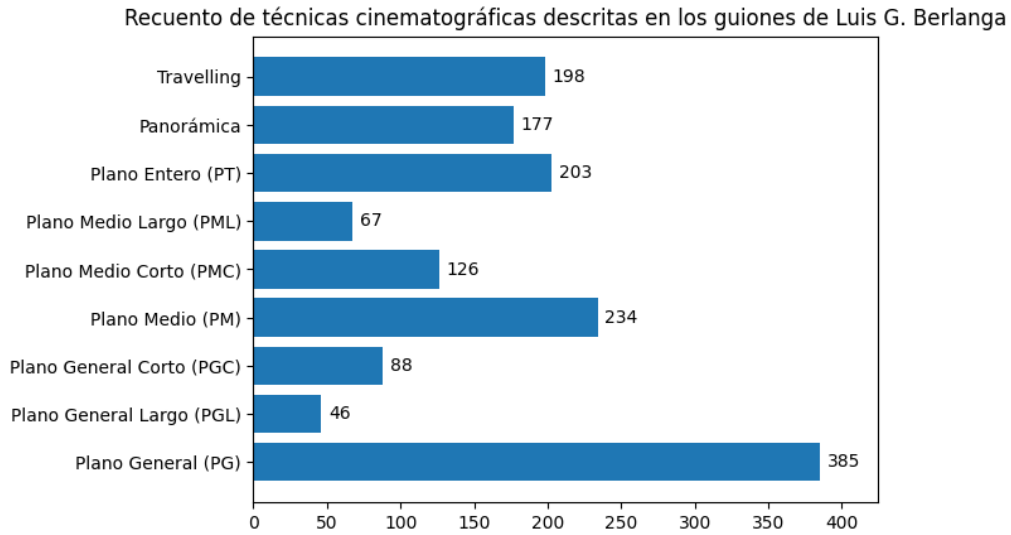
**Figura 2.12: Media de personajes por escena en cada película**

Observaremos como la coralidad es algo diferenciador para Berlanga en comparación con muchos otros directores de cine. Queda de esta forma demostrado que la cantidad de personajes en las películas es algo característico.

### 2.3.3. Encuadre

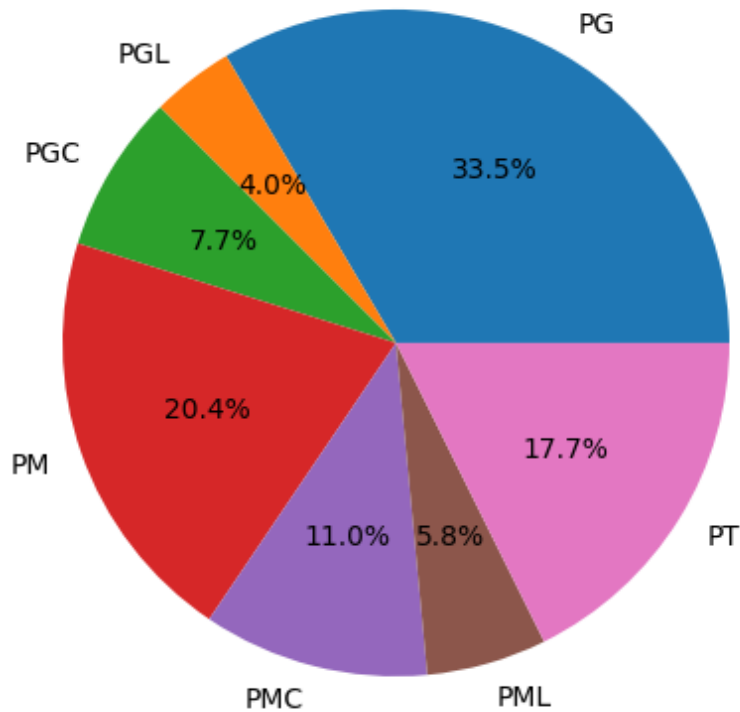
El objetivo de Berlanga era que la cámara fuese invisible e ignorable. Utilizaba mucho el plano general, la profundidad de campo y el plano de conjunto (como ya hemos visto). En interiores, la cámara se colocaba en un rincón de la casa, con el objetivo de ser olvidada y poder grabar todo el espacio posible. Además, era también habitual el uso del plano general en el inicio de la película para mostrar el entorno donde se ubican los personajes [13].

Puesto que tenemos acceso a muchos de los guiones de sus películas, y en estos se indica como son los planos a utilizar, se puede realizar un análisis de los planos más usados.



**Figura 2.13: Recuento de planos usados en los guiones digitalizados de Luis G. Berlanga**

Mediante una gráfica circular queda reflejado de forma más clara el uso de los distintos planos:



**Figura 2.14: Porcentaje de planos usados en los guiones digitalizados**

Hay que tener en cuenta que la información sobre el uso de los distintos planos solo la encontramos en sus primeros guiones. Más adelante dejaría de escribir estos tecnicismos y los decidiría en el mismo rodaje [6]. Es por ello que, aunque esta información tiene su punto de relevancia, podría no estar revelando su estilo cinematográfico durante toda su carrera y tan solo la de sus inicios. Es por ello necesario hacer una análisis más extenso.

El proyecto *CineScale* es un proyecto realizado por el Departamento de Ingeniería de la Universidad de Brescia (Italia) con el cual, mediante el uso de un modelo de Inteligencia Artificial, podemos analizar los diversos planos de sus películas [14, 15, 16]. El modelo presume de un 94 % de precisión por lo que los resultados que obtengamos serán bastante acertados.

El modelo es capaz de distinguir, a partir de un frame dado, el tipo de plano que se ha utilizado. Primero se han extraído los frames de todas las películas, se han convertido a una imagen de formato PIL [17] y posteriormente se ha enviado al modelo para que prediga que tipo de plano es el más probable para el frame dado. El código de todo el proceso de utilización de este modelo se puede encontrar en el anexo A.3.

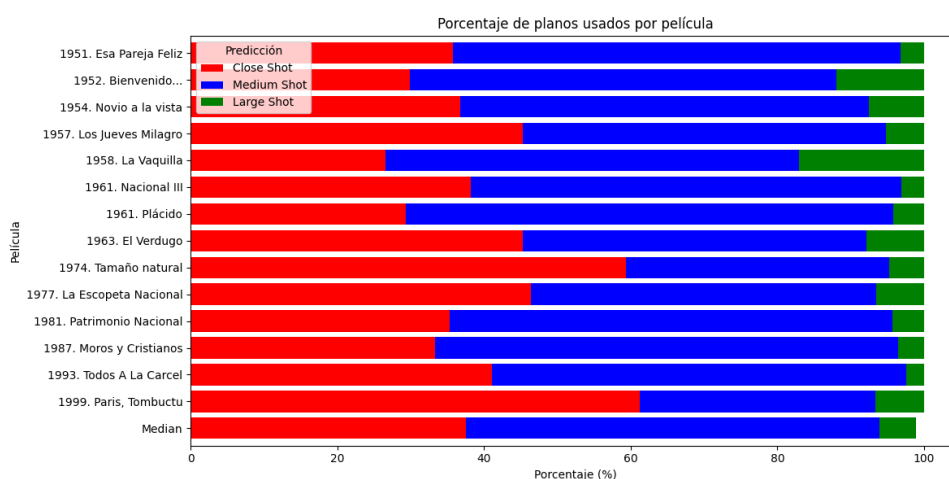


Figura 2.15: Resultados análisis de planos con CineScale

Observamos que la nueva gráfica coincide con la imagen 2.14. Lo más característico de Berlanga es el uso de Planos cortos (PM, PMC, PGC) y planos medios (PG, PT) por igual. Por otro lado, hay una gran ausencia de planos amplios, a excepción de los de inicio de película que comentábamos en el principio de este apartado.

### 2.3.4. Vocabulario

Aprovechando que tenemos digitalizados todos sus guiones podemos realizar una nube de palabras y ver qué palabras son las más utilizadas. Una nube de palabras es un representación visual de todas las palabras de un texto, siendo las palabras más grandes las que tienen mayor frecuencia [18].

Para ello se ha utilizado el paquete de python WordCloud [19] el cual, habiendo previamente contado el número de veces que aparece cada palabra con el paquete sklearn [20], hemos podido realizar la nubes de palabras representadas en la imagen 2.16. El código de este apartado se puede consultar en el anexo A.4.



Figura 2.16: Nube de palabras de cada las películas

Si se juntan todos estos guiones y extraemos una única nube de palabras obtendremos una representación más clara del vocabulario berlanguano.

La primera nube que podemos obtener es una que reúna únicamente las palabras de los diálogos entre personajes. Podemos destacar algunos conceptos como: ‘pobre’, que representa la importancia de las clases bajas; ‘papá’ y ‘mamá’, que muestra la importancia de la familia y ‘dios’ que sin duda demuestra la reiterada utilización de la religión en sus películas. También aparece ‘dinero’ pero este concepto lo comentaremos más adelante.





Figura 2.17: Nube de palabras de los diálogos de todas las películas

Por otro lado, si construimos una nube de palabra que incluya las descripciones de escenas del propio Berlanga obtenemos la imagen 2.18. Vemos como las palabras más usadas representan a los distintos estamentos de la sociedad: ‘cura’, que representa al clero, ‘pueblo’ que representa a las clases bajas, ‘teniente’ que representa a las fuerzas militares y ‘señor’ que es usado de forma constante para referirse a las clases altas. De esta forma observamos la gran importancia que la daba Berlanga a la diferenciación entre estas clases sociales.



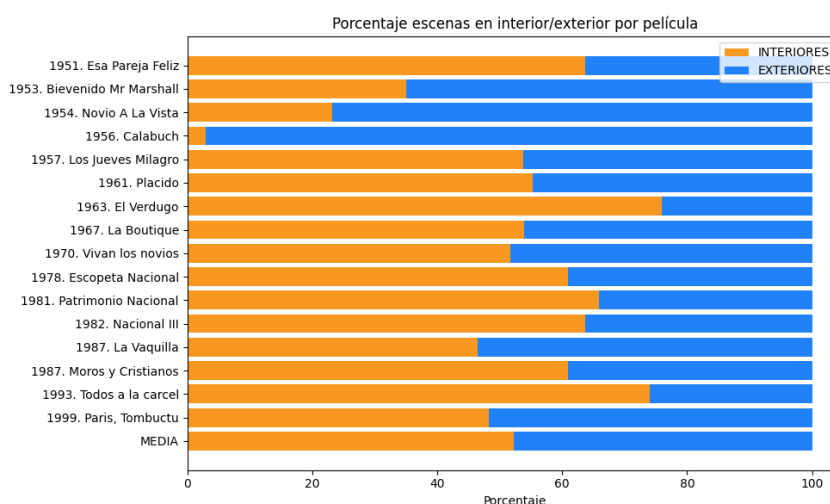
Figura 2.18: Nube de palabras de diálogos y descripciones de todas las películas

### 2.3.5. Otras características

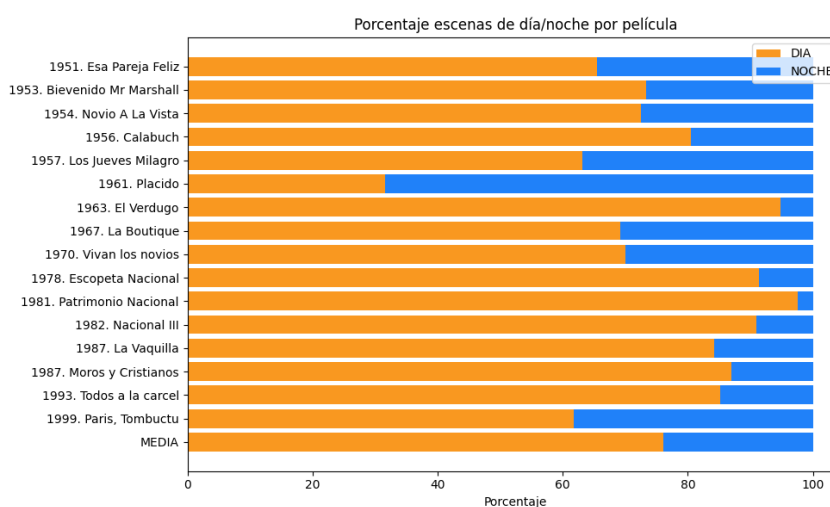
Aprovechando la digitalización de los guiones, podemos analizar ciertos aspectos que hasta ahora habían quedado fuera de las visiones más profundas de la obra de este director y que pueden dar lugar a una nueva reflexión sobre la misma.

#### 2.3.5.1. Uso de las ubicaciones

Si examinamos detalladamente las ubicaciones representadas en los guiones digitalizados, observaremos que el 76% de las escenas corresponden al período diurno, mientras que un 52.2% de estas tienen lugar en interiores. Esta información puede ser corroborada con mayor claridad al analizar las dos imágenes que se presentan a continuación (imagen 2.19 e imagen 2.20)



**Figura 2.19: Porcentaje escenas en interior/exterior por película**



**Figura 2.20: Porcentaje escenas de día/noche por película**

### 2.3.5.2. Austro-húngaro

Otro detalle, bien conocido por cualquier admirador de la obra del director valenciano, es su fijación por introducir la palabra ‘austro-húngaro’ en todas sus películas. Berlanga era una persona muy supersticiosa y creía que añadir esta palabra en sus películas le daría buena suerte [3]. ¿Pero es la única palabra que se nombra en todas sus películas? Aprovechando que se han digitalizado todas sus películas y tenemos acceso a todos los diálogos he decidido comprobar qué otras palabras salen en todas sus películas pero no nos habíamos percatado de ello hasta ahora.

Con un simple código de python podemos comprobar qué palabras se repiten en todos los guiones. Lógicamente estas palabras tienen que ser pronunciadas por un personaje, no vale si la palabra está en la descripción de una escena o de la ubicación.

Una vez ejecutado el código obtenemos muchas palabras sin valor, como pronombres y determinantes. De todas estas palabras podemos destacar el gran uso de la primera persona, en contraste con las demás personas gramaticales: ‘me’, ‘vamos’, ‘nuestro’, ‘nos’, ‘tenemos’, ‘hemos’, ‘digo’, ‘puedo’, ‘estoy’, ‘mis’, ‘mi’... Todos estos posesivos podría reflejar la importancia de la propiedad (en ocasiones la personal y en otras la comunal) en Berlanga. Según Berlanga el ser humano no puede ser libre, pues la sociedad, la familia y los compromisos nos atrapan de nuevo en el momento que el ser humano desea desatarse. Este uso constante de la primera persona del plural reflejaría un recuerdo constante a estas ataduras ineludibles. Por otro lado, el uso del singular reflejaría ese egoísmo e individualismo que tanto representa a los distintos personajes de sus películas.

Otros conceptos que se repiten en todos sus guiones son palabras como: ‘señor’, ‘mujer’ u ‘hombre’. No encuentro mucho valor en ‘hombre’ y ‘mujer’, pues en muchos casos no son más que interjecciones que no aportan un valor real. Sin embargo, ‘señor’ sí que muestra la constante aparición de una división jerárquica social en todas sus películas.

Por último, me gustaría destacar una palabra que sorprenderá a más de uno. Pues la palabra ‘dinero’ aparece en todas sus películas. Este constituye un claro ejemplo de la relevancia del dinero en su filmografía. Aunque tratado desde distintos puntos de vista, el dinero siempre es importante en el universo berlanguiano: bien para que las fuerzas vivas obtengan sus ganancias, bien porque el pobre lo necesita para poder vivir o bien porque el aristócrata necesita continuar lucrándose.

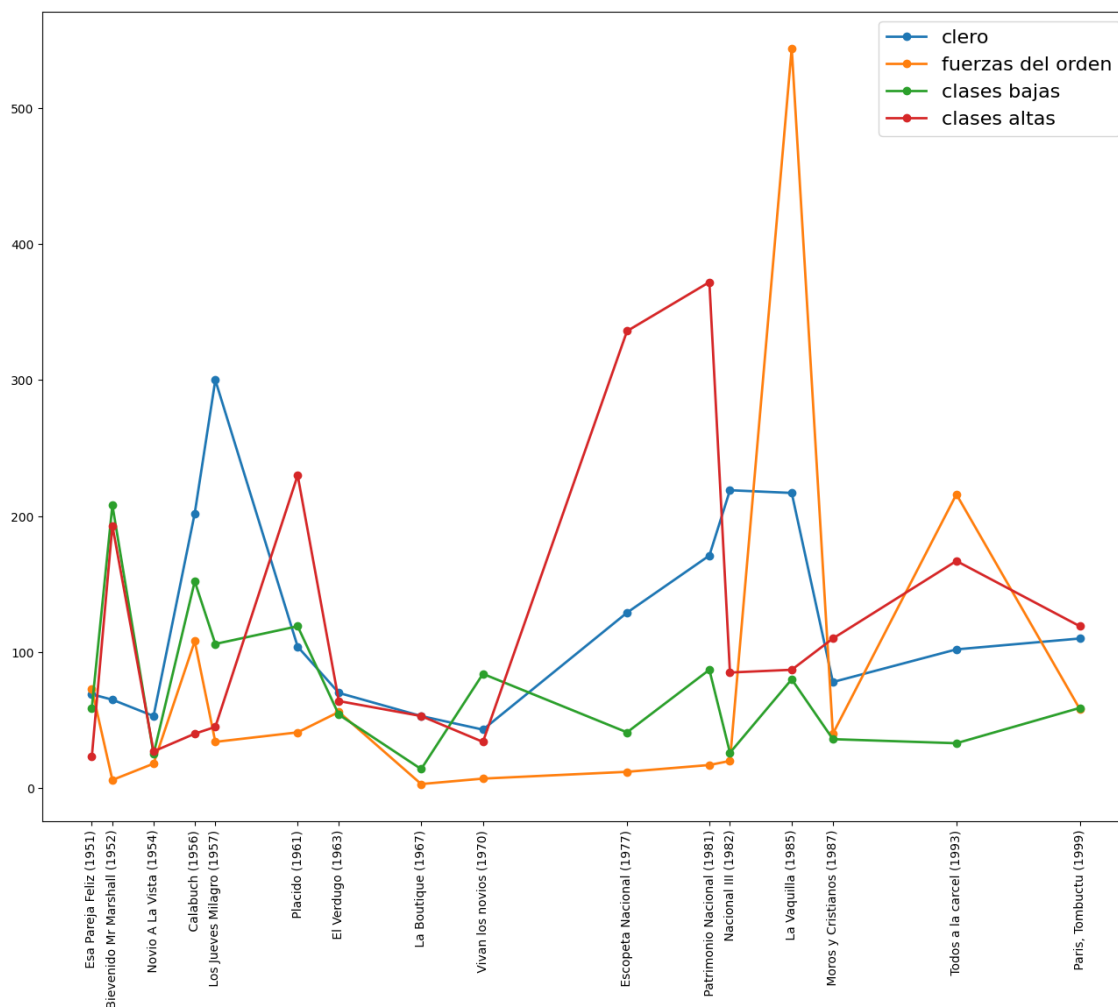
### 2.3.5.3. Evolución de conceptos

Hasta ahora hemos señalado la inexorable presencia en su filmografía de la tétada: clases bajas, clases altas, institución religiosa y fuerzas del orden. Podemos realizar un análisis de la evolución de la presencia de estas en cada una de sus películas. En la tabla 2.1 se muestran las palabras utilizadas para representar cada concepto.

Una vez hemos seleccionado las palabras que representarán cada concepto, podemos hacer un seguimiento cronológico de estos y comprobar cual es la presencia de cada categoría en cada una de las películas, tal y como se muestra en la imagen 2.21.

**Tabla 2.1: Tabla de palabras escogidas para cada categoría social**

Categoría	Palabras
Clero	San, santo, fraile, milagro, misa, epístola, Jesucristo, espíritu, clero, versículo, cura, monja, obispo, iglesia, religión, bendición, pecado, fray, monaguillo
Fuerzas del orden	Sargento, policía, teniente, guardia, carabinero, soldado, brigada, comandante, antidisturbios
Clases bajas	Pobre, empleado, pueblo, profesor, profesora, gente, pregonero, vecino, campesino
Clases altas	Rico, notario, príncipe, delegado, alcalde, rey, marqués, marquesa, condesa, ministro, empresario, político



**Figura 2.21: Evolución de conceptos en las películas de Berlanga**

La gráfica revela información muy interesante. Primeramente se observa una preocupación por las clases bajas en sus primeras películas que acabaría siendo sustituida con los años hacia una gran importancia hacia las clases altas. Sería durante la ‘Trilogía Nacional’ (Trilogía formada por ‘La Escopeta Nacional’, ‘Patrimonio Nacional’ y ‘Nacional III’) y ‘Plácido’ donde más queda patente la presencia de la burguesía y la aristocracia.

Por otro lado, la presencia de la iglesia se mantiene medianamente estable durante toda sus películas a excepción del gran salto en la película ‘Los Jueves, milagro’, cuyo argumento gira sobre la aparición de un milagro religioso. Aunque, como es lógico, el tratamiento que se hace a la Iglesia durante la dictadura franquista no es comparable a la etapa tras la muerte de Franco, pues en esta segunda oleada de películas se muestra mucho más crítico hacia los estamentos eclesiásticos.

Es, por tanto, el estamento religioso el único poder que aparece de forma clara en casi todas las películas, en ocasiones ejerciendo su gran influencia y en otras mostrando sus efectos, pero en todos los casos dejando siempre sentir su inmenso predominio en la sociedad. Algunos críticos han entendido esta constante presencia como un recordatorio de que está ahí y en cualquier momento puede despertar y ejercer su autoridad [2].

En cambio, las fuerzas del orden no llegan a tener gran relevancia durante sus películas, a excepción de esporádicas apariciones de guardias o carabineros, ni siquiera en una película rodada completamente en una prisión, como es el caso de ‘Todos a la cárcel’ llegan a tener un papel relevante. La única película donde alcanza una gran importancia es en ‘La Vaquilla’, la cual está ambientada en la Guerra Civil Española.

Por último, se puede observar la presencia de la censura franquista en esta evolución. Tras la película de ‘El verdugo’ la censura le puso mucho más difícil rodar películas. Al no poder rodar películas en España tuvo que rodar en otros países como Argentina o Francia, en donde, por contrato, tuvo que realizar cine más comercial y que, por lo tanto, no se aprecia tanto la crítica social durante el periodo 1963 - 1977.

#### **2.3.5.4. Objetivos fotográficos**

Para finalizar, podemos señalar los objetivos usados en sus películas. Los guiones no suelen incluir esta información, pero cuando lo hacen los más comunes son: 20 mm, 35 mm, 40 mm, 50 mm, 75 mm, 150 mm, 155 mm o 330 mm.

## Capítulo 3

# Datos: metodología y preprocesado

Para realizar el entrenamiento de una Inteligencia Artificial de generación de texto la parte más importante y laboriosa es la recolección de todos los textos con los que va a ser entrenado el modelo. A continuación se va a exponer el proceso realizado para la obtención de los guiones y su posterior procesamiento.

### 3.1. Digitalización de los guiones

Para este proyecto se ha obtenido los guiones de dos fuentes distintas. La primera ha sido mediante la página web <https://berlangafilmuseum.com> en la cual se encuentran digitalizados algunos de sus guiones (aunque no todos), tal y como se puede observar en la imagen 3.1.

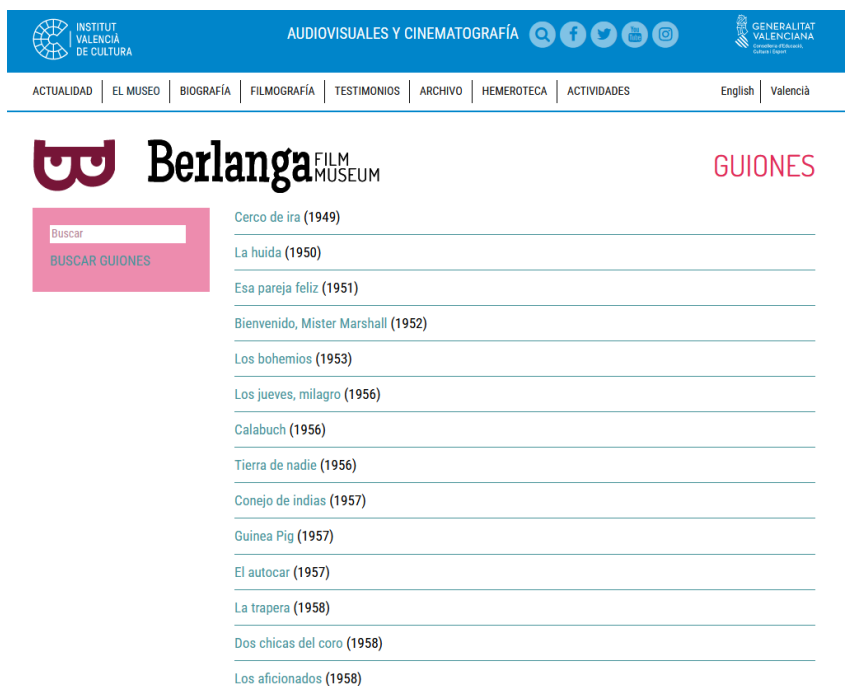


Figura 3.1: Captura de la página web BerlangaFilmMuseum

El segundo método ha sido con las propias películas. Gracias a la filmoteca de la UPV se han podido sacar las películas de cuyos guiones carezco y extraer el audio de cada una de ellas.

En la siguiente tabla (3.1) se muestra alguna información de las características y método de obtención de cada guion.

**Tabla 3.1: Tabla de Películas y Guiones**

Año	Película	¿Realizado?	Tipo de guion	Extracción	Palabras
1950	La Huida	✗	Escenas y diálogos	PDF	40,967
1951	Esa Pareja Feliz	✓	Escenas y diálogos	PDF	30,290
1952	Bienvenido, Mr Marshall	✓	Escenas y diálogos	PDF	23,568
1954	Novio A La Vista	✓	Solo diálogos	Audio	10,713
1956	Calabuig	✓	Escenas y diálogos	PDF	23,557
1957	Los Jueves Milagro	✓	Escenas y diálogos	PDF	38,440
1961	Plácido	✓	Solo diálogos	Audio	16,334
1963	El Verdugo	✓	Escenas y diálogos	Audio	22,532
1967	La Boutique	✓	Solo diálogos	Audio	13,226
1970	Vivan los Novios	✓	Solo diálogos	Audio	7,332
1976	Una Noche Embarazosa	✗	Escenas y diálogos	Audio	15,218
1978	La Escopeta Nacional	✓	Solo diálogos	PDF	16,026
1980	Patrimonio Nacional	✓	Solo diálogos	PDF	22,561
1982	Nacional III	✓	Solo diálogos	Audio	19,677
1984	La Vaquilla	✓	Escenas y diálogos	PDF	26,602
1987	Moros y Cristianos	✓	Solo diálogos	Audio	19,213
1993	Todos a la carcel	✓	Solo diálogos	Audio	16,099
1998	París, Tombuctú	✓	Escenas y diálogos	PDF	22,586
TOTAL					439,341

La columna de '¿Realizado?' representa aquellos guiones de películas que no se llevaron a grabar pero que aún así he digitalizado. En la columna 'Tipo de guion' se indica cómo está estructurada la información, ya que hay guiones que tienen todo tipo de detalles (Ubicación, escenas, diálogos, tipos de planos...) mientras que otros solo indican el personaje y su diálogo. En 'Extracción' se observa el método de extracción utilizado, los cuales se detallarán más profundamente a continuación. Por último, la columna final indica el número de palabras de cada guion.

Puesto que algunos de los guiones son de películas que no se llevaron a cabo por múltiples motivos (censura, falta de financiación o ideas desechadas) no se han escogido todos. Es importante remarcar que los guiones deben reflejar el estilo que queremos conseguir. Puesto que algunos guiones son muy tempranos o muy fuera de lugar, no se han escogido para realizar el entrenamiento. Este es el caso de 'Una Noche Embarazosa', el cual digitalizé en un principio, pero finalmente decidí descartar por no ser apropiado para este proyecto.

No todos los guiones que encontramos en esta web son guiones completos. Algunos de ellos como 'El autocar (1957)' o 'La trapería (1958)' son pequeñas ideas de guiones de Berlanga. Es por tanto, el objetivo de este proyecto conseguir convertir esas ideas en guiones completos.

Vamos a ver a continuación como se han realizado los dos métodos de extracción de texto de forma más detallada.

### 3.1.1. Extracción de guiones en PDF

El problema principal que encontré con estos guiones es que no eran pdf con texto plano, sino que eran imágenes de documentos mecanografiados de la época. Esto me impedía simplemente copiar y pegar el texto; era necesario extraerlo de otra forma.

Para ello, hice uso del software OCR ‘ABBY FineReader PDF ®’ el cual me permitía visualizar el documento a la vez que transcribía el resultado ofrecido por su OCR integrado.

Sin embargo, esto no solucionaba del todo el problema. Puesto que los guiones son documentos antiguos mecanografiados, se encontraban en muy mal estado (en ocasiones con manchas y tachones), por lo que el OCR fallaba mucho. Por este motivo me vi obligado a tener que transcribir los guiones de forma casi completa. En la imagen (3.2) se puede observar el mal estado en el que se encontraban algunos de los guiones.

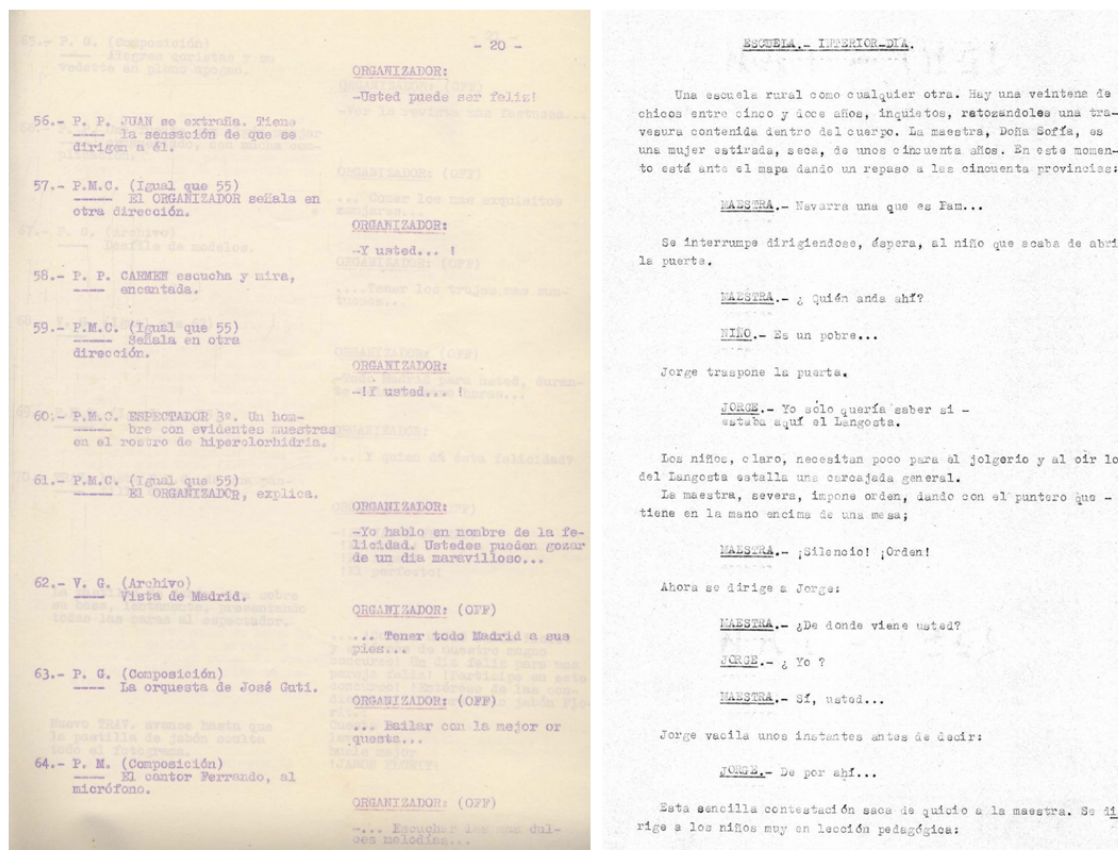


Figura 3.2: Estado de los guiones mecanografiados de ‘Esa Pareja Feliz’ y ‘Calabuch’

### 3.1.2. Extracción de guiones con Audio

Gracias a la filmoteca de la UPV he podido obtener el resto de películas que no tenía el guion escrito. El proceso de digitalización es más sencillo, he hecho uso de la herramienta de OpenAI: Whisper. Este es un red neuronal Open-Source capaz de transcribir audio en texto. El código utilizado para este proceso se puede encontrar en el anexo A.6



Una vez teníamos los diálogos transcritos era necesario añadir quién era el personaje que lo decía, por ello fue necesario la utilización del software opensource Subtitle Edit [21]. Con este iba visualizando la película a la vez que añadía en el archivo .srt el nombre del personaje que hablaba y la ubicación de la escena con el formato descrito previamente 3.5.

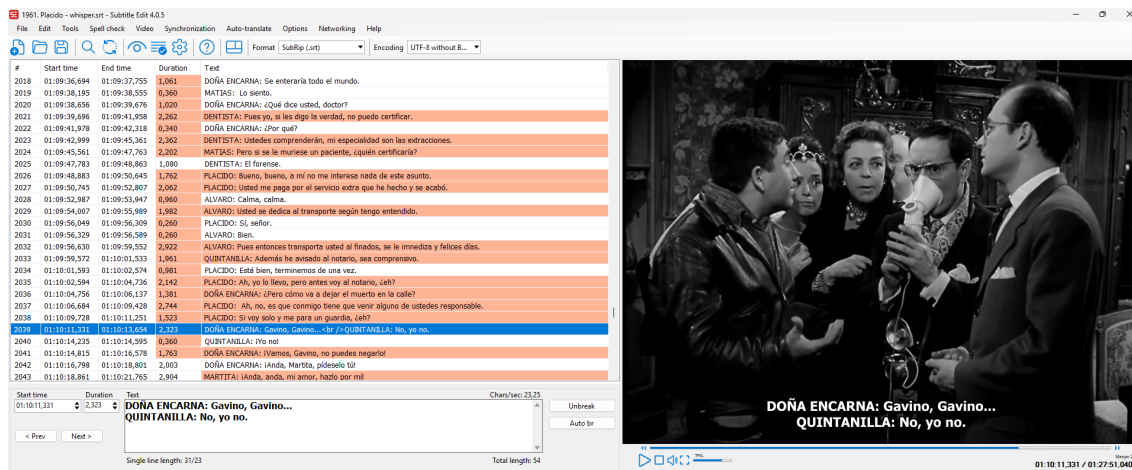


Figura 3.3: Subtitulando la película ‘Plácido’ mediante el software Subtitle Edit

### 3.2. Preprocesado

Los guiones se han intentado mantener lo más auténticos posibles. Sin embargo, me he visto obligado a procesar el texto y realizar ciertas modificaciones para ayudar al modelo a comprender mejor los textos. Estos cambios han sido:

- Traducción al español. Algunos diálogos son dichos por personajes extranjeros que hablan en otro idioma. Para no confundir al modelo LLM con idiomas que no sean español he preferido traducir estos textos e indicar al principio del diálogo entre paréntesis el idioma del personaje. Esto se ha hecho únicamente en los diálogos con idiomas extranjeros, cuando se habla en español no se ha añadido esta coletilla. V.g. ‘JUAN: (En inglés) ¡Hola!’
- Nombres en mayúsculas. Para ayudar al modelo a comprender mejor los nombres propios y darle más importancia a cada personaje, se ha puesto en mayúscula todos los nombres de personaje. De forma que un texto podría ser: *¿No habrás visto a CARMEN, verdad?*
- Universalización del formato de las ubicaciones. Durante la digitalización de los guiones no tardé en darme cuenta de que las ubicaciones de escena eran muy distintas en cada guion. Por ejemplo, algunos daban todo tipo de detalle (CASA JUAN. INT. DÍA), otros solo algunos (EXT. NOCHE), otros indicaban la hora (INT. 19:00) y otros no incluían la ubicación. Para ello he universalizado todas estas ubicaciones en un formato único que indica: el lugar, si es interior o exterior y si es de día, noche, atardecer... Como algunos guiones no daban esta información he tenido que ir comparando el texto con las escenas de las películas para ser lo más preciso posible.

- Faltas ortográficas. Se han corregido todas las que son inintencionadas por no aplicar un valor real al estilo. Aunque si un personaje habla de forma incorrecta de forma intencionada en ese caso se ha mantenido.

Una vez digitalizados todos los guiones era necesario distinguir los tipos de textos, pues, como veremos más adelante, no se puede entrenar el modelo con el texto sin procesar. Para ello era necesario organizar los textos según el tipo: escena, ubicación o diálogo. De ese modo se decidió utilizar el software Microsoft Word® y se crearon estilos personalizados, de forma que posteriormente un código python pudiese detectar estos estilos y guardar los distintos textos en un .csv. Para más claridad ver el código del anexo A.5

En definitiva, el resultado de este código se puede observar en la imagen 3.4. El csv se compone de dos columnas, la primera indica el tipo de texto, mientras que la segunda el texto en sí. Es por ello que hay tres tipos de textos: ESCENA, UBICACIÓN y (NOMBRE DE PERSONAJE). De esta forma podremos tratar el texto de forma mucho más cómoda en el futuro.

TIPO	TEXTO
UBICACION	PATIO PRIMERO CASA JUAN. Ext Noche
ESCENA	ESCENARIO NATURAL: Es un patio largo, destartalado, con ropa tendida en las ventanas. En el suelo, un conglomerado extraño de carruajes y objetos. Hay una berlina, una motocicleta con sidecar, un derrengado coche de niño, al
ESCENA	P. G. (Angulo bajo TRAV, avance desde principio de plano.)
ESCENA	La CAMARA se acerca lentamente, mostrándonos el lugar dejando siempre el viejo taxi a derecha de cuadro. Por debajo del taxi, asoman unas piernas enfundadas en un mono de mecánico. JUAN, en bicicleta, entra en cuadro por
ESCENA	G. JUAN a izquierda, en P. M. En S. T. frontal, las piernas bajo el coche, JUAN termina de atar su bici. Nuevamente, como una exhalación, entra LUIS por la derecha y cruza el FOT., saltando por encima de las piernas que salen de d
JUAN	¿Pero qué le pasa...? Detrás del motor, emerge la cabeza de MANOLO, que mira en dirección de la salida de LUIS, extrañadísimo. Emerge sola la cabeza; tenemos la sensación de que esa cabeza y las piernas que salen de debajo
MANOLO	¿Se debe estar entrenando...!
ESCENA	MANOLO continúa su faena; su cabeza, solo su cabeza, se desplaza a derecha e izquierda. O las piernas no son suyas o aquello es un Juego de magia.
ESCENA	P. G. En primer término las piernas que salen de debajo del taxi.
ESCENA	Juan en S. T. ha terminado de amarrar la bici y se va hacia el interior del patio, hacia derecha de cuadro. Pregunto como fórmulas
JUAN	¿Como va, "facundo"?
ESCENA	P. G. Frontal de la cabeza de MANOLO y las piernas. Continúa el juego de magia. MANOLO, sigue con la mirada la marcha de JUAN.
MANOLO	¡Pss...! Tiene mucho amor propio, pero los sábados, ya se sabe, limpieza general... Me da mucha lata. Y es lo que digo yo. Como se puede...
ESCENA	Durante su monólogo, MANOLO se ha puesto al fin de pie y ha roto el equívoco de la magia. Aquellas otras no son sus piernas.
ESCENA	P. G. JUAN, a punto de salir de cuadro y antes de desaparecer tras el taxi, corta el monólogo de MANOLO.
MANOLO	(OR) ... pensar en comprar...
JUAN	¿No habrás visto a Carmen, eh?
MANOLO	(OR) ¿Qué...? Ah, no, no la ha visto...
ESCENA	JUAN se despide de MANOLO con un gesto y desaparece tras el taxi.
ESCENA	P. G. JUAN, acaba de trasponer el taxi y viene hacia la CAMARA. Hay un corto TRAV, retro, para dar entrada por derecha de cuadro a LUIS. El movimiento cesa al encuadrar a LUIS en P. M. Luis, de perfil, mira a izquierda de cuadro y
LUIS	¡Pero si tiene que estar...! Bueno, bueno, está bien... Ya le he dicho que no puedo ir sin el gorro... Que sí, que sí, pero si voy sin gorro, me fusilan...
ESCENA	JUAN llega a la altura de LUIS, en el momento oportuno. Dice su comentario y sale de cuadro por la izquierda.
JUAN	¡A que no!
ESCENA	LUIS se da cuenta de lo que ha oído y mira en la dirección de JUAN, le reconoce y sonríe.
ESCENA	P. G. JUAN se aleja de espaldas a la CAMARA por derecha de cuadro, volviéndose un poco para saludar y sonreír a LUIS, en escorzo a izquierda de cuadro en P.M.C.
JUAN	¿Que...? ¿Las cosas van mal?
LUIS	De primera. ¡Esta noche a dormir al cuartel y mañana, maniobras...!
JUAN	Eso te conviene, recluta.
LUIS	¡Bah...!
ESCENA	JUAN entra en la casa y se despide de LUIS.
ESCENA	P. G. JUAN sube por la escalera, cruzándose con un personaje que baja penosamente, llevando un manojó de globos y un artilugio con molinillos de papel. Se saludan
PERSONAJE	¡Hola!
JUAN	¡Hola!
ESCENA	Abre y entra
UBICACION	COMPLEJO PASILLO-COMEDOR CASA JUAN. Int. Noche.
ESCENA	P. M. JUAN cierra la puerta por dentro. Al llegar a la entrada del comedor, se detiene un instante, mientras la CAMARA inicia un TRAV lateral hacia la izquierda, hasta descubrir el comedor. Cenando alrededor de la mesa, están EST
ESCENA	JUAN, ha avanzado, pasando junto a la mesa; le pega una palmada al ABUELO.

Figura 3.4: Ejemplo CSV de ‘Esa Pareja Feliz’

En el caso de los guiones obtenidos mediante transcripción de audio, una vez finalizado el proceso desarrollé un código que convirtiese el archivo .srt (formato estándar de archivo de subtítulos) resultante en un .csv con la misma estructura de antes (aunque sin descripciones de escena en este caso). Podemos ver un pequeño ejemplo de uno de los .csv obtenidos en la imagen 3.5.

### CAPÍTULO 3. DATOS: METODOLOGÍA Y PREPROCESADO

TIPO	TEXTO
UBICACION	CALLE. EXT. NOCHE
ALVARO	Voy a acercar al padre un momento, enseguida vuelvo, ¿eh?
EMILIA	Vamos, vamos.
JULIAN	Cuidado que se me escurre.
QUINTANILLA	No, no puedo más, no podíamos acercar el motocarro. Tire, tire, si son dos pasos.
EMILIA	Ayúdela, padre. Cójame a la chica.
PLACIDO	¿Y ahora qué hacemos? Suba, suba usted primero. Alá, arriba. Ande.
CONCHETA	Dios mío.
JULIAN	Echa usted una mano.
QUINTANILLA	Sí, sí, sí.
PLACIDO	Un poco más.
CONCHETA	Taparlo bien.
PLACIDO	Vamos a atarlo, Julián.
CONCHETA	Con lo que a él le gustaba ir en coche.
JULIAN	Deja, deja.
PLACIDO	Bueno, a todo esto, esa gente no me ha dado la indemnización.
QUINTANILLA	Ah, no, caramba. Bueno, eso mañana.
PLACIDO	¡Ni hablar! Que mañana no querrán saber nada.
QUINTANILLA	Que sí, hombre, para.
PLACIDO	Ahora de buena gana dejaba al muerto tirado en la calle.
QUINTANILLA	La viuda, por caridad. Además, lo importante era recoger la letra, ¿no?
EMILIA	Bueno, vámonos de una vez.
QUINTANILLA	Sujétenlo bien.
JULIAN	Descuide, señor.
QUINTANILLA	A ver si se nos va a perder.
PLACIDO	¡Que arranco! ¡Vamos!
UBICACION	CALLE. EXT. NOCHE
DOÑA ENCARNA	Bueno, menos mal, todo arreglado. Mucho gusto.
DOÑA MARIA	El gusto ha sido nuestro, señora. Matías, despídete de doña Encarnación.
MATIAS	Buenas noches, señora. Y a usted, don Polín, muchas gracias.
EDMUNDO	Adiós.
MATIAS	Adiós, buenas noches. Adiós, amigo. Siento no haberle podido proporcionar una buena noche.
ALFONSO	No, si yo no me voy. Bueno, lo digo porque como hemos dejado la cena a la mitad, a mí no me importa volver y terminarlo.
MATIAS	Sí, claro. En fin, pase usted. Pero estará fría. Pase, pase usted.
UBICACION	CASA NOTARIO. INT. NOCHE

**Figura 3.5: Ejemplo CSV de ‘Plácido’**

## Capítulo 4

# Introducción al Natural Language Processing (NLP)

Antes de comenzar con el proceso de entrenamiento de un modelo LLM debemos comprender cómo funcionan estos modelos exactamente para intentar obtener el mejor resultado posible [22].

Dentro del estudio de la Inteligencia Artificial se halla la rama de Natural Language Processing (NLP), la cual se encarga de entender y procesar el lenguaje humano. Esta tarea se complica al no poder los ordenadores trabajar con texto, sino con números. De ahí la importancia de estos modelos, pues son capaces de asignar un porcentaje de probabilidad de una palabra en un texto y así predecir qué palabra es más probable que aparezca a continuación dado un texto cualquiera.

En este capítulo se introducirá el proceso de Tokenización y las arquitecturas de las redes neuronales (ANN) más básicas [23].

### 4.1. Tokenización con BPE

La tarea de un tokenizador es ayudar al ordenador a procesar el texto de forma comprensible. Esta herramienta permite dividir cualquier texto en unidades más pequeñas conocidas como *tokens* [24].

Existen distintos tipos de tokenizadores, cada uno con una lógica diferente. El más usado actualmente por los modelos LLM del mercado es: *Byte-Pair Encoding* (BPE), utilizado por OpenAI y muchos otros modelos Transformer.

BPE funciona sustituyendo bytes consecutivos de texto por bytes que no ocurren en el texto [25]. De esta forma se asegura que las palabras más comunes sean representadas en el vocabulario como de un único token mientras que las palabras menos comunes son separadas en dos o más subtokens.

BPE comienza con un vocabulario con todas las letras individuales. Tras examinar todo el texto y va añadiendo al vocabulario concatenaciones de letras que sean muy comunes (como ‘ch’, ‘ll’ o ‘rr’ en el caso del español). De esta forma si un tokenizador ha sido entrenado con ‘grabar’ y ‘actuar’, al encontrarse con la palabra ‘filmar’, la describirá como el conjunto de ‘film + ar’, pues ambos tendrán su representación en forma de token.

Una vez conocemos los conceptos básicos del tokenizador podemos continuar con las ANN.

## 4.2. ANN

Las Artificial Neural Network (ANN) son herramientas computacionales fundamentales para el procesamiento de texto basadas en la estructura del cerebro humano y diseñadas para realizar tareas de aprendizaje. Estas redes consisten en un conjunto de unidades computacionales pequeñas las cuales, dados los valores de un vector de entrada, producen un único valor de salida. Es decir, son redes capaces de predecir cual es la palabra más probable dada una secuencia de texto.

La arquitectura más importante es Feedforward Network pues va computando iterativamente desde una capa de unidades a la siguiente. Pero vamos a ir paso a paso y vamos a enfocarnos por el momento en la unidad fundamental, la neurona artificial.

### 4.2.1. Neuronas artificiales

Una neurona artificial es el bloque de construcción fundamental de una red neuronal (ANN). Una neurona toma una serie de números con valor como input y, tras realizar una serie de cálculos, produce una salida.

El proceso concretamente es el siguiente: tenemos una serie de inputs  $(x_1, \dots, x_n)$ , que son los textos tokenizados previamente. Al entrar el input en la neurona se le aplican unos coeficientes o *weights*  $(w_1, w_2, \dots, w_n)$ . Estos coeficientes son valores numéricos que determinan la importancia de cada entrada y son aprendidos por la red durante el aprendizaje. Por último, se le añade el *bias* ( $b$ ), el cual desplaza la función de activación con el objetivo de ajustar el modelo correctamente a los datos. De esta forma podríamos representar las neuronas como la ecuación 4.1.

$$z = b + \sum_i w_i \cdot x_i \quad (4.1)$$

El resultado,  $z$ , es una función lineal de  $x$ . Sin embargo, las neuronas aplican una función no lineal al output, permitiendo de esta forma que la red neuronal aprenda y represente relaciones complejas entre las entradas y las salidas. A este cálculo se le conoce como *activation* ( $a$ ). Podemos ver representado todo el proceso en la imagen 4.5

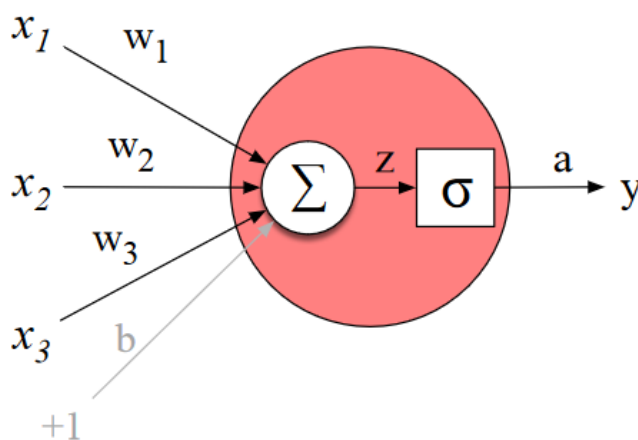
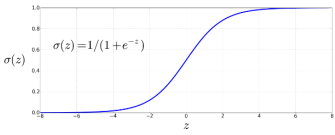
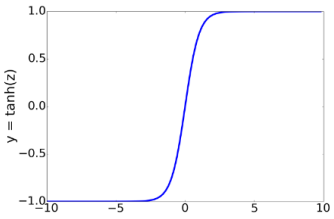
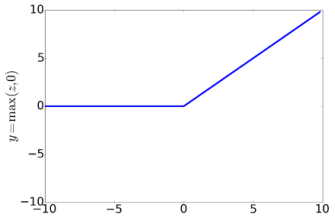


Figura 4.1: Proceso de una neurona en una red neuronal [22]

En la tabla podemos observar las distintas funciones de activación que se pueden añadir

**Tabla 4.1: Comparación de funciones de activación [22]**

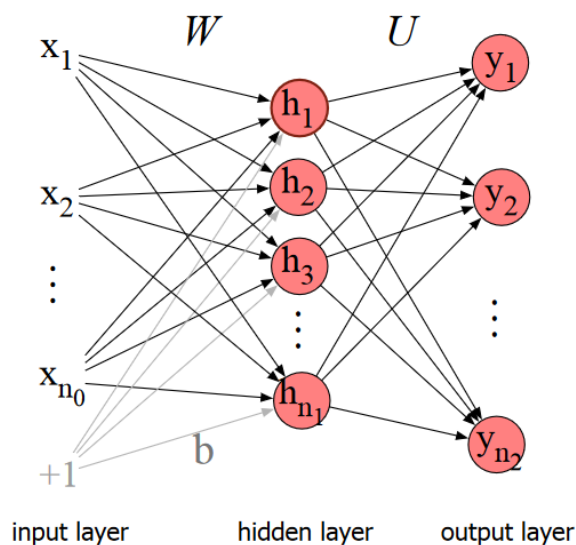
Sigmoid	Tanh	ReLU
Tiene la ventaja de mapear la salida en el rango de 0 a 1. Aunque no es usada en la práctica.	Devuelve un rango del (-1 a 1). Suele llevar los valores hacia la media.	La más simple y la más usada. Es z cuando z es positiva y 0 cuando no.
$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$	$y = \tanh(w \cdot x + b) = \frac{e^{w \cdot x + b} - e^{-(w \cdot x + b)}}{e^{w \cdot x + b} + e^{-(w \cdot x + b)}}$	$y = \text{ReLU}(w \cdot x + b) = \max(w \cdot x + b, 0)$
		
<b>Figura 4.2: Sigmoid Function</b>	<b>Figura 4.3: Tanh Function</b>	<b>Figura 4.4: ReLU Function</b>

En las funciones sigmoid o tanh, valores muy altos de z resultan en valores de y saturados o extremadamente cerca a 1, lo que da lugar a derivadas con resultado muy cercano a 0. Derivadas de 0 causan problemas durante el aprendizaje. La señal de error se va haciendo cada vez más pequeña hasta que ya no se puede usar durante el entrenamiento. Los ReLU no tienen este problema.

Lo realmente importante de las neuronas es su capacidad de poder unir gran cantidad de estas y crear una red neuronal (ANN). Una de las redes neuronales más simples y conocidas son las FNN.

#### 4.2.2. Feedforward Neuronal Networks (FNN)

Las Feedforward Neuronal Networks (FNN) son redes multicapas cuyas neuronas están conectadas sin ciclos, es decir, cada una de las salidas de las neuronas de cada capa son pasadas a neuronas de capas más altas (y ninguna salida pasa a capas más bajas). Estas redes se componen de tres capas: *input layer* (x), *hidden layer* (h) y *output layer* (y). En la imagen 4.5 se puede observar la forma de esta estructura.



**Figura 4.5: Estructura de una FNN [22]**

Cada neurona tiene su propio valor de *weights*, los cuales se recogen en un único vector ( $W = [w_1, w_2, w_3 \dots w_n]$ ), mientras que la *bias* ( $b$ ) es única para todas las capas. De forma que la ecuación 4.1 ahora quedaría con la forma de la ecuación 4.2 (suponiendo que usamos ReLU como función de activación)

$$h = ReLU \left( \sum_{i=1}^{n_0} W_{ji} \cdot x_i + b_j \right) \quad (4.2)$$

La *hidden layer* tiene el objetivo de representar el input, mientras que la *output layer* deberá calcular el resultado final. El resultado final será un vector con las probabilidades de distribución de todos los nodos de salida.

Para obtener este vector será necesario multiplicar los valores obtenidos en la capa anterior (*hidden layer*) por una nueva matriz de *weights* (conocida como  $U$ ). Como necesitamos un vector de probabilidades utilizaremos la función softmax para normalizar los valores reales y obtener un resultado multinomial (si quisiésemos únicamente una clasificación binaria usaríamos sigmoid). Véase la ecuación 4.3

$$z = softmax(Uh) = \frac{e^{z_i}}{\sum_{j=1}^d e^{z_j}} \quad 1 \leq i \leq d \quad (4.3)$$

### 4.2.3. Entrenamiento

Las redes neuronales FNN son entrenadas en un proceso supervisado, en el cual conocemos el output  $y$  correcto para una observación  $x$ . La red generará una predicción o estimación conocida como  $\hat{y}$ .

El objetivo del entrenamiento es que la red neuronal aprenda los parámetros  $W$  y  $b$  para cada capa

de forma que se obtenga una predicción  $\hat{y}$  lo más cercana posible a  $y$ .

Los pasos clave de este proceso, que incluyen en los transformers y LLM (que veremos más adelante), son la Forward Propagation y la Backpropagation.

1. **Propagación hacia adelante (Forward Propagation).** En este paso, se introducen los datos de entrada  $x$  en la red, y se calcula la salida a través de las capas de la red utilizando las ecuaciones descritas anteriormente 4.2 y 4.3. Esto implica calcular las *activations* de las unidades en cada capa, comenzando desde la capa de entrada hasta la capa de salida. Es el proceso que venimos explicando hasta ahora.
2. **Función de pérdida (Loss Function).** Se evalúa la predicción devuelta. Comparamos la predicción  $\hat{y}$  con el valor real  $y$ . Esta función mide la distancia entre ambos valores.
3. **Propagación hacia atrás (Backpropagation).** La retropropagación es el proceso de ajustar los *weights* y las *bias* de la red neuronal para minimizar la función de pérdida. Esto se hace calculando el gradiente de la pérdida con respecto a cada parámetro de la red y actualizando los parámetros en la dirección opuesta al gradiente para reducir la pérdida.

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} \quad b \leftarrow b - \eta \frac{\partial L}{\partial b} \quad (4.4)$$

Donde  $\eta$  es la tasa de aprendizaje, un hiperparámetro que controla el tamaño de los pasos.

4. **Optimización mediante Stochastic Gradient Descent (SGD).** SGD es uno de los algoritmos más comunes para optimizar los *weights*. Este actualiza los parámetros utilizando un solo ejemplo de entrenamiento a la vez. Los pasos seguidos son:

- I Inicialización. Inicializamos aleatoriamente los *weights* y *bias* del modelo.
- II Selección. Se escoge aleatoriamente un pequeño conjunto de datos del entrenamiento.
- III Cálculo. Calculamos el gradiente de la función de pérdida respecto a los parámetros del modelo utilizando el ejemplo seleccionado. Usamos la función 4.5

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t; x_i, y_i) \quad (4.5)$$

- IV Actualización de parámetros. Usando el gradiente calculado previamente, actualizamos los *weights* y *bias* del modelo en la dirección opuesta al gradiente (Backpropagation)
- V Repetición. Estos pasos se repiten iterativamente para cada ejemplo en el conjunto de datos de entrenamiento hasta que se alcanza un número predeterminado de iteraciones o hasta que se alcance un valor esperado en la función de pérdida.

Hay variantes del SGD que superan algunas de las limitaciones del SGD. Por ejemplo, una muy usada es el ADAM (Adaptive Moment Estimation), el cual permite que la tasa de aprendizaje varíe durante el entrenamiento.



#### 4.2.4. RNNs y LSTMs

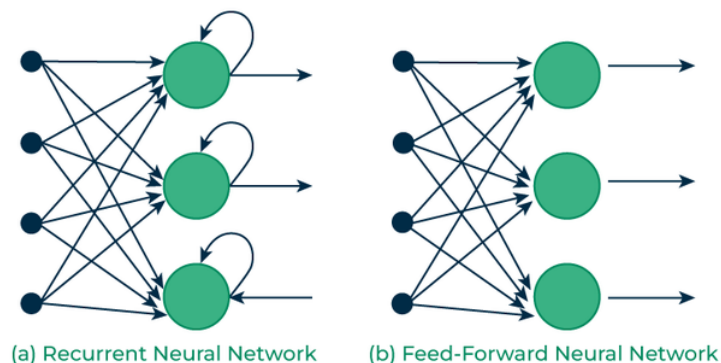
El problema que existía hasta ese momento era la falta de contexto. Con las FNN no había forma de recordar los datos que ya habían pasado. Si se pretendía emular el lenguaje humano era muy necesario que los modelos fuesen capaces de recordar grandes cantidades de contexto, tal y como nosotros hacemos en el día a día. Las primeras soluciones a este problema fueron dadas por las RNNs y las LSTMs

Vamos a realizar un pequeño repaso de los conceptos generales de estas dos arquitecturas, RNN y LSTMs, para poder comprender mejor los modelos LLM.

##### 4.2.4.1. RNNs

Una Recurrent Neural Network (RNN) es una clase de red neuronal que es cíclica en sus conexiones, lo que significa que el valor de una neurona dependerá directa o indirectamente de los outputs anteriores. Es decir, a diferencia de las redes neuronales descritas hasta ahora, cuyas entradas son independientes, en las RNN las conexiones forman un ciclo, permitiendo así que la información persista y no se deseche.

En la siguiente imagen 5.2 podemos observar esta diferencia. Vemos como en el caso de las RNNs las salidas se dirigen de nuevo hacia la misma neurona y afectará a la siguiente entrada de datos.



**Figura 4.6: Comparación estructuras RNN y FNN [26]**

Lo que se hace es recuperar el estado anterior e introducirlo de nuevo en el modelo. De forma que la nueva ecuación (4.6) ahora tendrá la forma

$$h_t = g(Uh_{t-1} + Wx_t) \quad (4.6)$$

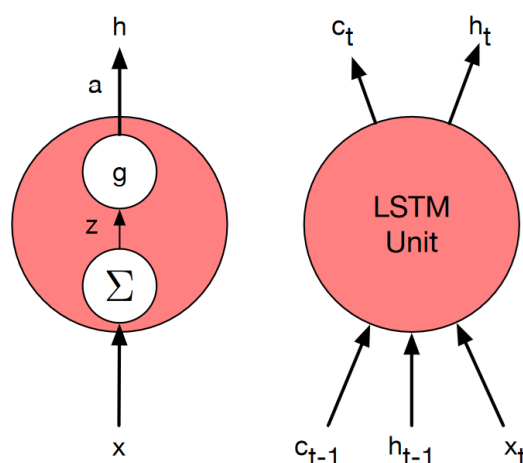
Sin embargo, las RNNs tienen una serie de problemas. Primero la computación es lenta para secuencias largas. Segundo, los gradientes van desapareciendo con el tiempo a causa de la multiplicación repetida de pequeños valores durante la backpropagation, lo que dificulta el aprendizaje de información a largo plazo.

#### 4.2.4.2. LSTMs

Una red Long short-term memory (LSTM) es una extensión de las RNNs que realiza dos acciones: eliminar información no necesitada del contexto y añadir información que pueda ser necesitada más tarde para tomar una decisión. De esta forma se elimina el problema del desvanecimiento del gradiente que comentábamos previamente.

Esta nueva estructura introduce mecanismos de puertas (o *gates*) que controlan el flujo de información. Las tres puertas principales son: la puerta de entrada, la puerta de ‘olvido’ y la puerta de salida.

La forma más simple de visualizar esta red es fijándonos en sus propias neuronas. Estas contienen tres entradas: el hidden state anterior ( $h_{t-1}$ ), el contexto previo ( $c_{t-1}$ ) y el input actual ( $x$ ). En la salida obtendremos el hidden state actual ( $h_t$ ) y el contexto actual ( $c_t$ ), los cuales serán usados para la siguiente entrada ( $x_{t+1}$ ). Véase la imagen 4.7 para una mejor comprensión.



**Figura 4.7: Comparación entre una neurona básica y una neurona de una red LSTM [22]**

No voy a dedicar mucho tiempo a esta parte por no ser tan relevante al proyecto. Solo comentar que la solución aportada por las LSTMs siguieron sin ser suficientes, pues aún tenía problemas con secuencias muy largas a causa de su procesamiento secuencial.

Una vez ya tenemos un pequeño contexto de como funcionan las redes neuronales básicas podemos ahora comprender y utilizar redes más complejas como los *Transformers*.



## Capítulo 5

# Transformers y LLMs

En 2017 se publicó un artículo que cambiaría por completo la forma de entender los modelos de procesamiento de texto y daría lugar a los modelos LLMs que tanto usamos en la actualidad. Los transformers fueron introducidos en el artículo ‘Attention is All You Need [27]’ por Vaswani. El impacto producido por los transformers ha permitido grandes mejoras en diferentes campos como: traducción, generación de texto y tareas de comprensión de texto [22].

A diferencia de los modelos vistos hasta ahora, que procesaban las secuencias de datos secuencialmente, los transformers utilizan un novedoso mecanismo de atención que permite procesar todas las palabras de una secuencia de texto en paralelo. De esta forma se superaban los problemas que suponían las redes neuronales RNN o LSTM.

Vamos a comentar a estudiar detalladamente en qué consiste la arquitectura propuesta.

### 5.1. Arquitectura de los Transformers

La arquitectura de los Transformers se compone de dos bloques principales: el codificador (*encoder*) y el decodificador (*decoder*). Ambos bloques trabajan conjuntamente para poder generar nuevas secuencias de texto, aunque cada uno tiene un objetivo final [28].

La idea principal de los Transformers es que, a través de una serie de capas se construyan representaciones contextualizadas cada vez más ricas del significado de los tokens de entrada. En cada capa del transformer, para calcular la representación de una palabra  $i$ , combinamos información de su representación y de la capa anterior con información de las representaciones de las palabras vecinas. El objetivo es producir una representación contextualizada para cada palabra en cada posición.

En la imagen 5.1 podemos observar la arquitectura de los Transformers. A continuación se procederá a explicar detalladamente paso a paso

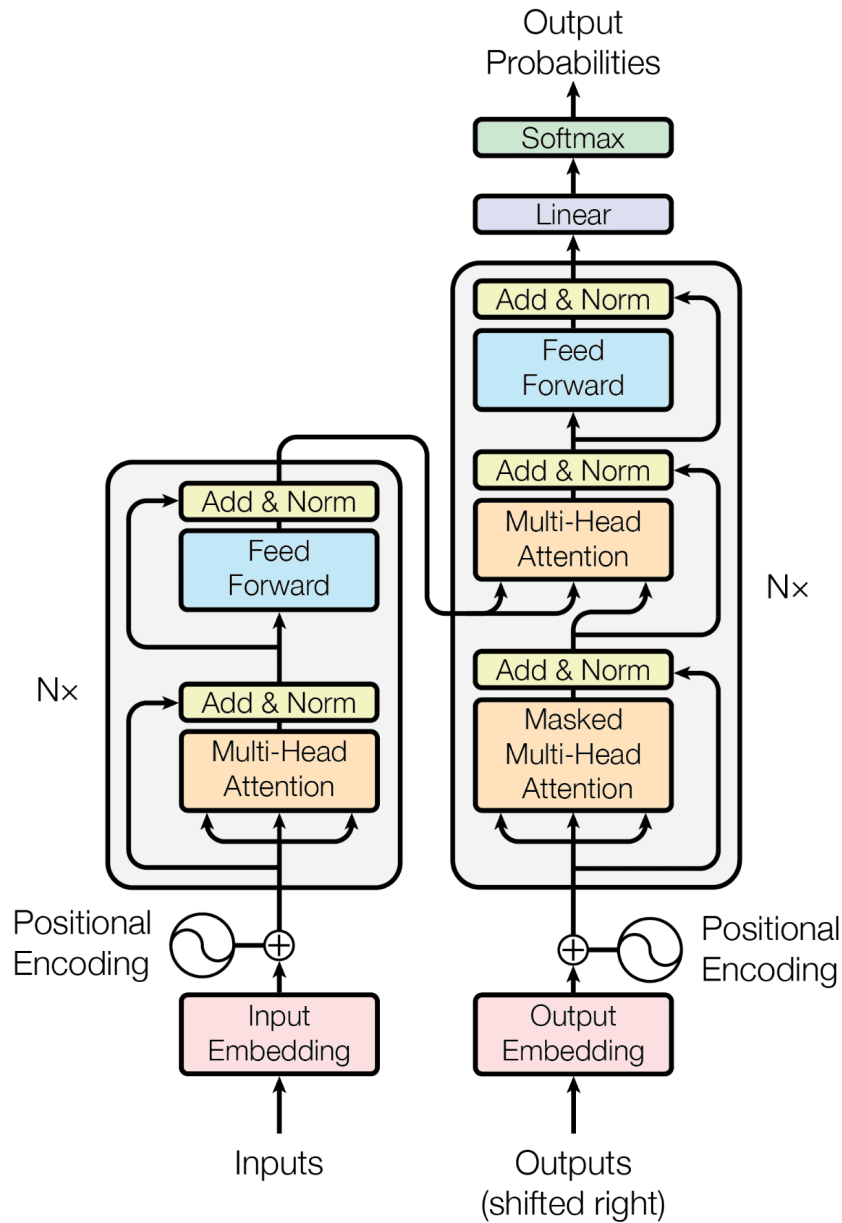


Figura 5.1: Arquitectura Transformers [27]

### 5.1.1. Input

La entrada al modelo se compone de frases tokenizadas. El objetivo de este proceso es convertir el texto en números (conocidos como Inputs IDs). El Input ID se obtiene a partir de cada palabra en el vocabulario y es fijo e irremplazable

Una vez tenemos las palabras representadas por números, se mapean en un vector de tamaño 512. Este vector es conocido como los *embeddings*. El valor dado no será fijo, sino que durante el entrenamiento el modelo irá cambiando este *embedding* de forma que represente correctamente

el significado de la palabra. Hay distintos métodos para generar estos embeddings, siendo el más conocido: *Word2Vec*

La lógica detrás de los embeddings consiste en que palabras que tengan significados parecidos o que tengan algún tipo de relación, tendrán valores más próximos que palabras que sean antónimas o que no tengan ninguna relación.

Por ejemplo, la palabra ‘cámara’ podría tener un ID con valor 3687 y su embedding correspondiente estaría formado por un vector de tamaño 512x1

Al modelo no solo se le introducirá el texto, sino que además se le debe añadir información sobre su posición en la frase. De esta forma el modelo tratará a cada palabra de una manera u otra según esté más cerca o más lejos de esta y se tendrá en cuenta la posición de cada palabra dentro de la frase. Este positional encoding, al igual que los embeddings anteriores, puede ser aprendido por el propio modelo durante el entrenamiento.

### 5.1.2. Encoder

El encoder es el encargado de transformar la entrada de texto en una representación contextualizada que capture la información relevante de la secuencia de entrada. Se utiliza principalmente para tareas de clasificación. Se compone de: el positional encoding, multihead attention, layer normalization y feed forward.

Esta estructura descrita se puede repetir tantas veces como sea necesario. Por ejemplo, los modelos T5 y GPT3-small utilizan 12 capas, mientras que GPT3-large usa 96 capas.

El primer paso es la capa de *Self-Attention*. Esta contiene el mecanismo que nos permite mirar ampliamente en el contexto y utilizar las palabras del contexto de una capa anterior para construir la representación de nuevas palabras en la capa actual. Es decir, estamos permitiendo al modelo la capacidad de relacionar palabras entre sí.

La *attention* es la idea de comparar un item de interes con una colección de otros *items* de manera que revele su relevancia en el contexto actual.

Para saber cuanta atención se le debe aplicar a cada palabra en la secuencia se utilizan tres matrices: Q, K y V las cuales poseen cada una una matriz de *weights* propia:

- Query (Q). Foco de atención actual. Es comparado con los inputs previos
- Key (K). Input previo que está siendo comparado con Q.
- Value (V). Usado para calcular la salida del foco de atención actual.

Gracias a estas tres matrices podemos representar una single-head attention con la ecuación (5.1):

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5.1)$$

Podemos entender este procedimiento más fácilmente con el siguiente gráfico:

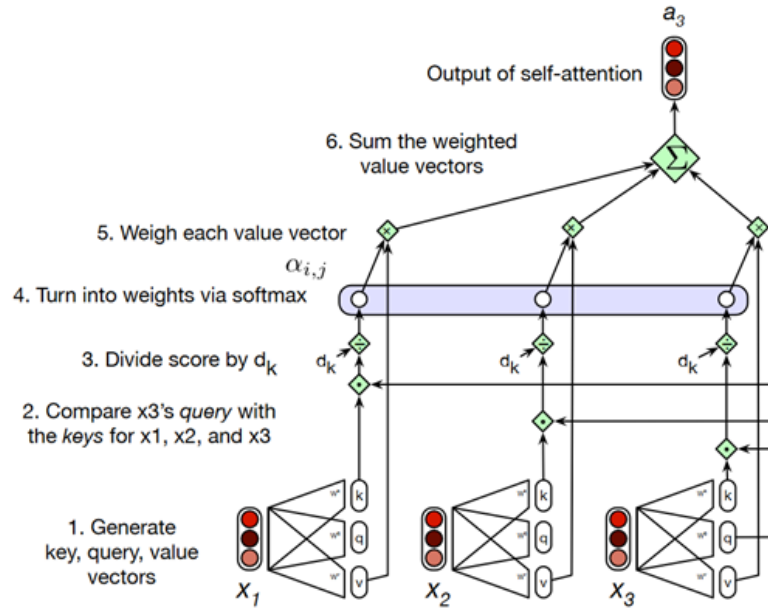


Figura 5.2: Estructura de una *single-head attention* [22]

Puesto que sería complicado para un único modelo self-attention aprender los diferentes tipos de relaciones entre sus inputs, los Transformers solucionan este problema con *multihead self-attention layers*. Estas son capas (o *heads*) paralelas con parámetros propios que permiten que el modelo pueda aprender diferentes aspectos de las relaciones entre los distintos inputs.

De esta forma obtenemos la ecuación final de las *multihead self-attention layers* (imagen 5.3), que no es más que la concatenación de las distintas capas de atención.

$$MultiHead(Q, K, V) = Concat(head_1 \dots head_h)W^0 \tag{5.2}$$

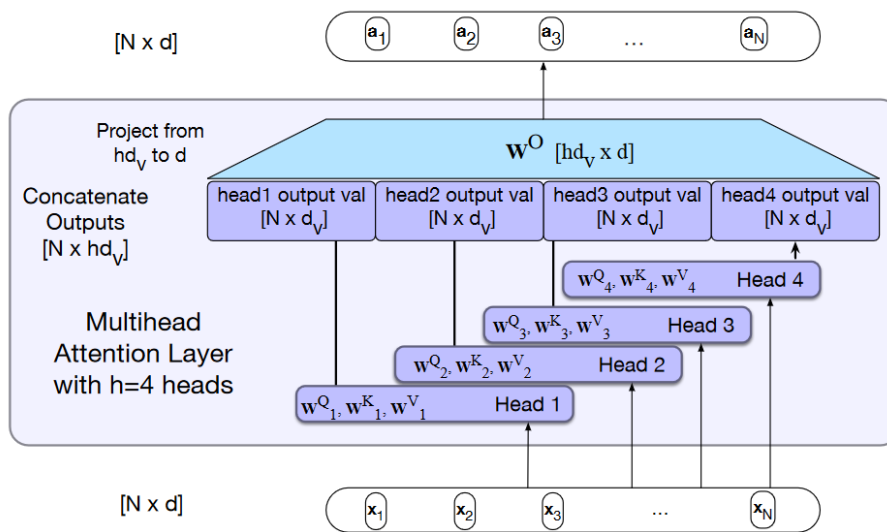


Figura 5.3: Estructura de una *multi-head attention* [22]

A continuación se hacen uso de las residual connections, las cuales son conexiones que pasan información desde una capa baja a una capa alta sin ir a través de capas intermediarias. Al permitir que la información del *activation* vaya hacia delante y el gradiente vaya hacia atrás estamos permitiendo que capas altas accedan directamente a la información de capas bajas.

Por último, se realiza la tarea de la Layer Normalization. Esta normaliza los vectores obtenidos hasta ahora. De esta forma mejoramos la eficiencia del entrenamiento manteniendo a su vez los valores de la *hidden layer* en un rango que facilite un entrenamiento basado en gradientes.

La entrada a esta capa es un único vector el cual, al pasar por ella, se obtiene un vector normalizado. Para ello se debe calcular la media  $\mu$  y la desviación estándar  $\sigma$  sobre los elementos del vector. A continuación se normalizan dando como resultado un nuevo vector con media 0 y desviación estándar de 1.

$$\hat{x} = \frac{x - \mu}{\sigma} \quad (5.3)$$

Finalmente, en la implementación estándar del *layer normalization* se introducen dos parámetros que serán aprendidos por el modelo:  $\gamma$  y  $\beta$ , que representan la ganancia y el offset respectivamente. :

$$\text{LayerNorm} = \gamma \hat{x} + \beta \quad (5.4)$$

### 5.1.3. Decoder

El decoder desempeña un papel fundamental en la generación de texto en lenguaje natural y en la traducción de textos. Este tiene la tarea de generar una secuencia de salida basada en el contexto proporcionado por una secuencia de entrada.

Su arquitectura se compone de casi los mismos elementos pero se diferencia por la utilización de una *masked multi-head attention*.

La capa Masked Self-Attention es la encargada de prestar atención únicamente a las palabras que hayan sido generadas previamente. De esta forma se evita el acceso a información futura durante la generación autoregresiva.

Lo que queremos evitar es que el mecanismo de atención comparta ningún tipo de información sobre cual es la palabra siguiente a predecir. El modelo debe predecir las palabras sin conocer cual es la palabra real.

### 5.1.4. Output

Una vez la secuencia de salida ha pasado por todas las capas del decoder, se llevan a cabo los últimos pasos en la arquitectura del transformer para obtener la salida. Estos se encargan de convertir los vectores de números en palabras. Los dos últimos pasos son: linear layer y softmax.

La *linear layer* es una red neuronal que proyecta el vector producido por el decoder en un vector mucho mas grande conocido como *logit*. Este tendrá el tamaño del Vocabulario que haya aprendido el modelo.



Por último, el objetivo de la softmax layer es coger los logits obtenidos en la capa anterior y convertirlo en probabilidades. El valor con la mayor probabilidad es la escogida y será la palabra asociada la que será generada.

## 5.2. Estrategias de Muestreo

El método más comúnmente utilizado para generar texto en modelos basados en lenguaje natural es el muestreo (*sampling*). Consiste en seleccionar palabras aleatorias de acuerdo con una probabilidad asignada. En cada paso de la generación, muestreamos palabras basadas en la probabilidad condicionada en nuestras elecciones previas.

Algunos métodos para el muestreo son:

- **Top-k Sampling.** En lugar de elegir la palabra más probable, seleccionamos un conjunto de las k palabras más probables. Después las renormalizamos para obtener una nueva distribución de probabilidad y posteriormente muestreamos aleatoriamente estas k palabras según sus probabilidades renormalizadas.
- **Top-p Sampling.** Se seleccionan las palabras que acumulen una probabilidad total de al menos 'p'. Se trunca la distribución para eliminar las palabras más improbables (p). Calculando la probabilidad en vez de el número de palabras conseguimos que la medida sea más robusta en distintos contextos, incrementando y decrementando dinámicamente las palabras candidatas.
- **Temperature sampling.** Se ajusta la temperatura de la distribución de probabilidad antes de muestrear la próxima palabra. Para una temperatura baja, incrementamos la probabilidad de las palabras más probables y reducimos las probabilidades de las palabras raras. Para una temperatura alta hacemos lo contrario.

## 5.3. LLM

Los Large Language Model (LLM) son una categoría específica de modelos basados en Transformers que están entrenados para entender y generar lenguaje natural a gran escala.

Estos modelos son capaces de aprender patrones complejos en el texto y generar respuestas coherentes y relevantes a partir de preguntas o prompts específicos. Algunos ejemplo de LLMs son: GPT (Generative Pre-trained Transformer) desarrollado por OpenAI o BERT (Bidirectional Encoder Representations from Transformers) desarrollado por Google.

Tras la salida de BERT, surgieron muchos otros modelos que mantuvieron este nombre, como: RoBERTa, ALBERT, DistilBERT... He aquí de donde cogí la inspiración para llamar a mi modelo BERTlangua.

## 5.4. Fine-tuning

El Fine-tuning es el proceso por el cual creamos una interfaz de un modelo LLM entrenado con un gran dataset para que sea capaz de realizar una tarea más concreta. El objetivo es conseguir optimizar el modelo original de tal forma que el modelo pueda realizar una tarea eficazmente sin necesidad de re-entrenar el modelo.

Este es un proceso de *transfer-learning*, pues lo que estamos haciendo es transferir el conocimiento aprendido por el modelo a partir de su conjunto de datos original a nuestro propio conjunto de datos. Esto nos permite aprovechar modelos gigantescos que han sido entrenados por grandes empresas como Google o Meta, adaptando sus capacidades hacia una tarea más específica.

En la imagen (5.4) se puede comprender este proceso de forma clara. Observamos como el modelo original (una CNN de detección de imágenes) podía distinguir entre diversos tipos de vehículos. Tras el fine-tuning, el nuevo modelo está especializado en distinguir si una imagen es un ‘coche’ o un ‘camión’, de forma que será mucho más efectivo en este segundo caso.

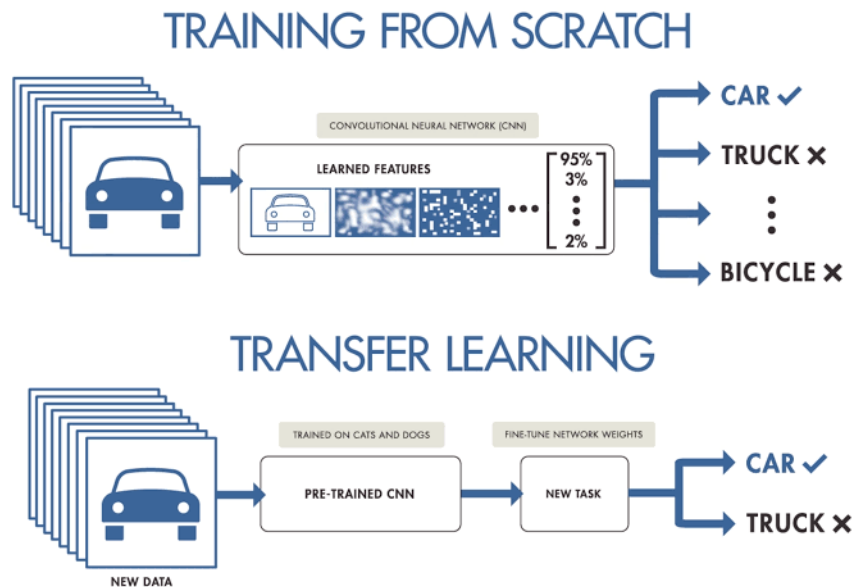


Figura 5.4: Ejemplo de Finetuning [28]

Esta es la parte que realmente nos interesa, pues vamos a coger un modelo pre-existente ya entrenado y vamos a especializarlo en la tare de escribir guiones con el estilo del director de cine Luis G. Berlanga.

Para ello, el modelo se inicializa con los *weights* aprendidos durante el proceso de entrenamiento descritos anteriormente. Durante el fine-tuning, el modelo es adaptado a la tarea especifica usando los *weights* de la capa de salida de forma que se generen palabras y frases que sean características del vocabulario y estructura del conjunto de datos del entrenamiento.

Este proceso se consigue ‘congelando’ las capas cuyos *weights* no queremos que se modifiquen durante nuestro entrenamiento. Es crucial que gran parte del modelo permanezca intacta y que solo sean las capas más superficiales en las que introduzcamos los cambios necesarios.

El fine-tuning se realiza durante varias épocas, recorriendo todo el conjunto de datos de entrenamiento varias veces para que el modelo aprenda de manera adecuada. Además, se suelen utilizar ejemplos que no forman parte del conjunto de entrenamiento para evaluar el rendimiento del modelo. De esta forma, el modelo mejora iterativamente al aproximarse a los ejemplos de evaluación en cada época. En nuestro caso estos ejemplos son escenas con las que no ha sido entrenado.

Hay diversos hiperparámetros que se pueden modificar a la hora de realizar este proceso y que afectan al entrenamiento, como la tasa de aprendizaje, el tamaño del contexto o el número de épocas entre otros. Estos hiperparámetros serán tratados más detalladamente en el siguiente apartado.

## Capítulo 6

# Entrenamiento del modelo

Este proyecto comenzó en marzo de 2023, por lo que los modelos de LLM no estaban tan avanzados como lo están actualmente. Durante los primeros entrenamientos me topé con grandes problemas por no usar modelos lo suficientemente grandes para poder obtener resultados aceptables. Fue el caso de ChatGPT 2 o Llama 2, dos modelos OS con los cuales invertí grandes cantidades de tiempo infructuosamente.

Sin embargo, durante estos últimos meses los modelos LLM OS han mejorado exponencialmente con la aparición de Mistral o Llama 3. Es por ello que a continuación me voy a centrar principalmente en el proceso y los resultados obtenidos con estos dos modelos, aunque simplemente quería remarcar que este proyecto ha tenido muchos más pasos e intentos de los que se van a reflejar a continuación.

Además, un detalle bastante importante que ha caracterizado este proyecto ha sido la falta de recursos económicos. Entrenar un modelo de estas características requiere de GPUs muy potentes de las cuales carezco. Es por ello que ha sido necesario depender de la contratación de GPUs en la nube, en concreto la herramienta de Google: Colab. Aunque gracias a esta herramienta he podido entrenar los modelos de Mistral y Llama 3, solo lo he podido hacer con sus versiones más reducidas, con 7B y 8B de parámetros respectivamente. Estos modelos tienen versiones mucho más grandes y complejas, pero que no puedo permitirme entrenar debido a lo costoso de las GPUs en el mercado actual.

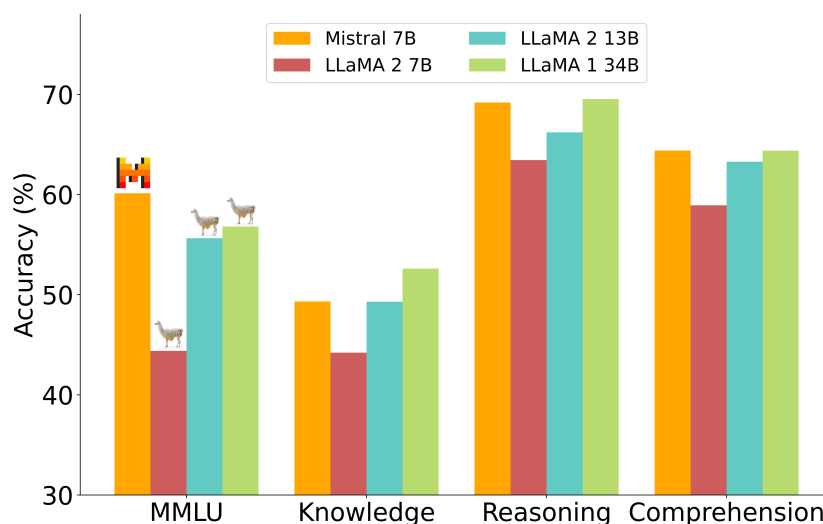
Es por esto que, aunque estos dos modelos han generado guiones aceptables, decidí hacer un finetuning de ChatGPT 3.5, un modelo de 175B de parámetros, con el cual se explotan todas las últimas novedades de los modelos LLM y podemos obtener resultados mucho mejores como referencia a que se podría obtener con un presupuesto mucho mayor.

### 6.1. Mistral 7B

Con la aparición en septiembre de 2023 de Mistral 7B, mi proyecto viró en esta dirección. Mistral 7B es un modelo open-source creado por una startup francesa: Mistral AI fundada en 2023. Lo novedoso del modelo fue su capacidad de ponerse a la altura del modelo creado por META, Llama 2, y su potencial para competir con las grandes corporaciones tecnológicas [29].

Como se puede observar en la imagen 3.5 el modelo superó al modelo reinante hasta el momento,

LLaMa2, en diferentes benchmarks. Por ejemplo, MMLU es un test de referencia que mediante ciertas métricas es capaz de comprobar la efectividad de un modelo, para así poder compararlo con otros modelos LLM [30].



**Figura 6.1: Eficiencia Mistra 7B con respecto a los modelos Llama en diferentes benchmarks [29]**

Lo realmente sorprendente fue que un modelo tan pequeño puede alcanzar e incluso mejorar a otros que duplican o incluso cuaduplican su tamaño. Esto se consiguió gracias a tres mejoras en la arquitectura:

1. **Sliding Window Attention.** Permite ampliar la distancia de acceso a tokens lejanos, ampliando aún más el contexto. En lugar de restringirse a una ventana estática de tokens, tal y como se ha explicado en el apartado anterior, este nuevo enfoque permite que se pueda atender a estados de capas anterior dentro de un rango específico, lo que amplía significativamente el acceso a tokens lejanos.
2. **Rolling Buffer Cache:** Esta técnica emplea una memoria caché de tamaño fijo para limitar el tamaño de la caché. Se consigue sobrescribiendo las posiciones de caché anterior y deteniendo así un crecimiento desmesurado. De esta forma se puede reducir la memoria caché hasta 8 veces su tamaño normal sin comprometer la calidad del modelo.

Una vez ya tenemos claro el funcionamiento de este modelo podemos comenzar a describir paso a paso como ha sido el proceso de entrenamiento.

### 6.1.1. Preparación de la DB

Todos los modelos han sido entrenados con un formato de prompt específico y es por tanto preferible mantener este formato para no confundir innecesariamente al modelo.

Los prompts se componen de 3 roles: ‘SYSTEM’, que es la información que recibe el modelo sobre como se debe comportar ante cualquier petición (V.g. Eres BERTlanga, una IA capaz de generar guiones con el estilo de Luis G. Berlanga); el ‘INPUT’, que es la petición que se le hace (V.g. Genera un guion sobre un Alcalde que...); y el ‘OUTPUT’, que es lo que se espera que el guion genere de forma autónoma.

Además, se le añaden dos Tokens especiales. Todo el prompt estará envuelto con los tokens <s> y </s>, tokens conocidos como bos\_token y eos\_token respectivamente. Estos tokens distinguirán los distintos prompts, pues el modelo recibirá todos los prompts a la vez durante el entrenamiento, son estos tokens lo que le hace darse cuenta de que son distintos.

El otro token especial es [INST] e [\INST], los cuales envuelven a la parte del ‘SYSTEM’ y la del ‘INPUT’. De esta forma se le indica claramente al modelo que parte es la que se le dará de normal y que parte es la que tiene que generar él.

En definitiva este sería un prompt de entrenamiento cualquiera del modelo (El prompt es mucho más largo, pero lo realmente importante es la estructura, por lo que se ha eliminado parte del texto):

```
'<s>[INST]Eres BERTlanga, una IA de generación de texto capaz de imitar el
estilo del director de cine español Luis García Berlanga.
Dado el nombre de la película, el número de escena y un pequeño resumen
de la escena escribe la escena con menos de 500 palabras. Utiliza
únicamente ubicaciones y diálogos. Película: La Boutique. Escena: 17 de
29. Resumen: En el banco, Ricardo intenta resolver el pago de una letra
con el banquero...[\INST]
BANCO. INT. DÍA
RICARDO: Señor, por favor, ¿para pagar una letra?...</s>'
```

### 6.1.2. Cargar modelo base

El modelo se carga gracias a la interfaz de huggingface, la cual permite mediante su paquete Transformers cargar estos modelos de forma sencilla a partir del ID del modelo base.

El modelo se carga con una cuantización de 4bits. La cuantificación es el proceso utilizado para reducir la precisión numérica de los parámetros del modelo. Esto ayuda reducir los requisitos de memoria y mejorar la eficiencia de la GPU. Esto es muy útil en mi caso, pues dispongo de unos recursos limitados para entrenar estos modelos.

El proceso de cuantificación implica representar los valores de los parámetros con menos bits de los que normalmente se usarían en punto flotante de precisión completa, lo que conlleva cierta pérdida de precisión, pero también reduce significativamente la carga computacional y de almacenamiento. Es una práctica muy utilizada en el ámbito de fine-tuning.

El código de este apartado se puede consultar en el anexo (A.7)

### 6.1.3. Tokenización

En este paso cargamos el Tokenizador. Lo más útil es cargar el tokenizador propio del modelo. Aunque como se comentará más adelante, en algunos entrenamientos introduce tokens especiales

nuevos que no forman parte del Tokenizador inicial, es en este paso donde se introducirían nuevos tokens especiales en caso de ser necesario.

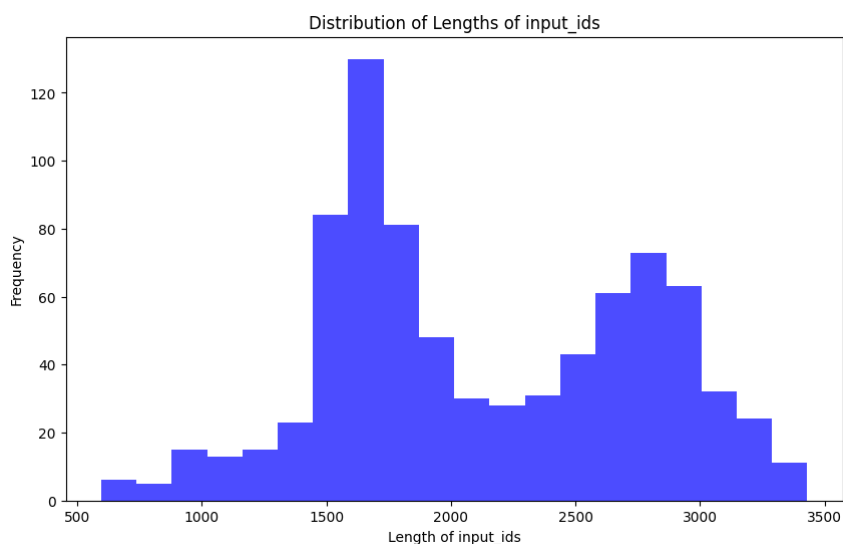
En este paso también se activan los tokens a utilizar, en este caso es necesario indicar que los tokens añadidos previamente (eos\_token y bos\_token) deberán estar funcionales.

El código de este apartado se puede consultar en el anexo (A.8)

#### 6.1.4. Elección de la longitud máxima

Es un paso crucial indicarle correctamente al modelo a partir de que número de tokens puede realizar el truncado del prompt. En nuestro caso es un punto muy delicado pues es necesario que el modelo sea capaz de procesar todas las escenas por completo para que luego sea capaz de generar una escena de principio a fin.

En la imagen 6.2 se puede observar la distribución de longitud de secuencias de nuestro Dataset.



**Figura 6.2: Distribución longitud de los inputs\_ids del Dataset**

Vista la gráfica, lo lógico sería escoger el valor 3.400 para poder así alcanzar todo el dataset sin truncarlo. Sin embargo, esto exigía muchos recursos de los cuales carecía y tuve que reducirlo a 3000. Este nuevo valor, aunque inferior, seguía representando bastante bien la gran mayoría de los casos, por lo que no considero que esta reducción haya causado ningún efecto relevante.

Una vez seleccionado este valor será necesario volver a tokenizar el dataset para asegurarnos de que cumplen esta condición. El código de este apartado se puede consultar en el anexo (A.9).

#### 6.1.5. LoRA

Dentro de las diferentes técnicas diseñadas para hacer el finetuning de modelos grandes más eficiente, conocido como Parameter-Efficient Fine-Tuning (PEPFT), destaca LoRA. Low-Rank

Adaption (LoRA) es una herramienta que permite realizar fine-tuning de manera eficiente, rápida y consumiendo poca memoria, de manera que no haya necesidad de utilizar GPU muy potentes.

En vez de realizar el fine-tuning de todos los *weights* del modelo pre-entrenado, solo se realiza sobre dos pequeñas matrices de las capas superiores, tal y como se explicó en el apartado de Fine-tuning. De esta forma entrenamos los *weights* del modelo que tienen un impacto más grande sobre la tarea a realizar. Esto contrasta con otros métodos previos que utilizaban muchos más *weights* de forma innecesaria durante el entrenamiento.

LoRA permite que de los cerca 3 mil millones de parámetros que contiene Mistral, sean entrenables 8 millones, es decir, el 2.27 %.

A la hora de cargar el modelo con LoRA hay dos valores importantes, *alpha* y *r*:

- **r**. Rango de matrices de bajo rango que se utilizan. Cuanto menor sea menos parámetros podremos entrenar. En el paper oficial se indica que un valor correcto para buenas generalizaciones es de 64, en nuestro caso pondremos  $r = 32$  para que se centre más en nuestros datos.
- **alpha**. Factor de escala que ajusta la influencia de las matrices de actualización en el modelo final. Para obtener mejores resultados, se ha seleccionado un valor de  $alpha = 64$

El código de este apartado se puede consultar en el anexo (A.10)

### 6.1.6. Entrenamiento

Una vez ya tenemos el modelo preparado para ser entrenado falta seleccionar los últimos parámetros de entrenamiento. Los parámetros más importantes son:

- **Learning Rate**. Tasa de aprendizaje. Controla la magnitud de las actualizaciones de los *weights* del modelo en cada paso. Cuanto más grandes más grande será la corrección, por lo que puede complicar la tarea de encontrar el valor de *weight* precisa. En nuestro caso usaremos como valor 0.000025
- **Tipo de estrategia**. Los modelos se pueden entrenar mediante dos estrategias: épocas o pasos. Las épocas indican el número de veces que el modelo recorrerá todo el dataset. Los pasos indicarán cuantos pasos tienen que dar hasta terminar, sin importar que haya alcanzado el final del dataset. En nuestro caso es mucho más interesante que el modelo sea entrenado varias veces con todo el dataset, por lo que se ha escogido las épocas.
- **Optimización**. Se escoge el optimizador a utilizar. Usaremos el Adam, que ya ha sido introducido previamente.
- **Warmup steps**. Es una técnica que permite que el Learning Rate aumente gradualmente desde cero hasta su valor inicial al comienzo del entrenamiento para evitar posibles inestabilidades.

Conforme se inicializa el entrenamiento comenzaremos a visualizar el proceso en cada paso, indicando el tiempo estimado restante. Además se muestra el Training Loss y el Validation Loss obtenidos en cada época, tal y como se observa en la imagen 6.3



Epoch	Training Loss	Validation Loss
1	1.437600	1.385009
2	1.231800	1.265195
3	1.053900	1.188823
4	0.887800	1.159533

Figura 6.3: Captura del resultado del entrenamiento de Mistral7B

Como se puede observar, en la última época se obtuvieron resultados en la Training Loss inferiores a 1. Es preferible no utilizar pérdidas menores a 1. Así que puesto que en cada época se realiza un checkpoint donde se guardan los parámetros entrenados se puede utilizar el modelo tal y como quedó en la época 3 sin problema alguno.

El código de este apartado se puede consultar en el anexo (A.11)

### 6.1.7. Testing

Por último, ya solo queda probar en el modelo. En este paso se realizan peticiones al modelo con la misma estructura con la que creamos el dataset con el que fue entrenado el modelo. Una vez devuelto los resultados se comienza un proceso de reflexión sobre los distintos puntos de mejora que se le pueden aplicar.

La generación del modelo está condicionada por los valores que se explicaron anteriormente: Temperature, top\_p y top\_k. Los mejores resultados se han obtenido con una Temperatura de 0.5 a 0.7, un top\_p de 50, un top\_k de 0.95.

En el anexo (A.11) se puede consultar el simple código necesario para cargar el modelo almacenado y ejecutarlo.

Sin embargo, para la generación de texto he preferido utilizar la herramienta llamada ‘Text generation web UI’, la cual utiliza Gradio para crear una interfaz gráfica que permita de forma cómoda modificar estos hiperparámetros y probar el modelo en tiempo real. En la imagen 6.4 se observa la gran cantidad de parámetros personalizables que permite.

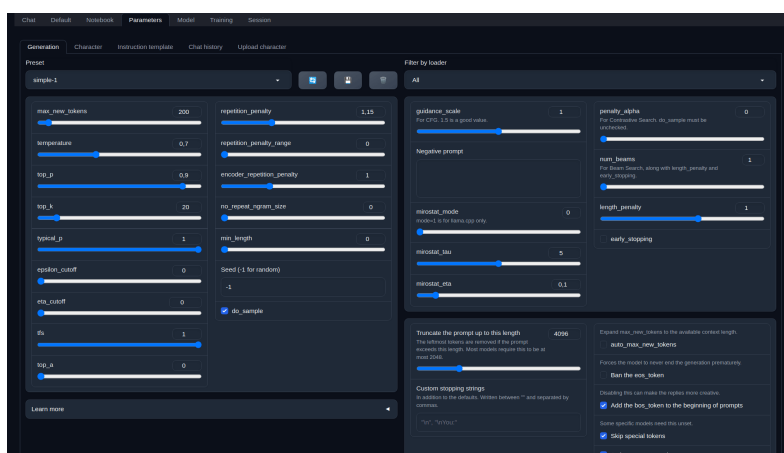


Figura 6.4: Herramienta ‘Text Generation Web UI’ para LLMs

## 6.2. LLaMa 3

Cuando ya estaba preparando la memoria de este proyecto me encontré con la salida del nuevo modelo de META: Llama 3 8B. Puesto que ya tenía todo la DB preparada no dudé en probar este nuevo modelo.

Llama-3 es la tercera versión del modelo LLM lanzado por META AI a finales de abril de 2024. Esta nueva versión prometía superar a Mistral 7B y a Gemma 7B (Modelo de Google), tal y como se puede observar en la imagen (6.5) con diferentes benchmarks [31].

Meta Llama 3 Instruct model performance

	Meta Llama 3 8B	Gemma 7B - It Measured	Mistral 7B Instruct Measured		Meta Llama 3 70B	Gemini Pro 1.5 Published	Claude 3 Sonnet Published
MMLU 5-shot	68.4	53.3	58.4	MMLU 5-shot	82.0	81.9	79.0
GPQA 0-shot	34.2	21.4	26.3	GPQA 0-shot	39.5	41.5 CoT	38.5 CoT
HumanEval 0-shot	62.2	30.5	36.6	HumanEval 0-shot	81.7	71.9	73.0
GSM-8K 8-shot, CoT	79.6	30.6	39.9	GSM-8K 8-shot, CoT	93.0	91.7 11-shot	92.3 0-shot
MATH 4-shot, CoT	30.0	12.2	11.0	MATH 4-shot, CoT	50.4	58.5 Minerva prompt	40.5

**Figura 6.5: Comparación Benchmarks de Llama3 con otros modelos [31]**

Hay gran cantidad de mejoras sobre su versión anterior, Llama 2, entre ellas destaca: capacidad de utilizar hasta 30 idiomas (aunque idiomas distintos al inglés solo forman un 5% de la base de datos de entrenamiento), un contexto de hasta 8912 tokens (el doble que Llama 2) y entrenado con 15T de tokens (Llama 2 había sido entrenada con 2T). Además hay tres versiones disponibles, con 8B, 70B y 400B de parámetros.

El proceso de entrenamiento es muy parecido al descrito en Mistral 7B. Los únicos cambios significativos son en la estructura del Dataset y la utilización de unsloth durante la cuantificación.

### 6.2.1. Preparación de la DB

El formato del dataset de entrenamiento de Llama 3 es conocido como ‘Alpaca’. Este tiene los mismos campos que Mistral 7B: ‘SYSTEM’, ‘INPUT’ y ‘OUTPUT’, aunque ahora renombrado como: ‘INSTRUCTION’, ‘INPUT’, ‘RESPONSE’. Además se le añade una frase inicial que es básica para cualquier entrenamiento y se le añade únicamente el token de finalización (eos\_token)

De forma que el resultado final de un prompt cualquiera se vería de la siguiente forma:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response in spanish that appropriately completes the request.

### Instruction:

Eres BERTlangua, una IA de generación de texto capaz de imitar el estilo del director de cine español Luis García Berlanga.

### Input:

Dado el nombre de la película, el número de escena y un pequeño resumen de la escena escribe la escena con menos de 500 palabras. Utiliza únicamente ubicaciones y diálogos. Película: *La Boutique*. Escena: 17 de 29. Resumen: En el banco, Ricardo intenta resolver el pago de una letra con el banquero...

### Response:

BANCO. INT. DÍA

RICARDO: Señor, por favor, ¿para pagar una letra?...</s>'

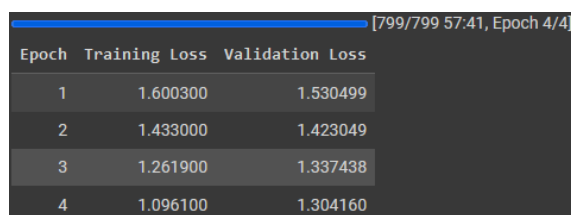
### 6.2.2. Cuantificación

Para este modelo se ha escogido directamente un modelo cuantificado, en concreto: llama-3-8b-bnb-4bit, de esta forma ya está preparado para ser utilizado. Se utiliza de esta forma la librería *unsloth*, la cual permite que el proceso de fine-tuning de los LLMs sean 2.2 veces más rápido y se un 80 % menos VRAM.

En el código adjunto en el anexo A.14 se puede observar que es muy parecido al formato de entrenamiento con Mistral 7B .

### 6.2.3. Entrenamiento

El proceso de entrenamiento de es igual que en Mistral 7B. Solo se puede señalar como la utilización de *unsloth* permite reducir bastante el tiempo de procesado. Para nuestro dataset, con 799 ejemplos, se ha conseguido reducir 30 minutos de ejecución, en comparación con el entrenamiento de Mistral 7B. Esta diferencia se puede apreciar comparando la imagen 6.6 con la imagen 6.3.



Epoch	Training Loss	Validation Loss
1	1.600300	1.530499
2	1.433000	1.423049
3	1.261900	1.337438
4	1.096100	1.304160

Figura 6.6: Distribución longitud de los inputs\_ ids del Dataset

Se puede consultar los códigos de todo el proceso de preparación y entrenamiento en los anexos: A.12, A.13, A.14 y A.15.

## 6.3. ChatGPT

ChatGPT es el conocido modelo de generación de texto creado por la empresa OpenAI. Es uno de los modelos más potentes en la actualidad, aunque cada vez más la competencia comienza a acercarse a las capacidades de ChatGPT. Sin embargo, de momento este modelo sigue liderando todas las benchmarks [32].

Decidí entrenar este modelo para poder comparar estos modelos previos con uno más grandes. Pues ChatGPT 3.5 consta de 175B de parámetros, 20 veces más que Llama8B y Mistral7B. Aún a pesar de que ya está disponible la versión de ChatGPT 4 esta aún no se puede utilizar para el finetuning, por lo que se ha utilizado la versión anterior (3.5). En la imagen (6.7) se puede observar la comparativa entre ambos modelos:

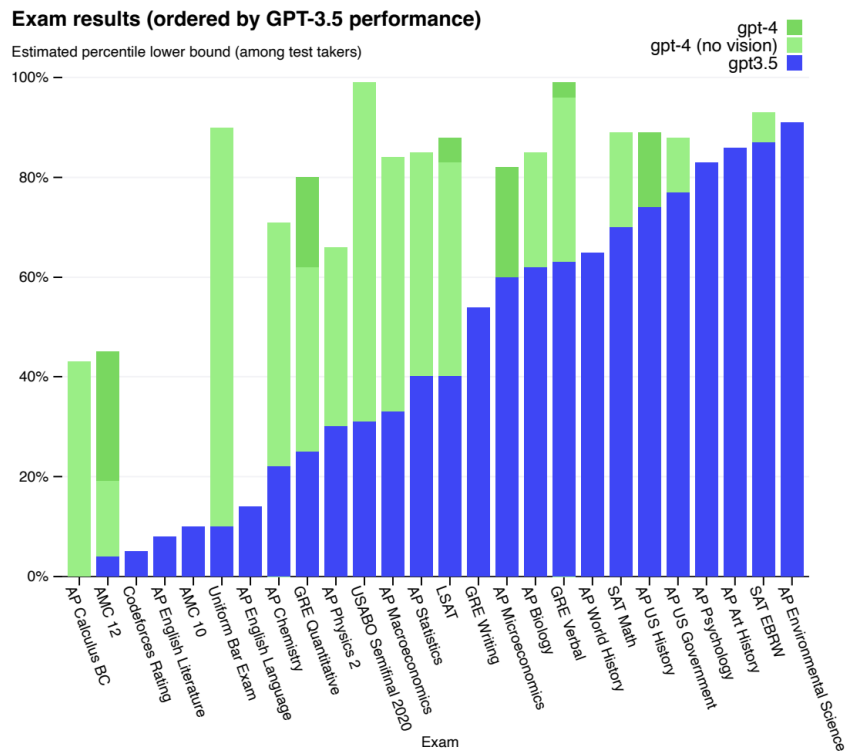
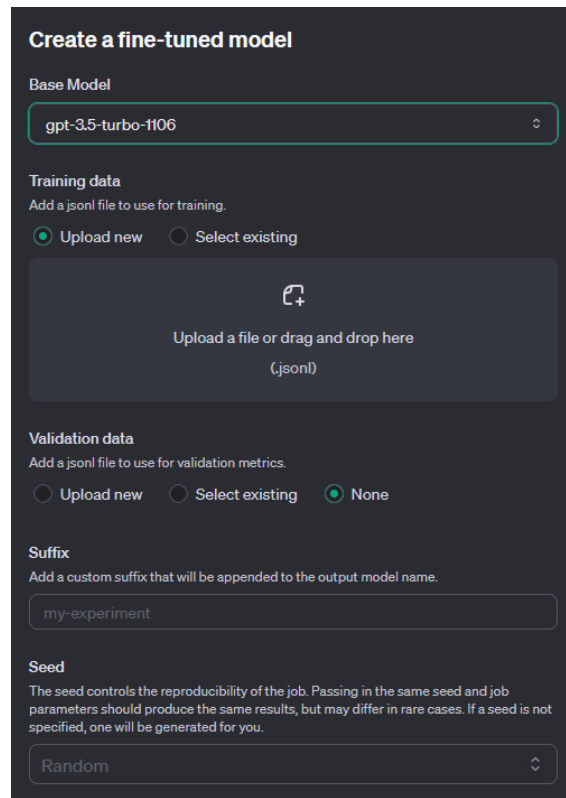


Figura 6.7: Comparativa entre ChatGPT 3.5 y ChatGPT4 en diferentes benchmarks [32]

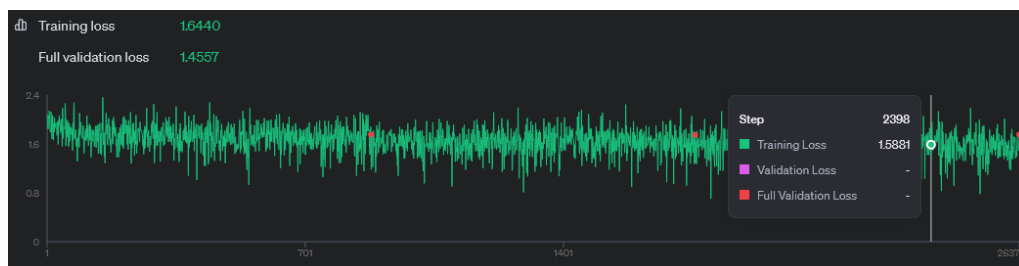
### 6.3.1. Entrenamiento

La diferencia fundamental de ChatGPT con respecto a los modelos previos es que se trata de una empresa privada y, por tanto, el entrenamiento se realiza en sus GPUs mediante el uso de una API. Por tanto, el proceso no es muy personalizable, ya que solo permite la introducción del DB y la selección del modelo a entrenar. En la imagen 6.8 se observa cómo de simple es el inicio del entrenamiento



**Figura 6.8: Proceso de Finetuning con ChatGPT 3.5**

El formato de la base de datos necesaria es muy similar al comentado hasta ahora. Esta contiene tres roles: ‘SYSTEM’, ‘USER’ y ‘ASSISTANT’ y se deben añadir todos los prompts en un único archivo con formato .jsonl. Una vez creado el proceso comenzará el entrenamiento y mostrará en tiempo real el proceso. En la imagen 6.9 se puede observar el resultado final de uno de mis entrenamientos, con todas las evoluciones de la Training Loss y la Validation Loss.



**Figura 6.9: Resultado Finetuning con ChatGPT 3.5**

## Capítulo 7

# Resultados

A continuación se van a exponer los resultados obtenidos tras el entrenamiento de cada uno de los modelos. Aunque previamente se va a realizar una exposición de la evolución de este proyecto, pues como es lógico, no se obtuvieron los resultados finales en el primer intento. Todo lo contrario, ha sido un proceso arduo de aprendizaje con numerosos intentos y errores que han conseguido que poco a poco fuese aprendiendo mucho sobre como funcionan las redes neuronales y que al final se pudiese construir lo que ha acabado siendo el modelo BERTlanga.

### 7.1. Evolución del proyecto y resolución de problemas

#### 7.1.1. Introducción

En el momento que comencé este proyecto aún no estaban disponibles los LLM con los que he entrenado el modelo definitivo, tal y como se ha explicado previamente. Eso dio lugar a numerosos intentos infructuosos que estuvieron simplemente marcados por la incapacidad del propio modelo a comprender algo tan complejo como el estilo cinematográfico de un director. Sin embargo, voy a explicar como ha sido este proceso paso a paso para que quede constancia de ello.

Téngase en cuenta que estos entrenamientos iniciales se realizaron sin haber terminado de digitalizar todos los guiones. La digitalización era un proceso muy largo y entre medias quería comenzar a probar como funcionaba el entrenamiento. No dudo que con todos los guiones digitalizados los siguientes ejemplos hubiesen sido algo mejor, pero lo importante es que son insuficientes para crear un guion coherente independientemente de la cantidad de texto con la que hayan sido entrenados.

Por último, los distintos resultados se van a mostrar en una tabla, en cuya primera fila se podrá consultar el modelo usado, el número de épocas y los hiperparámetros usados durante la generación. En la segunda fila se mostrará el Prompt que se ha introducido como contexto y en la tercera fila, con tipografía mecanografiada, se mostrará el resultado obtenido por el modelo.

#### 7.1.2. Primeros intentos

Los primeros intentos estuvieron marcados indudablemente por mi inexperiencia en el campo de la Inteligencia Artificial en general. Estos consistieron en entrenar el modelo con el texto sin más;

junté todos los guiones en un archivo .txt y se lo dí al modelo para que ‘aprendiese’. Es por esto que el modelo era incapaz de generar nada con sentido, ya que durante el entrenamiento no se le estaba dando un *prompt* para que cogiese como contexto. Se puede ver uno de los resultados en la tabla 7.1.

**Tabla 7.1: Entrenamiento de Llama2 con texto plano**

---

**MODELO:** Llama 2 7B. 10 Épocas. **Temperature:** 0.7, **Top p:** 0.9, **Top k:** 20.

---

El alcalde sale al balcón del ayuntamiento. Abajo, en la plaza, hay una multitud reunida esperando a que dé su discurso.

ALCALDE: ¡¡¡Vecinos!!! He de contaros algo muy importante...

---

EL RIVAL DEL ALCALDE se asoma tras él y le interrumpe. `Los enemigos ya están en movimiento! Han cruzado el Trencó Enel y han sido avistados en Villar del Campo!'

El Alcalde vuelve a mirar sus notas. `Pero escuchadme atentamente: Si victoriam gibamus, hacemos un gran paseíl. Pero si perdemos, os arrepienta eternally!'

El Alcalde levanta los ojos hacia los balcones de las casas y sigue escribiendo. `En cuanto a eso del paseíl, lo explica el marido de nuestra prima Juana. Un poco de paciencia, por favor... Digo que viene el ejército nacional. Porque, según me dicen, los de arriba.'

---

En este ejemplo anterior se pueden observar varias cosas. Primero se demuestra que el modelo Llama 2 7B estaba principalmente entrenado en inglés y apenas en español. Es por ello que en ocasiones escribe en inglés y comete faltas ortográficas en español. Segundo, vemos como aún siendo una escena tan pequeña no tiene ningún sentido, no parece ser capaz de crear una escena coherente porque no ha sido entrenado para ello. Por último, tampoco parece que comprenda del todo los tipos de textos. Es muy importante que el modelo entienda que cuando hay una palabra en mayúscula seguida de dos puntos significa que es un personaje hablando y solo en ese caso puede utilizar la primera o segunda persona. En este ejemplo observamos confunde e incluso ignora este formato.

Otro de los grandes problemas que tenía durante los primeros entrenamientos era forzar muchas épocas. Creía que a cuantas más épocas mejor comprendería el texto dado. Tras muchos entrenamientos he comprobado que con 3 épocas es más que suficiente.

Mi siguiente paso fue intentar forzar al modelo que comprendiera cuando es un diálogo y cuando es una escena. Para ello creé tokens especiales propios. Estos tokens me permitían marcar un texto como de un tipo u otro, en mi caso si se trataba de un diálogo, una ubicación o texto descriptivo de escena.

Generalmente estos tokens se suelen esconder durante la generación final para mejorar la legibilidad del texto. Sin embargo se puede forzar para que así poder ver si se están aplicando correctamente. Como se podrá comprobar en el siguiente ejemplo (tabla 7.2) no se consiguió hacer comprender al modelo las diferencias, aún a pesar de los reiterados intentos.

**Tabla 7.2: Entrenamiento de Llama2 con tokens especiales**


---

**MODELO:** Llama 2 7B. 10 Épocas. **Temperature:** 0.5, **Top p:** 0.95, **Top k:** 50.

---

El alcalde sale al balcón del ayuntamiento. Abajo, en la plaza, hay una multitud reunida esperando a que dé su discurso.

<LOCATION>DESPACHO AYUNTAMIENTO. DÍA. INTERIOR<LOCATION>

El alcalde se encuentra en su despacho cuando entra el señor Delegado General por la puerta.

<CHARACTER>DELEGADO<CHARACTER> : Buenos días señor Alcalde

<CHARACTER>ALCALDE<CHARACTER> : Buenos días... No le esperaba señor Delegado. ¿Qué desea?

---

<CHARACTER>DELEGADO<CHARACTER> : Me han enviado un telegrama. Me dicen que hay que discutir algo muy importante con usted. Se trata de una propuesta para la ciudad. Es del Ministerio. Y como siempre en estos casos, hay que estudiar bien las cosas antes de dar respuesta. Por eso he venido a verle. Para discutirla. ¿No estaría mejor en el Ayuntamiento? ¿Que dice usted?

El alcalde mira el telegrama y luego los documentos adjuntos.

Luego dice: Estamos en el ministerio, pero hay que estudiar la cosa seriamente. Hasta que no lo hagamos no podemos dar respuesta. Venga conmigo.

Se dirige hacia la salida e intenta pasar ante el secretario. Pero el secretario le detiene.

Secretario (off) ¡Señor Alcalde! No se puede ir así...

Alcalde (off) ¿Pero qué ocurre?

Secretario (off) El señor delegado general ha venido especialmente..

Alcalde (off) ¡Papá!

---

Hasta este momento se habían conseguido algunos avances, sobre todo si se compara con el primer ejemplo. A partir de muchas generaciones puede comprobar que el modelo era capaz de comprender el contexto histórico de la época, aunque en ocasiones lo confundía con conceptos actuales (como el COVID-19). Era capaz de realizar algunos diálogos simples, aunque no con el formato exigido. Por último, era capaz de continuar el prompt aportado de forma medianamente coherente. Aun así cada vez quedaba más patente que era necesario aportar algún tipo de contexto con el que se pudiese guiar el modelo.

### 7.1.3. Introducción de contexto

Fue en este momento cuando decidí aplicarle una lógica distinta, creando una estructura prompt/respuesta. El primer paso fue dividir todos los guiones en escenas. Una vez estaban divididas procedía a resumir todas las escenas con menos de 100 palabras. Empecé haciéndolo a mano pero pronto, tras la salida de Mistral 7B, comencé a utilizarlo para resumir los textos y corregir yo sus imperfecciones.

Una vez obtuve todos los resúmenes y escenas procedí a entrenar un nuevo modelo, esta vez ya con Mistral 7B. A este nuevo modelo le di en el prompt un rol nuevo: BERTlangua, pues es sabido que darle un nombre a las IA de generación de texto les felicita la tarea generar una personalidad propia.



Además, se le indicaba el nombre de la película (para darle un poco de contexto) y el resumen de la escena anterior, a excepción de la primera escena de cada película que no incluía este resumen. También descarté la idea de añadir tokens personalizados por comprobar que confundía más al modelo que ayudar y alargaba el proceso de entrenamiento. Analizemos la salida obtenida con este modelo (tabla 7.3).

Téngase en cuenta que el resumen de la escena anterior no se muestra completamente por ser demasiado largo. Lo realmente importante es el código generado.

**Tabla 7.3: Entrenamiento con contexto de la escena anterior**

---

**MODELO:** Mistral 7B. 5 Épocas. **Temperature:** 0.5, **Top p:** 0.95, **Top k:** 50.

---

<s> [INST] Eres BERTlanga, una IA de generación de texto capaz de imitar el estilo del director de cine español Luis García Berlanga. Debes escribir guiones de cine con el estilo de este director. Debes utilizar todo tipo de recurso: diálogos entre personajes, descripción de escenas, descripción de planos de cámara y descripción de ubicaciones.  
Dado el nombre de la película, el número de escena y un pequeño resumen de la escena anterior como contexto, genera la continuación de la escena con el estilo del director de cine Luis García Berlanga. Película: Esa Pareja Feliz. Subescena anterior: Juan llega a un patio destartalado que sirve como cochera y taller de reparaciones. Luis, vestido de soldado, atraviesa el lugar para saludar a Juan, tropezando por el camino con Manolo... [\INST]

---

PATIO - (Exterior-Día)

JUAN sale al patio, que es además la zona de garaje, de reparaciones y de todo tipo de almacenes de la casa de MANOLO. JUAN llega hasta donde MANOLO estaba parado, cruzándoles el paso a un TIPO que se dirige hacia él corriendo. Es LUIS, el personaje de los pantalones cortos y blanca camisa. El TIPO va saltando sobre piernas que le empiezan a mostrar. JUAN se queda perplejo. Ahora ve cómo de pronto las piernas desaparecen y aparece MANOLO encima de la parte trasera del taxi, que se ha convertido en la puesta portada del coche. MANOLO lleva su gorro en la mano.

MANOLO: ¿Tú te vas?

JUAN asiente y cruza el patio.

LUIS: Pero... pero... no me dejas ir sin mi gorro. Si no me fusilan...

Juan se detiene un momento, luego sigue hacia la puerta de entrada de la casa. Debe hacerse cargo de algo extraño, porque se ríe.

LUIS: Yo no he hecho nada, pero si me fusilan a mí, a ver quién se hace responsable...

Juan hace un gesto de impaciencia y entra en la casa. Al pasar frente a la escalerilla se encuentra con un TIPO que sube con un saco lleno de globos y molinillos.

PORTERO: ¿Hasta cuando espero?

La puerta se abre y JUAN sale.

---

No tardé en darme cuenta de mi error. El modelo no iba a continuar generando una nueva escena, sino que intentaba crear la escena que yo había descrito en el prompt.

Por eso decidí cambiar de nuevo la lógica. Esta vez subdividí manualmente cada escena en subescenas y en el prompt introduje el resumen de la escena actual en vez de la anterior. Al indicarle al guion en qué número de escena y subescena se encontraba, este era capaz de crear guiones de acuerdo con su posición con respecto a la película. Por ejemplo, si le indicabas que se encontraba en la escena 1, este comenzaba describiendo a los personajes. Por otro lado si le indicabas que estaba en la escena 26 de 26, el guion generaba la palabra 'FIN' al final.

De esta forma, el modelo comenzaba a crear guiones a partir de un pequeño resumen de la escena. Así el modelo ya comenzaba a crear escenas con el estilo de Berlanga, aunque con ciertas limitaciones. Por ejemplo, comprobé que era complicado juntar distintas escenas generadas para crear un guion único, pues al desconocer que había generado para la subescena anterior comenzaba a añadir cosas que se contradecían con su generación anterior. Además nunca parecía terminar de generar, le costaba mucho generar una escena con final.

Para solucionar este problema creé el que es el formato de entrenamiento que quedó como definitivo. Volví a subdividir todas las escenas en subescenas, pero esta vez con un máximo de 500 palabras por subescena. Esta condición se le indicará al modelo para que sepa que tiene que escribir hasta un máximo de 500 palabras. De esta forma obligamos al modelo a terminar las escenas y no quedarse a medias. Esta nueva subdivisión supuso volver a generar un resumen para cada subescena.

También detecté que en ocasiones inventaba personajes que no tenía sentido dentro del guion. Por ello decidí indicar en el prompt los personajes que quiero que hablen en la escena.

Se introdujo además una gran novedad. Hasta ahora no podías indicarle al modelo qué tipo de documento se deseaba obtener. Para un mismo prompts en distintas generaciones obtenías guiones de solo diálogos, luego otro con descripciones de cámara... Ahora se le indicaba en el prompt de entrenamiento de qué tipo de documento se trataba, para que a la hora de probar el modelo tuviésemos acceso a un LLM tan versátil que te pueda generar únicamente los diálogos o, si lo prefieres, todo tipo de descripciones de cámara y ubicaciones. Esta modificación la añadí inicialmente en el SYSTEM prompt, pero descubrí que es mejor no añadir grandes complejidades en este apartado, es mejor añadir los detalles concretos en la parte del ASSISTANT.

En resumen, en el nuevo prompt se introduce: el nombre de la película, el tipo de documento que se va a generar (solo diálogos, con escenas...) el número de escena, el número de subescena, los personajes que hablan, la escena anterior por completo y un resumen de la escena actual.

Al juntar los 17 guiones que se ha entrenado el modelo y subdividirlo en subescenas de 500 palabras se obtuvieron 898 prompts de entrenamiento en total. De los cuales se escogieron 1 o 2 de cada película para separarlos en el database de validación.

Los resultados obtenidos con este nuevo y último formato de prompt desarrollado será analizado en el respectivo apartado de cada modelo. En los siguientes apartados se van a describir algunas de las escenas de las películas y vamos a analizar los resultados.

## 7.2. Análisis de los resultados de Mistral 7B

Para comprobar la eficiencia de este modelo vamos a intentar describir una escena de ‘Los Jueves, Milagro’. En esta película, un grupo de Fuerzas Vivas de un pequeño pueblo decide fingir la aparición de ‘San Dimas’ con el objetivo de atraer turismo gracias a este supuesto milagro. En concreto, en esta escena uno de los personajes intenta engañar a los feligreses de un iglesia mediante el reflejo de unas luces. El resultado se puede leer en la tabla 7.4

**Tabla 7.4: Escena generada con Mistral7B entrenado**

<b>MODELO:</b> Mistral 7B. 3 Épocas. <b>Temperature:</b> 0.6, <b>Top p:</b> 0.95, <b>Top k:</b> 50.
[INST]Eres BERTlanga, una IA de generación de texto capaz de imitar el estilo del director de cine español Luis García Berlanga. Debes escribir guiones de cine con el estilo de este director. Utiliza menos de 500 palabras.
Dado el nombre de la película, el número de escena, los personajes y un pequeño resumen de la escena escribe la escena con el estilo del director de cine Luis García Berlanga con menos de 500 palabras. Utiliza: diálogos entre personajes, descripciones de escenas y descripción de ubicaciones. Película: ‘Los Jueves, Milagro’. Escena: 13 de 20. Subescena: 1 de 2. Personajes: ‘DON FIDEL, DON MANUEL, DON RAMON, DON EVARISTO, MARTINO, MUJERES’. Resumen: En la iglesia de Fontecilla, durante la misa dominical, Don Fidel y los feligreses quedan asombrados al presenciar un resplandor inexplicable alrededor de la imagen de San Dimas. Este fenómeno crea un murmullo entre los congregados. Más tarde, Don Manuel se apresura a encontrar a Martino en el campanario de la iglesia, quien revela que el efecto luminoso fue creado usando un pequeño espejo para reflejar la luz. Sin embargo, el cura Don Fidel descubre el truco y se enfrenta a Martino. [INST]
IGLESIA DE FONTECILLA. INT. DÍA
Un resplandor extraño rodea ahora a la imagen de San Dimas. DON FIDEL, otro cura, DON SALVADOR, DON RAMON, DON MANUEL, don EVARISTO y los feligreses que llenan la iglesia, están asombrados. Se oye un murmullo general.
DON RAMON: ¡Miren! ¡Miren!
DON FIDEL: ¿Pero qué es eso?
Don EVARISTO ha salido de su banco y avanza con aire solemne hacia la imagen.
DON EVARISTO: ¡Un milagro! ¡Un milagro!
En este momento, DON MANUEL ve en lo alto de la Iglesia a MARTINO y le hace seña de bajar. MARTINO se echa la soga al hombro y empieza a bajar por la escalera del campanario.
DON EVARISTO se ha acercado ya a San Dimas, que sigue envuelto en aquel resplandor.
El murmullo crece de punto mientras Martino sale del coro y llega al fondo de la nave. Don Manuel se le acerca.
DON MANUEL: ¿Eres tú el del milagro?
Martino asiente con modestia.
MARTINO: Bueno... Sabe... Es muy sencillo... Un espejito... Nada más.

DON MANUEL: No me digas... ¿Cómo lo haces? Porque está teniendo mucho éxito...

Martino se viste un poco más de prestidigitador.

MARTINO: Planchas... Te lo aseguro, cualquiera podría hacerlo mejor que yo.

DON MANUEL: No me creo nada... Pero dime, ¿dónde está el espejito?

MARTINO y DON MANUEL continúan hablando disimuladamente. Abajo, junto al cura DON FIDEL, un grupo de mujeres comentan maravilladas:

MUJERES: ¡Una aparición...!

Muertas de miedo, se agarran y empiezan a rezar en voz baja:

MUJERES: Padrenuestro..

MUJERES: Ave María Purísima...

DON FIDEL sospecha de MARTINO y DON MANUEL y se acerca a ellos:

DON FIDEL: ¡Te vas a arrugar, canalla! ¡Te vas a arrugar!

Se abre paso hasta llegar ante él, mientras exclama:

DON FIDEL: ¡Conque te vas a quedar aquí, eh?!

MARTINO: Pero, DON MANUEL... ¿qué le pasa?

DON FIDEL: Usted, usted tiene que marcharse.

MARTINO: No se ponga así DON FIDEL, que no había visto nunca un milagro y yo creía que iba a poder ver uno gratis.

DON FIDEL: ¡Fuera de aquí!

---

Mistral 7B consigue hasta cierto punto arañar la superficie del estilo berlanguiano. Vemos en los diálogos como los personajes se desenvuelven de manera correcta, aunque en ocasiones dicen frases que no llegan a tener mucho sentido. Lo mismo pasa con las descripciones de escenas, que aunque se comprenden, son un poco confusas en algunos momentos. El mayor problema de este modelo es que ha sido entrenado mayormente en inglés, por lo que se nota, a la hora de escoger ciertas palabras, que no controla tanto el idioma español.

He decidido mostrar una escena sin descripción de planos porque no parece ser capaz de indicar los planos de forma correcta, las descripciones de cámara de este modelo suelen ser un poco confusas. También parece en ocasiones liarse mucho con el prompt y no seguir el contexto dado. Está claro que un modelo tan pequeño no puede seguir perfectamente un prompt tan complejo como el propuesto.

Aún así, para ser un modelo tan pequeño se consiguen grandes avances e incluso se consigue generar alguna que otra frase que encuentro realmente berlanguiana. Por ejemplo, en este mismo texto uno de los personajes indica que nunca había visto un milagro y quería ver uno gratis. Podría parecer una frase inocente, pero realmente es de lo que va la película. La película critica los falsos milagros que se crearon durante los años 50 en algunos pueblos para atraer turismo y en esta frase se indica como el personaje quiere ver un milagro gratis, pues en la época no era raro que cuando querías acercarte a ver el lugar donde ocurrió el milagro se te cobraba por algún ungüento u objeto milagroso.

Como conclusión, este modelo no parece capaz de generar guiones complejos. Le cuesta crear escenas largas, sobre todo cuanto más se aleja del prompt más comienza a alucinar y a olvidar el prompt inicial. Aún así, en varias ocasiones me ha conseguido sorprender con diálogos ingeniosos.

---

### 7.3. Análisis de los resultados de Llama3 8B

Llama3 es el modelo, de los pequeños, que mejor ha respondido, acercándose bastante al estilo de Luis G. Berlanga. Para visualizarlo mejor voy a generar una escena de la película ‘Bienvenido Mr. Marshall’

La película trata de como los americanos van a ir a un pueblo de la Mancha para dar dinero por el Plan Marshall. El pueblo se prepara para su llegada. En concreto, hay un conjunto de escenas donde varios personajes sueñan con lo que van a hacer los americanos cuando lleguen. Uno de estos personajes es DON LUIS, un noble cuyos antepasados colonizaron América y lamenta la pérdida de las colonias españolas. En su sueño teme que los americanos se venguen, creyendo que estos no son más que indios en taparrabos. A continuación se muestra la escena reinventada por el modelo (tabla 7.5). (He añadido en el anexo la tabla B.1 la escena original de la película, para poder realizar una comparativa si se desea.)

**Tabla 7.5: Escena generada con Llama8B entrenado**

<b>MODELO: Llama 3 8B. 3 Épocas. Temperature: 0.6, Top p: 0.95, Top k: 50.</b>
<p>### Instruction:</p> <p>Eres BERTlanga, una IA de generación de texto capaz de imitar el estilo del director de cine español Luis García Berlanga. Debes escribir guiones de cine con el estilo de este director. Utiliza menos de 500 palabras.</p> <p>### Input:</p> <p>Dado el nombre de la película, el número de escena, los personajes y un pequeño resumen de la escena escribe la escena con el estilo del director de cine Luis García Berlanga con menos de 500 palabras. Utiliza: diálogos entre personajes, descripciones de escenas, descripción de planos de cámara y descripción de ubicaciones. Película: 'Bienvenido Mr Marshall'. Escena: 9 de 21. Subescena: 1 de 1. Personajes: 'DON LUIS, NARRADOR y GASPAR DE ARCE ALONSO DE QUESADA'. Resumen: Don Luis intenta leer un libro en su sala, pero se duerme. Un narrador comenta que pronto se dormirá. Don Luis despierta brevemente, pero finalmente se queda dormido. En su sueño, se encuentra en una playa tropical del siglo XVI, donde desembarca y planta un estandarte. Luego, es capturado por indios y sumergido en un caldero hirviente. Despierta en su sala, asustado, y se da cuenta de que todo fue un sueño. Un narrador bromea sobre su aventura onírica y Don Luis se va a la cama, apagando la luz.</p> <p>### Response:</p> <p style="text-align: center;">SALA DE DON LUIS. INT. DÍA</p> <p>P.G. DON LUIS sentado en una butaca, el libro sobre las piernas. Fuma una pipa y relee un párrafo. La cámara se acerca hasta encuadrar en P.P. el libro que está leyendo: ‘El día de los trífidos’ de John Wyndham.</p> <p style="padding-left: 40px;">NARRADOR: (Off) Parece que DON LUIS se va a dormir.</p> <p>Luego DON LUIS se queda dormido y la pipa se apaga en el cenicero. La cámara se levanta y encuadra en P.P. el rostro de DON LUIS dormido</p>

PLAYA TROPICAL. EXT. DÍA

(En sueños) P.M.L. Plano general de una playa tropical del siglo XVI, con sus cocoteros, su mar azul... Entra DON LUIS por la derecha del cuadro, caminando bajo el sol resplandeciente; lleva un uniforme marcial del siglo XVI, con un sombrero de plumas semejante al de los mosqueteros, y una espada al cinto. Viene andando muy erguido y mirando hacia todas partes con aire altanero; cuando llega a un punto conveniente se detiene y echa mano a su espada; saca de la vaina un estandarte pequeño, lo ondea un poco y luego lo planta en tierra.

P.M.C. El estandarte plantado en la arena.

P.M.L. DON LUIS viendo su obra; sonrío satisfecho y luego adopta una actitud más marcial.

P.M.C. Aparece desde el fondo de la escena un INDIO, vestido como un indígena americano. El INDIO sonrío a DON LUIS.

INDIO: (Off) ¡Hola! ¡Hola!

P.M.C. DON LUIS saludando con la espada.

DON LUIS: ¡Hola! ¡Hola!

INDIO: Y... ¿Quién eres? ¿Qué haces por aquí?

DON LUIS: Me presento. Yo soy GASPARD DE ARCE ALONSO DE QUESADA y he venido a civilizaros...

P.M.C. INDIO sonrío confuso.

INDIO: Pero... No hace falta... Si aquí ya estamos todos civilizados...

P.M.C. DON LUIS queda sorprendido.

DON LUIS: Hombre... No creo...

P.M.C. INDIO sonrío todavía más.

INDIO: Ven conmigo...

Se vuelve sobre su camino y señala hacia un punto de adentro del bosque hacia donde se dirigen ambos

P.M.C. Al entrar en el bosque entran por la izquierda del cuadro un grupo de indios desnudos que llevan prisionero a DON LUIS; éste viene maniatado entre dos indios que le empujan bruscamente; vienen directos hacia la cámara; cuando llegan al P.T., paran y hacen señales a otros dos indios; éstos traen un gran caldero de barro cocido lleno de agua hirviendo; sujetan una cuerda atada al cuello de DON LUIS y le van sumergiendo en el agua hirviendo.

SALA DE DON LUIS . INT. DÍA

Abre los ojos y mira asustado a su alrededor. Se pone en pie sobresaltado y comprueba que todo ha sido un sueño. Está en su sala vestido con pijama contemporáneo. Se limpia el sudor de la frente con un pañuelo.

NARRADOR: (OFF) ¡Vaya! ¿Ha tenido usted alguna pesadilla?

DON LUIS se queda un poco agitado. DON LUIS sale del cuadro encogiéndose de hombros, como diciendo `qué cosas más raras me pasan'

NARRADOR: (OFF) Pero no importa, porque ya es hora de apagar la luz e irse a dormir...

---

Ya ente ejemplo vemos como se cumplen las proporciones de Planos y se usan correctamente, a diferencia de Mistral7B que comprobé que los añadía sin realmente haber relación con la escena.

Además comprende que el Narrador tiene que ser una voz en OFF. Utiliza correctamente las ubicaciones y realiza diálogos con un cierto tono humorístico e irónico.

Me parece muy curiosa la elección del libro que ha hecho. El libro que está leyendo no se le ha indicado en el prompt ni en la película se nombra tampoco. Sin embargo, aquí el modelo ha decidido que el libro sería ‘El día de los trífidos’. Esta es una novela del 1951. Puesto que la película se estrenó en 1952 su aparición no sería anacrónica. Además el libro trata de como una invasores llegan a la tierra y colonizan a los seres humanos, lo que se podría entender como un símil con la condición de explorador español del protagonista.

Como conclusión, me parece que Llama3 tiene mucho potencial de alcanzar satisfactoriamente el estilo deseado si se entrenase con un modelo mucho más potente como el Llama70B. Aún así, esta versión tan pequeña ya es capaz de generar escenas siguiendo el prompt de forma completa y con descripciones detalladas y correctas del entorno y los planos.

## 7.4. Análisis de los resultados de ChatGPT 3.5

Con el entrenamiento de ChatGPT quiero demostrar que realmente sí que es posible imitar el estilo de un escritor o director de cine mediante el uso de Inteligencia Artificial.

Los modelos anteriores, aunque han devuelto buenos resultados, siguen mostrando algunas carencias. Estas carencias no son más que debido a su pequeño tamaño. Estoy convencido que si hubiese tenido los recursos materiales y financieros para entrenar un modelo como LLama70b o Mixtral8x7B se hubiesen conseguido los resultados que voy a describir a continuación. ChatGPT no es más que una solución medianamente barata para paliar esta carencia.

Dicho esto, con ChatGPT se ha conseguido generar un guion entero. Este guion procede de la idea original de Berlanga llamada ‘El Autocar’. Fue un proyecto escrito junto con Rafael Azcona y Enrique Llovet en 1957 pero que nunca se llevó a cabo a causa de la censura franquista. En este documento se encuentran los resúmenes de lo que ocurre en cada escena a falta de ser desarrollados los diálogos, plano, ubicaciones y una mayor precisión en las descripciones. Se ha conseguido, de esta forma, rescribir una idea de solamente 13 páginas a un guion completo con descripciones de plano, de ubicaciones y diálogos completos entre personajes.

Debido a la larga extensión del guion generado no puedo mostrar el guion entero en este documento (aunque será adjuntado durante la entrega). Aún así muestro a continuación un pequeño extracto para que comentemos la calidad del mismo.

Antes de leer la escena debo dar un pequeño contexto. La película se ubica en un pequeño pueblo español. En este, las ”Damas Cruzadas”, son un grupo de solteras y monjas lideradas por la autoritaria Doña Gloria, quienes imponen su moralidad estricta, aterrorizando a los habitantes. Hastiados, el maestro y otros vecinos idean un plan para enviarlas en peregrinación a Lourdes, pagando entre todos un autocar.

La escena que se muestra a continuación (tabla 7.6) es la escena en la que las damas montan en el autobús en dirección hacia Lourdes mientras todo el pueblo se reúne ansioso ante su inminente ida. Antes de irse, Doña Gloria decide interrogar desde el autocar a algunas personas del pueblo para

asegurarse que no van a hacer ningún acto impuro durante su ausencia mientras que los interrogados mienten descaradamente para asegurarse de que se vayan. (Téngase en cuenta que el texto que se muestra es una subescena que es continuación de la anterior, de ahí el inicio a mitad)

**Tabla 7.6: Escena generada con ChatGPT 3.5 entrenado**

---

P.G. En último término DOÑA GLORIA y JUAN al pie del autocar.  
 DOÑA GLORIA: ... ¿Está todo?...

JUAN le sigue muy interesado por hacerse bien mirar.  
 JUAN: Ya está todo ... ¿Está bien así?  
 DOÑA GLORIA: Perfectamente...

JUAN se queda un poco preocupado porque todavía no ha cobrado el viaje.

P.G. DOÑA GLORIA sube al autocar después de haberse despedido con exagerados gestos hacia los que se quedan.

P.M. DOÑA GLORIA se asoma a una de las ventanillas del autocar.  
 DOÑA GLORIA: ... ¿Está todo dispuesto?... ¿Nada de impurezas?... ¿Nada de vicios?... ¿Nada de corrupción?...

P.M. DOÑA GLORIA se dirige a una mujer  
 DOÑA GLORIA: ... ¡Su hija!... ¡No tiene medias puestas!...  
 ¿No las tiene?...

MADRE: ... No, señora... Se rompieron y todavía no he tenido dinero para comprarle otras.  
 DOÑA GLORIA: ... Pues cómpreselas... Y si no se las compra, usted será la responsable de que su hija se vaya a condenar.

La MADRE se lleva las manos a la boca. DOÑA GLORIA se dirige al MAESTRO:  
 DOÑA GLORIA: ¿Y usted?... ¿Todavía sigue usted con esa manía de los griegos?... ¡Si nos hicieran caso...!

P.M. El MAESTRO se ha quedado muy serio y muy formal.  
 MAESTRO: ... Para mí han muerto los griegos... Por inmorales... Yo no les perdono ni el juramento hipocrático...

P.M.L. Hacia el Alcalde  
 DOÑA GLORIA: ... ¿Y usted señor ALCALDE?  
 ALCALDE: ... ¡Yo sigo prohibiendo el baile agarrado!  
 DOÑA GLORIA: ... ¡Bien! ¡Muy bien!  
 ALCALDE: ... ¡Y si me apuran prohibiré también el baile suelto!

El TONTO DEL PUEBLO se acerca a DOÑA GLORIA y le coge un poco de la falda.  
 TONTO DEL PUEBLO: ... Señora presidenta... ¿Puedo tirarle piedras a los forasteros que vengan al río?  
 DOÑA GLORIA: ¡Sí, hijo! ¡Sí! ¡Pero mucho cuidado con herirlos! ¡Que luego vienen las denuncias!

El TONTO DEL PUEBLO, muy contento, corre hacia el río.  
 DOÑA GLORIA: ... ¡Hasta luego!... ¡Dentro de diez días estaré aquí!... ¡Y quiero encontrar todo tan limpio como hoy!...  
 ¡Nada de perversión!... ¡Y nada de bailar!... ¡Eh!...

---



P.G. El autocar arranca y se aleja. Mientras toda la gente reunida en la plaza se despide con los brazos en alto.

P.G.L. El autocar alejándose por una carretera polvorienta. A cada lado hay campos labrados. Al fondo se recorta la silueta del pueblo.

Aquí ya podemos distinguir sin duda un estilo muy claro. Los planos son introducidos de forma lógica y correcta. Los diálogos representan muy bien a esa forma que tienen los personajes berlanguianos de hablar y además el modelo es capaz de añadir ideas nuevas que no estaban en el prompt aportado de forma consecuente con la trama.

Recordemos que al principio de este proyecto se realizó un análisis minucioso de que características tiene el estilo de este director de cine. Por ello, vamos a comparar los resultados obtenidos y averigüemos si realmente se ha conseguido el objetivo buscado.

## 7.5. ‘El Autocar’: Análisis y comparación de la obra de BERTlangua

Vamos a repasar paso a paso las mismas características que analizamos previamente pero ahora con el nuevo guion generado por el modelo LLM BERTlangua. La única característica que no se va a comprobar es la de los plano secuencias, pues no se puede comprobar únicamente con texto.

Es necesario apuntar que un estilo artístico humano no es replicable tan solo por seguir tres simples reglas. El estilo berlanguiano es algo mucho más complejo que lo que puede parecer a continuación. Este apartado solo sirve para confirmar si el guion creado sigue las características más básicas y conocidas de este director.

### 7.5.1. Coralidad

Si realizamos la misma lógica que aplicamos inicialmente, consideraremos que para que un personaje sea considerado como relevante este tiene que tener al menos 10 líneas de diálogo. En el guion generado se obtiene que hay 15 personajes con 10 líneas de diálogo, por lo que se cumpliría esta característica que ha sido demostrada previamente. En la imagen 7.1 se puede comprobar esta característica en comparación con todas las películas.

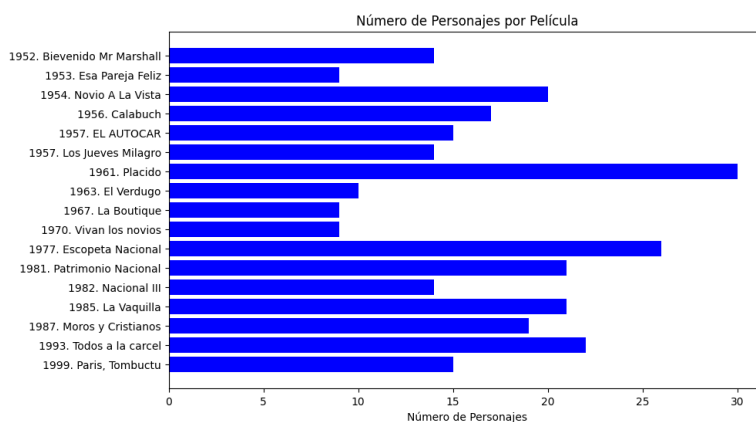
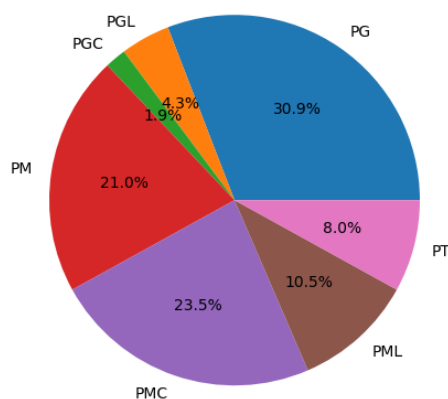


Figura 7.1: Porcentaje de planos usados en el guion ‘El Autocar’

### 7.5.2. Encuadre

Si generamos una gráfica circular con los planos usados obtenemos la imagen 7.2



**Figura 7.2:** Número de personajes de las películas de Berlanga junto con en el nuevo guion 'El Autocar'

Al compararlo con la gráfica (2.14) obtenida con el análisis de los guiones de Berlanga se puede apreciar unos porcentajes muy parecidos. Vemos que los porcentajes de PG y PM son muy parecidos. La mayor diferencia sería con una superior importancia de los planos PMC, esto se debe a que gran parte del guion los personajes se encuentran dentro de un Autocar, por lo que es normal que se utilicen planos más cortos.

De esta forma podemos concluir que se ha cumplido satisfactoriamente la proporción de planos usados.

### 7.5.3. Vocabulario

En la imagen 7.3 se puede comprobar el Wordcloud realizado para el guion generado.



**Figura 7.3:** WordCloud guion 'El Autocar'

El guion fue escrito poco después de ‘Bienvenido Mr Marshall’ y es por ello tiene una temática y desarrollo muy parecido. El modelo ha sido capaz de entender la relación de este guion y ha generado de forma reiterada numerosas palabras relacionadas con esa película iniciales. Son el caso de las palabras: ‘pueblo’, ‘plaza’, ‘pobre’, o ‘pareja’ entre otras.

#### 7.5.4. Análisis de conceptos

De la misma forma podemos incluir este guion dentro de la gráfica de evolución de conceptos y observar si cuadra con la tendencia seguida en la época (imagen 7.4).

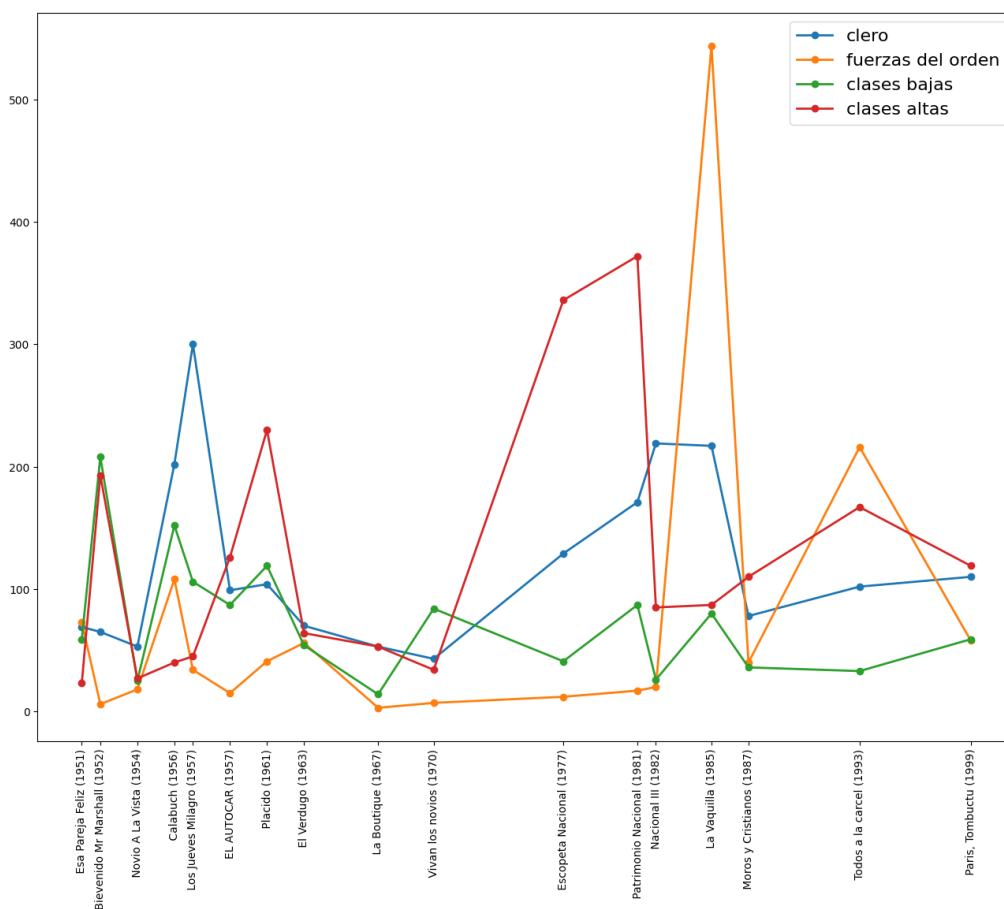


Figura 7.4: Evolución de conceptos con ‘El Autocar (1957)’

Si comparamos con la tabla original observaremos que se sigue aproximadamente la tendencia seguida entre Los Jueves Milagro y Plácido (Que son las dos películas que hay entre medias de la redacción de esta idea). Si bien es cierto que los valores son algo inferiores a los esperados, pero como se puede observar en la gráfica la evolución de las distintas temáticas es bastante ‘caótica’ y no sigue una lógica muy escrita más allá de lo explicado en su correspondiente apartado, por lo que dentro de lo cabe los valores obtenidos no desentonan con la evolución de conceptos de la época.

### 7.5.5. Análisis de ubicaciones

Por último, se puede observar su posición en las gráficas de proporción de interiores y exteriores (imagen ), y de proporción de escenas de día y de noche (imagen 7.6). Esto lo añado más como curiosidad, pues no tiene tanto valor como los análisis anteriores.

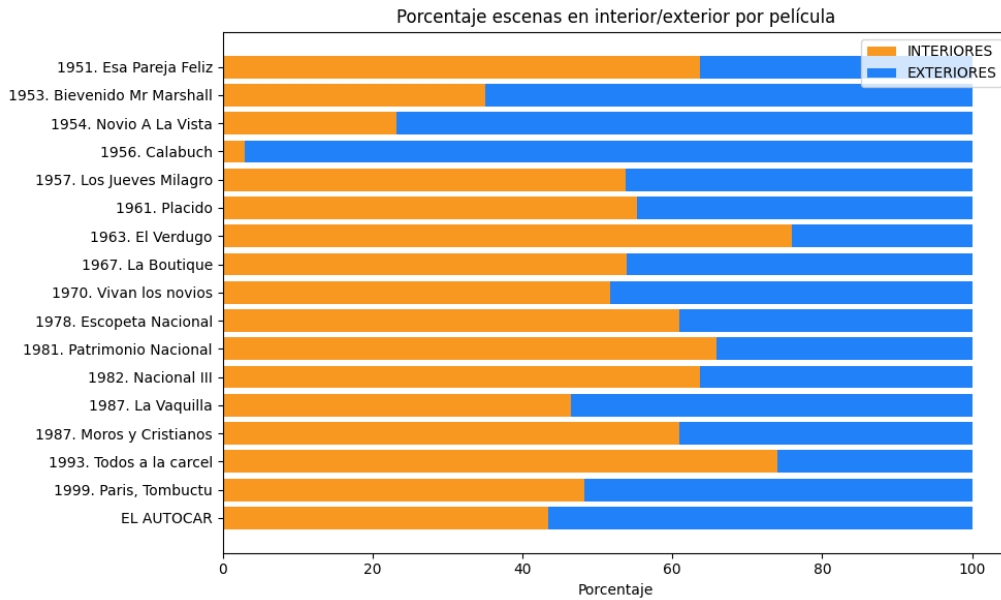


Figura 7.5: Porcentaje escenas en interior/exterior por película con 'EL AUTOCAR'

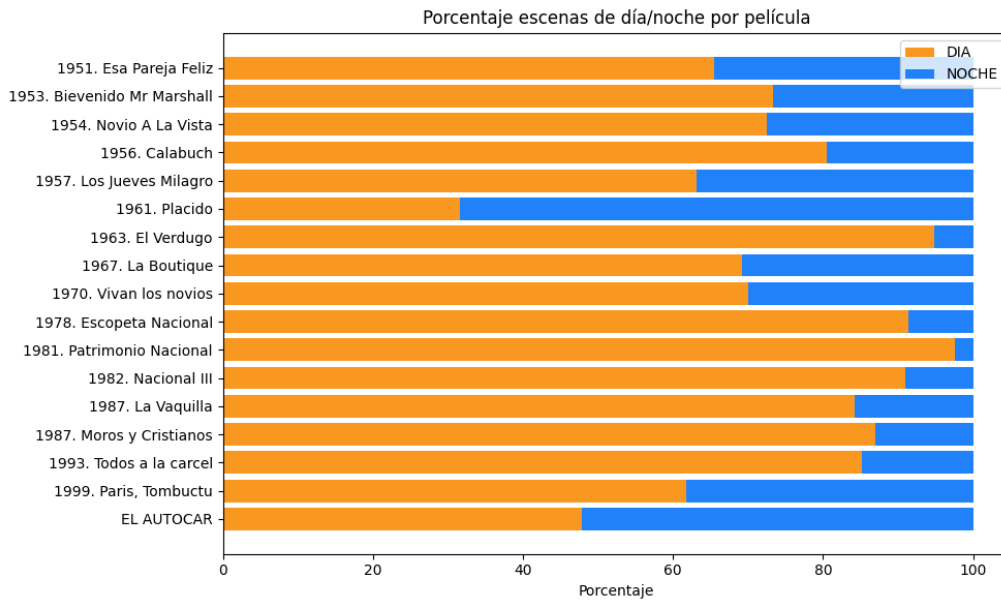


Figura 7.6: Porcentaje escenas de día/noche por película con 'EL AUTOCAR'

### **7.5.6. Conclusión**

Hemos podido observar en los últimos apartados como este nuevo guion generado por el modelo BERTlanga cumple mucha de las características que cumplen los propios guiones de Berlanga.

Animo a una lectura detallada del guion completo para observar todas estas características por escrito y apreciar las capacidades de los modelos de generación de texto LLM actuales. De esta forma se podrá comprobar lo fluidas y realistas que son muchas de las conversaciones entre personajes, aunque también caóticas, como es característico de este director.

Además todas las descripciones de movimientos de cámara y encuadres facilitarían mucho un posible proyecto de rodaje de una nueva película con este guion.

## Capítulo 8

# Conclusión final

En este último apartado voy a realizar un repaso sobre el coste que ha significado este proyecto junto con como enfocar el mismo hacia el futuro.

### 8.1. Presupuesto del proyecto

Este proyecto ha necesitado de GPUs potentes para poder entrenar los distintos modelos. Las GPUs se han cosegido con una suscripción a Google Colab, el cual permite acceso a todo un rango de GPUs de todo tipo de potencia: TPU, T4 GPU, L4 GPU, A100 GPU... Además todos ellos con una gran capacidad de RAM de hasta 60GB.

Esta suscripción tiene un precio de 12€/mes. Puesto que me suscribí al comenzar el proyecto, en marzo de 2023, el entrenamiento de los modelos LLM opensource ha supuesto un coste de 168€ en total. Esto ha sido así a causa de mi desconocimiento inicial sobre como entrenar estos modelos. Esta claro que si desde un principio hubiese tenido claro como realizarlo con una simple suscripción se hubiesen podido realizar los dos entrenamientos.

Por otro lado, el entrenamiento de ChatGPT supuso otra gran inversión, pues esta tiene un precio de 0.008€/1000 tokens. En mi caso, al tener que entrenar el modelo con 4627749 tokens aproximadamente el entrenamiento final supuso un gasto de 37€. Si se hubiese querido entrenar el modelo con la versión más reciente de ChatGPT, la 4, el precio sería mucho más elevado, pues vale 0.09€/1000 tokens, lo que para mi dataset se traduciría en 417€.

De esta forma quiero hacer hincapié y reivindicar los modelos LLM opensource, pues con un precio mucho más reducido hemos sido capaces de obtener resultados verdaderamente sobresalientes para modelos tan pequeños. Y, en caso de querer alcanzar la eficacia de ChatGPT, se podría realizar sin ningún tipo de duda con los modelos de Llama70B o Mistral8x7B.

### 8.2. Próximos objetivos

Este proyecto abre las puertas hacia muchas nuevas direcciones a cada cual más interesante y provechosa.

Primeramente, el paso siguiente lógico sería realizar ese tan prometido entrenamiento con modelos open sources grandes, como Llama70B o Mistral8x7B. De esta forma se podría confirmar que efectivamente son tan capaces como ChatGPT para realizar un guion de estas características. Para ello harían falta la compra de varias GPUs potentes con una gran inversión, pero que a futuro sería muy útil para realizar proyectos como este.

Además, me hubiese gustado también realizar un análisis mucho más exhaustivo de los resultados obtenidos con Mistral7B y Llama3. Pero decidí centrarme más en un único modelo y sacar conclusiones más sólidas.

Por otro lado, me parecería interesante usar la Inteligencia Artificial no solo para analizar su obra, sino para remasterizarla. Es bien conocido como muchas películas antiguas en blanco y negro están empezando a ser colorizadas. También se ha conseguido que se mejore la calidad tanto visual como auditiva de películas antiguas. Es por ello que me gustaría introducirme en un proyecto que tenga como objetivo la restauración de sus películas antiguas para que estas no caigan en el olvido.

También sería relevante no olvidar a muchos otros directores de cine español que merecen tanto respeto como Berlanga y que quizás también se debería realizar un análisis de su obra con nuevas técnicas como las descritas hasta ahora. Es el caso de Juan Antonio Bardem, Luis Buñuel o Carlos Saura entre otros, cuya trayectoria cinematográfica puede llegar a ser tan relevante en la historia del cine español como la es la de Berlanga.

Por último, la capacidad de generar nuevos guiones nos permitiría rodar una nueva película (o cortometraje si se quiere), pues ya tenemos en el guion indicando detalladamente qué planos usar, que movimientos de cámara hacer y en qué ubicaciones rodar. Lógicamente este es un proyecto que necesitaría de una subvención o colaboración monetaria grande de algún tipo para poder realizarse pero que no descarto intentar en un futuro.

También existe la opción de crear la película con un modelo de generación de imágenes o vídeos. Ya vimos en febrero de 2024 como el modelo Sora de OpenAI era capaz de generar vídeo a partir de texto. Sería una conclusión interesante finalizar este proyecto con la creación de la película mediante Inteligencia Artificial.

### **8.3. Conclusión**

Independientemente de las opiniones personales que pueda suscitar un proyecto como este, quiero dejar constancia de la importancia de la Inteligencia Artificial en nuestro día a día y de como podemos utilizarla en diversos campos y con múltiples objetivos. A partir de las nuevas herramientas de IA que aparecen todos los días se pueden realizar nuevos análisis sobre nuestras vidas presentes y pasadas, como es el caso del cine.

En este proyecto, se ha conseguido mediante la Inteligencia Artificial que se confirmen todas aquellas características que los cinéfilos, estudiantes de estudios cinematográficos y expertos de cine llevan años reivindicándole a Berlanga. Pero no sólo eso, sino que además se han descubierto características que hasta ahora no hay registro de que se hayan nombrado y pueden dar lugar a un nuevo debate sobre su obra.

Finalmente, confío en que este proyecto haya contribuido, al menos, a revitalizar la imagen de un director tan fundamental para el cine español y la historia de España como lo es Luis García Berlanga. Si es así, me doy por satisfecho.

# Bibliografía

- [1] Quatre Films. *Luis García Berlanga*. Nov. de 2022. URL: <https://quatrefilms.com/2022/11/28/luis-garcia-berlanga/> (visitado 14-04-2024).
- [2] Miguel Ángel Villena. *Berlanga: Vida y cine de un creador irreverente*. Spanish. Barcelona: Tusquets Editoriales, 2021. ISBN: 8490669260.
- [3] Juan A. Hernández Les. *El último austrohúngaro: Conversaciones con Berlanga*. Spanish. Madrid: Alianza Editorial, 2020. ISBN: 8413621275.
- [4] David Felipe Arranz. *El universo de Luis García Berlanga*. Spanish. Hardcover. Madrid: Notorious Ediciones, mayo de 2021, pág. 358. ISBN: 9788418181214.
- [5] Elvira Lindo. *Lo Berlanguiano*. Conferencia audiovisual. Charla impartida en CaixaForum Valencia, España. Evento dentro del ciclo "Todo me parece Berlanga", abr. de 2024.
- [6] Berlanga Film Museum. *Berlanga Film Museum*. Consultado el 14 de abril de 2024.
- [7] FilmAffinity. *Película 'Plácido'*. Consultado el 14 de abril de 2024. 1961. URL: <https://www.filmaffinity.com/es/film895672.html>.
- [8] Gustavo Mercado. *The Filmmaker's Eye: Learning (and Breaking) the Rules of Cinematic Composition*. 1st. Paperback. United States: Routledge, sep. de 2010, pág. 192. ISBN: 9780240812175.
- [9] PySceneDetect. *PySceneDetect*. Consultado el 14 de abril de 2024. URL: <https://www.scenesdetect.com/>.
- [10] Gunārs Cīvjans. *Cinematics. Movie Measurement and Study Tool Database*. Consultado el 14 de abril de 2024. University of Chicago. URL: <https://cinematics.uchicago.edu/>.
- [11] Adam Geitgey. *face\_recognition*. [https://github.com/ageitgey/face\\_recognition](https://github.com/ageitgey/face_recognition). Python library for facial recognition. 2017.
- [12] Ultralytics. *YOLOv8*. <https://github.com/ultralytics/ultralytics>. Real-Time Object Detection. 2023.
- [13] Javier Meléndez Martín. *Cómo dirigir una película como Berlanga*. Yorokobu. Jun. de 2021. URL: <https://www.yorokobu.es/dirigir-una-pelicula-como-berlanga/>.
- [14] M. Savardi et al. "Shot Scale Analysis in Movies by Convolutional Neural Networks". En: *2018 25th IEEE International Conference on Image Processing (ICIP)*. Oct. de 2018, págs. 2620-2624. DOI: 10.1109/ICIP.2018.8451474.
- [15] M. Svanera et al. "Who is the Film's Director? Authorship Recognition Based on Shot Features". En: *IEEE MultiMedia* 26.4 (2019), págs. 43-54. DOI: 10.1109/MMUL.2019.2940004.



- [16] S. Benini et al. “On the influence of shot scale on film mood and narrative engagement in film viewers”. En: *IEEE Transactions on Affective Computing* (2019), págs. 1-1. DOI: 10.1109/TAFFC.2019.2939251.
- [17] Alex Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>.
- [18] Wikipedia. *Nube de palabras — Wikipedia, The Free Encyclopedia*. <http://es.wikipedia.org/w/index.php?title=Nube%20de%20palabras&oldid=148705304>. [Online; accessed 14-April-2024]. 2024.
- [19] Andreas Mueller. *WordCloud for Python documentation*. Consulted on 2024-04-14. GitHub Pages. 2020. URL: [https://amueller.github.io/word\\_cloud/](https://amueller.github.io/word_cloud/).
- [20] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.
- [21] Nikolaj Lyngø Olsson. *Subtitle Edit: Free Subtitle Editor*. Consulted on 2024-05-18. Nikse.dk. 2024. URL: <https://www.nikse.dk/SubtitleEdit/>.
- [22] Daniel Jurafsky y James H. Martin. *Speech and Language Processing*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall, 2024.
- [23] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition*. <http://cs231n.stanford.edu/>. Accessed: 2024-05-30. 2024.
- [24] Hugging Face. *Curso de NLP*. <https://huggingface.co/learn/nlp-course/es>. Accessed: 2024-05-30. 2024.
- [25] I. Made Agus Dwi Suarjaya. “A New Algorithm for Data Compression Optimization”. En: *International Journal of Advanced Computer Science and Applications* 3.8 (2012), págs. 14-17. DOI: 10.14569/IJACSA.2012.030803. URL: <http://dx.doi.org/10.14569/IJACSA.2012.030803>.
- [26] AI Planet. *Getting Started with Deep Learning: Convolutional Neural Networks*. 2024. URL: <https://aiplanet.com/learn/getting-started-with-deep-learning-es/convolutional-neural-networks/1613/learning-unit-some-other-popular-neural-networks>.
- [27] Ashish Vaswani et al. “Attention Is All You Need”. En: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 2017. URL: <https://papers.nips.cc/paper/7181-attention-is-all-you-need>.
- [28] Jay Alammar. *The Illustrated Transformer*. 2018. URL: <http://jalammar.github.io/illustrated-transformer/>.
- [29] Mistral AI. *Mistral AI*. Consultado el 20 de mayo de 2024. 2024. URL: <https://www.mistral.ai/>.
- [30] Mistral AI. *Mistral-7b: A Powerful AI Language Model*. Disponible en <https://www.mistral.ai/models/mistral-7b>. Consultado el 20 de mayo de 2024. 2024.
- [31] LLAMA. *LLAMA 3 Documentation*. 2024. URL: <https://llama.meta.com/llama3/> (visitado 30-05-2024).
- [32] OpenAI. *GPT-4 Research*. 2024. URL: <https://openai.com/index/gpt-4-research/> (visitado 30-05-2024).

**Parte II**

**Anexos**



## Apéndice A

### Códigos

```
1 from scenedetect import VideoManager, SceneManager, detect, AdaptiveDetector, ContentDetector
2
3 def detect_scene_changes(video_path):
4     scene_list = detect(video_path, AdaptiveDetector(adaptive_threshold=2.5))
5
6     for i, scene in enumerate(scene_list):
7         print('    Escena -: Comienzo %s / Frame %d, Final %s / Frame %d' % (
8             i+1,
9             scene[0].get_timecode(), scene[0].get_frames(),
10            scene[1].get_timecode(), scene[1].get_frames(),))
11     return scene_list
12
13 video_path = "../1977. La Escopeta Nacional.mp4"
14 scenes = detect_scene_changes(video_path)
```

Figura A.1: Código detección de escenas con scenedetect en python

```

1 from ultralytics import YOLO
2 from datetime import timedelta
3 from tqdm import tqdm
4 import cv2, os, csv
5
6 def detect_persons(VIDEO_SOURCE):
7     # Inicializamos la captura del vídeo y ajustamos el tamaño de la ventana
8     cap = cv2.VideoCapture(VIDEO_SOURCE)
9     frame_width = int(cap.get(3))
10    frame_height = int(cap.get(4))
11    resize_width = 640
12    resize_height = 480
13    if frame_width > 0: resize_height = int((resize_width / frame_width) * frame_height)
14
15    skip_frames = 13 # Procesamos cada 13 frames
16    frame_count = 0
17
18    chosen_model = YOLO("yolov8n.pt") # Cargamos el modelo YOLO_v8
19
20    fps = cap.get(cv2.CAP_PROP_FPS)
21    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
22    total_processed_frames = total_frames // skip_frames
23    pbar = tqdm(total=total_processed_frames)
24    csv_route = filename.replace(".mp4", "_persons.csv")
25
26    with open(csv_route, 'w', newline='') as file:
27        writer = csv.writer(file)
28        writer.writerow(['filename', 'Timestamp', 'Persons detected'])
29
30    while True:
31        success, img = cap.read()
32        if not success:
33            break
34
35        frame_count += 1
36        if frame_count % skip_frames != 0: # Saltamos frames
37            continue
38
39        img = cv2.resize(img, (resize_width, resize_height))
40        results = chosen_model.predict(img, conf=0.5) # Obtenemos las predicciones
41
42        for result in results: # Dibujamos los cuadros y etiquetas
43            for box in result.boxes:
44                color = (0, 255, 0) if result.names[int(box.cls[0])] == "person" else (0, 0, 255)
45                cv2.rectangle(img, (int(box.xyxy[0][0]), int(box.xyxy[0][1])), (int(box.xyxy[0][2]), int(box.xyxy[0][3])), color, 2)
46                cv2.putText(img, f"{result.names[int(box.cls[0])]", (int(box.xyxy[0][0]), int(box.xyxy[0][1]) - 10), cv2.FONT_HERSHEY_PLAIN, 1, color, 1)
47
48        # Contamos el número de personas detectadas
49        persons_detected = sum([1 for result in results for box in result.boxes if result.names[int(box.cls[0])] == "person"])
50
51        # Calculamos el timestamp
52        seconds = int(frame_count / fps)
53        milliseconds = int((frame_count / fps - seconds) * 1000)
54        timestamp = str(timedelta(seconds=seconds)) + '.' + str(milliseconds).zfill(3)
55
56        # Escribimos los resultados en un csv
57        writer.writerow([os.path.basename(VIDEO_SOURCE), timestamp, persons_detected])
58        pbar.update(1)
59
60    pbar.close()

```

Figura A.2: Código detección de peronas con YOLOv8

```

1 import os, glob, imageio, numpy as np, pandas as pd, holoviews as hv
2 from PIL import Image as pil_image
3 from tensorflow.keras.models import load_model
4 from tqdm import tqdm
5
6 id2cls = ['CU', 'MS', 'MLS', 'LS', 'VLS'] # Close-up, Medium Shot, Medium Long Shot, Long Shot, Very Long Shot
7
8 def shot_scale(image, model, dims = (125, 224)):
9     # Dimensiones de las imágenes
10    ancho_altura = (dims[1], dims[0])
11    color_value = 0
12
13    # Convertimos la imagen a un objeto de PIL
14    raw_img = pil_image.fromarray(image)
15
16    # Creamos copia de la imagen y ajustamos sus dimensiones
17    img = raw_img.copy()
18    img.thumbnail(ancho_altura, pil_image.NEAREST)
19
20    # Creamos la imagen final con el color de fondo seleccionado
21    final_img = pil_image.new(img.mode, ancho_altura,
22                              (color_value if img.mode == 'L'
23                               else (color_value, color_value, color_value)))
24
25    # Pegamos la imagen en el centro de la imagen final
26    final_img.paste(img, ((ancho_altura[0] - img.size[0]) // 2, (ancho_altura[1] - img.size[1]) // 2))
27
28    # Convertimos la imagen a un array de numpy y normalizamos
29    image_c = np.asarray(final_img, dtype='float32') / 255.
30    image_bn = np.asarray(final_img.convert('LA').convert('RGB'), dtype='float32') / 255.
31    image = np.stack([image_c, image_bn], axis=0)
32
33    # Realizamos la predicción
34    prediction = np.sum(model.predict(image), axis=0)
35
36    return np.argmax(prediction)
37
38 def shot_analysis(videos):
39     for filename in videos: # Recorremos todos los videos
40         time_step = 5 # en segundos
41
42         # Cargamos el video
43         video = imageio.get_reader(filename, 'ffmpeg')
44         pelicula = os.path.basename(filename)[: -4]
45         out = []
46
47         # Obtenemos el número total de frames y recorremos el los frames
48         num_frame = video.get_meta_data()['duration'] * video.get_meta_data()['fps']
49         for num in tqdm(range(int(num_frame // (video.get_meta_data()['fps'] * time_step)))):
50             # Obtenemos el frame actual
51             image = video.get_data(int(time_step * num * video.get_meta_data()['fps']))
52
53             # Llamamos a la función de shot_scale que nos devolverá la predicción
54             preds = shot_scale(image, model)
55
56             # Añadimos la predicción a la lista de salida
57             out.append([pelicula, int(time_step * num * video.get_meta_data()['fps']), preds, image])
58
59         # Creamos un dataframe con los resultados y lo guardamos en un archivo csv
60         pd_output = pd.DataFrame([o[: -1] for o in out], columns=['pelicula', 'frame', 'prediction'])
61         pd_output.to_csv(folder_path + "/" + os.path.basename(filename).replace(".mp4", ".csv"))
62
63     return out, pd_output
64
65 model = load_model('/content/model_shotscale_967.h5')
66 folder_path = "/content/drive/MyDrive/Berlanga/Peliculas/Films/"
67 mp4_files = glob.glob(f"{folder_path}/*.mp4")
68
69 out, pd_output = shot_analysis(mp4_files)

```

Figura A.3: Código detector de planos usados en cada frame

```
1 import pandas as pd
2 import os, glob, re, string, pickle
3 from sklearn.feature_extraction.text import CountVectorizer
4 from wordcloud import WordCloud
5 import matplotlib.pyplot as plt
6
7 def count_words(text):
8     word_list = text.split()
9     return len(word_list)
10
11 def data_preparation_txt(txt_file, data):
12     data_text = ""
13     with open(txt_file, 'r', encoding='utf-8') as file:
14         read = file.readlines()
15
16     read = [line for line in read]
17     for line in read:
18         data_text += " " + line.strip()
19
20     filename = os.path.splitext(os.path.basename(txt_file))
21     if filename[0].isdigit():
22         filename = filename[2:]
23
24     data = pd.concat([data, pd.DataFrame({'PELICULA': [filename], 'TEXTO': [data_text]})])
25     return data
26
27 def clean_text(text):
28     '''Texto en minúscula, sin paréntesis, sin signos de puntuación, sin números...'''
29     text = text.lower()
30     text = re.sub('&#91;.*?¿\]\%', ' ', text)
31     text = re.sub('&#91;%s]' % re.escape(string.punctuation), ' ', text)
32     text = re.sub('\w*\d\w*', '', text)
33     return text
34
35 CSV_PATH = "TFG/BERTlanga/Utils/ANALYSIS/Planos"
36 extension = '*.txt'
37 column_names = ['PELICULA', 'TEXTO']
38 data = pd.DataFrame(columns=column_names)
39
40 archivos_csv = glob.glob(os.path.join(CSV_PATH, extension))
41 for csv in archivos_csv:
42     if ".txt" in csv:
43         data = data_preparation_txt(csv, data)
44 data.set_index('PELICULA', inplace=True)
45
46 clean_text = lambda x: clean_text(x)
47
48 data_clean = pd.DataFrame(data.TEXT0.apply(clean_text))
49
50 with open('spanish.txt') as f: # Archivo con palabras que no queremos
51     stop_words = f.read().splitlines()
52
53 wc = WordCloud(stopwords=stop_words, background_color="white", colormap="Dark2",
54               max_font_size=150, random_state=42)
55
56 for index, anio in enumerate(data.columns):
57     wc.generate(data_clean.TEXT0[anio])
58     plt.subplot(6, 3, index+1)
59     plt.imshow(wc, interpolation="bilinear")
60     plt.axis("off")
61     plt.title(data_clean.index[index])
62
63 plt.show()
```

Figura A.4: Código generación de Wordclouds

---

```

1 from docx import *
2 from docx.enum.style import WD_STYLE_TYPE
3 import pandas as pd
4 from docx.shared import Pt
5 import re
6
7 # Cargamos el documento de Word
8 document = Document("D:/BERLANGA/Guiones/ElVerdugo.docx")
9
10 # Obtenemos los estilos del documento. Serán tres cuatro tipos: ESCENA, Ubi, Personaje y Dialogo
11 styles = document.styles
12
13 # Creamos un DataFrame vacío para almacenar los datos
14 df = pd.DataFrame(columns=['Tipo', 'Texto'])
15
16 # Inicializamos el índice del DataFrame y la variable para almacenar el nombre del personaje
17 i = 0
18 personaje = ""
19
20 # Iteramos sobre los párrafos del documento.
21 # Según el tipo de estilo se almacenará en el DataFrame con un formato u otro
22 for para in document.paragraphs:
23     if para.style.name == "ESCENA":
24         df.loc[i, 'Tipo'] = 'ESCENA'
25         df.loc[i, 'Texto'] = para.text
26         modoDialogo = False
27     elif para.style.name == "Ubi":
28         df.loc[i, 'Tipo'] = 'UBICACION'
29         df.loc[i, 'Texto'] = para.text
30         modoDialogo = False
31     elif para.style.name == "Personaje":
32         personaje = para.text.rstrip().replace(":", "").replace(".", "").replace(", ", "").upper()
33     elif para.style.name == "Dialogo":
34         df.loc[i, 'Tipo'] = personaje
35         df.loc[i, 'Texto'] = para.text
36     i+=1
37
38 # Guardamos el DataFrame como un archivo CSV
39 df.to_csv('D:/BERLANGA/Guiones/ElVerdugo.csv', header=False, index=False)

```

**Figura A.5: Código conversión archivo .docx con estilos a .csv estructurado**



```

1 import whisperx
2 import json
3
4 device = "cuda"
5 batch_size = 4
6 compute_type = "float16"
7
8 audio_file = "D:/Audios Berlanga/Moros y Cristianos.wav"
9
10 # Cargamos audio
11 audio = whisperx.load_audio(audio_file)
12
13 # Cargamos el modelo de transcripción
14 model = whisperx.load_model("large-v2", device, language="es", compute_type=compute_type)
15
16 # Transcribimos el audio
17 result = model.transcribe(audio, batch_size=batch_size)
18
19 # Alineamos el texto con el audio para obtener los segmentos. Guardamos el resultado
20 model_a, metadata = whisperx.load_align_model(language_code=result["language"], device=device)
21 result = whisperx.align(result["segments"], model_a, metadata, audio, device, return_char_alignments=False)
22
23 # Distinguimos los hablantes
24 diarize_model = whisperx.DiarizationPipeline(use_auth_token="{token}", device=device)
25 diarize_segments = diarize_model(audio, min_speakers=2)
26
27 # Asignamos los hablantes a los segmentos
28 result = whisperx.assign_word_speakers(diarize_segments, result)
29
30 # Convertimos el resultado a un string y lo guardamos en un archivo
31 result_str = json.dumps(result, ensure_ascii=False, indent=4)
32
33 with open('D:/Audios Berlanga/Moros y Cristianos.json', 'w', encoding="utf-8") as f:
34     f.write(result_str)

```

Figura A.6: Código conversión de audio a texto mediante Whisper

```

1 import torch
2 from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig
3
4 base_model_id = "mistralai/Mistral-7B-Instruct-v0.2" # ID del modelo base
5 bnb_config = BitsAndBytesConfig( # Configuración de cuantización
6     load_in_4bit=True, # Cargamos los pesos en 4 bits
7     bnb_4bit_use_double_quant=True, # Usamos cuantización doble
8     bnb_4bit_quant_type="nf4", # Tipo de cuantización
9     bnb_4bit_compute_dtype=torch.bfloat16 # Datos en formato de punto flotante de 16 bits
10 )
11
12 model = AutoModelForCausalLM.from_pretrained(base_model_id, quantization_config=bnb_config, device_map="auto")

```

Figura A.7: Finetunning Mistral7B. 1º Paso: Cargar modelo cuantificado

```
1 def formatting_func(sample):
2     """
3     Función que recibe un diccionario con las claves "instruction", "input" y "output"
4     y devuelve un string con el formato adecuado para el modelo
5     """
6     bos_token = "<s>"
7     system_message = "[INST]" + sample["instruction"]
8     input = sample["input"].strip()
9     response = sample["output"]
10    eos_token = "</s>"
11
12    full_prompt = ""
13    full_prompt += bos_token
14    full_prompt += system_message
15    full_prompt += "\n" + input
16    full_prompt += "[\INST]"
17    full_prompt += response
18    full_prompt += eos_token
19    return full_prompt
20
21    tokenizer = AutoTokenizer.from_pretrained( #Cargamos el tokenizador propio del modelo
22        base_model_id,
23        padding_side="left", # El padding a la izquierda ayuda a usar menos memoria
24        add_eos_token=True,
25        add_bos_token=True,
26    )
27    tokenizer.pad_token = tokenizer.eos_token
28
29    # En caso de querer añadir tokens especiales adicionales al tokenizador.
30    num_added_tokens = tokenizer.add_tokens(new_special_tokens)
31    model.resize_token_embeddings(len(tokenizer)) # Reajustamos el tamaño de la matriz de embeddings
32
33    def generate_and_tokenize_prompt(prompt):
34        return tokenizer(formatting_func(prompt))
35
36    tokenized_train_dataset = train_dataset.map(generate_and_tokenize_prompt)
37    tokenized_val_dataset = eval_dataset.map(generate_and_tokenize_prompt)
```

**Figura A.8: Finetuning Mistral7B. 2º Paso: Cargar tokenizador**

```

1 max_length = 3000 # Nuevo tamaño máximo de la secuencia
2 def generate_and_tokenize_prompt2(prompt):
3     result = tokenizer(
4         formatting_func(prompt),
5         truncation=True,
6         max_length=max_length,
7         padding="max_length",
8     )
9     result["labels"] = result["input_ids"].copy()
10    return result
11
12    tokenized_train_dataset = train_dataset.map(generate_and_tokenize_prompt2)
13    tokenized_val_dataset = eval_dataset.map(generate_and_tokenize_prompt2)

```

**Figura A.9: Finetuning Mistral7B. 3º Paso: Selección de longitud máxima**

```

1 from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training
2
3 model.gradient_checkpointing_enable() # Activamos el checkpointing de gradientes
4 model = prepare_model_for_kbit_training(model) # Preparamos el modelo para el entrenamiento con LoRA
5
6 config = LoraConfig(
7     r=32, # Número de cabezas de atención
8     lora_alpha=64, # Número de bloques de atención
9     target_modules=[
10        "q_proj",
11        "k_proj",
12        "v_proj",
13        "o_proj",
14        "gate_proj",
15        "up_proj",
16        "down_proj",
17        "lm_head",
18    ],
19    bias="none", # Bias de la capa de atención
20    lora_dropout=0.05, # Valor convencional
21    task_type="CAUSAL_LM",
22 )
23
24 model = get_peft_model(model, config)

```

**Figura A.10: Finetuning Mistral7B. 4º Paso: Aplicamos LoRA**

```
1 import transformers
2 from datetime import datetime
3
4 if torch.cuda.device_count() > 1: # Activamos Cuda y la paralelización del modelo
5     model.is_parallelizable = True
6     model.model_parallel = True
7
8 trainer.train()
9
10 output_dir = # Directorio donde se guardarán los checkpoints y los logs
11
12 trainer = transformers.Trainer(
13     model=model,
14     train_dataset=tokenized_train_dataset,
15     eval_dataset=tokenized_val_dataset,
16     args=transformers.TrainingArguments(
17         output_dir=output_dir,
18         warmup_steps=1,
19         per_device_train_batch_size=2,
20         gradient_accumulation_steps=1,
21         gradient_checkpointing=True,
22         #max_steps=500,
23         learning_rate=2.5e-5, # Learning rate pequeño
24         bf16=False,
25         optim="paged_adamw_8bit",
26         logging_steps=50, # Cuando se mostrarán los logs
27         logging_dir="./logs",
28         save_strategy="epoch", # Guardar checkpoints al final de cada época
29         save_steps=50,
30         evaluation_strategy="epoch", # Evaluar el modelo al final de cada época
31         eval_steps=50,
32         do_eval=True, # Realiza la evaluación del modelo
33         num_train_epochs=4,
34         fp16=True,
35         run_name=f"{run_name}-{datetime.now().strftime('%Y-%m-%d-%H-%M')}" # Name of the W&B run (optional)
36     ),
37     data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False),
38 )
39
40 model.config.use_cache = False # Silencia los warnings
41 trainer.train()
```

**Figura A.11: Finetuning Mistral7B. 5º Paso: Entrenamiento final**

```
1 from unsloth import FastLanguageModel
2 import torch
3 max_seq_length = 3000 # Secuencia máxima
4 dtype = None # Con none se autodetecta el tipo de datos
5 load_in_4bit = True # Usa cuantificación 4bit.
6
7 model, tokenizer = FastLanguageModel.from_pretrained(
8     model_name = "unsloth/llama-3-8b-bnb-4bit",
9     max_seq_length = max_seq_length,
10    dtype = dtype,
11    load_in_4bit = load_in_4bit,
12 )
```

**Figura A.12: Finetuning Llama 3. 1º Paso: Cargar modelo cuantificado**

---

```

1 alpaca_prompt = """Below is an instruction that describes a task,
2 paired with an input that provides further context. Write a response
3 in spanish that appropriately completes the request.
4
5 ### Instruction:
6 {}
7
8 ### Input:
9 {}
10
11 ### Response:
12 {}"""
13
14 EOS_TOKEN = tokenizer.eos_token # Must add EOS_TOKEN
15 def formatting_prompts_func(examples):
16     instructions = examples["instruction"]
17     inputs       = examples["input"]
18     outputs      = examples["output"]
19     texts = []
20     for instruction, input, output in zip(instructions, inputs, outputs):
21         # Añadimos el token EOS para que la generación no sea infinita
22         text = alpaca_prompt.format(instruction, input, output) + EOS_TOKEN
23         texts.append(text)
24     return { "text" : texts, }
25 pass
26
27 from datasets import load_dataset
28 data_files = {"train": "BERTlanga_train.json", "test": "BERTlanga_eval.json"}
29 dataset = load_dataset("json", data_files=data_files)
30 train_dataset = dataset["train"]
31 eval_dataset = dataset["test"]
32 tokenized_train= train_dataset.map(formatting_prompts_func, batched=True)
33 tokenized_val = eval_dataset.map(formatting_prompts_func, batched=True)

```

**Figura A.13: Finetuning Llama 3. 2º Paso: Cargar tokenizador**

```
1 model = FastLanguageModel.get_peft_model(  
2     model,  
3     r = 32,  
4     target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",  
5                       "gate_proj", "up_proj", "down_proj",],  
6     lora_alpha = 64,  
7     lora_dropout = 0,  
8     bias = "none",  
9     use_gradient_checkpointing = "unsloth", # Usa 30% menos de VRAM  
10    random_state = 3407,  
11    use_rslora = False,  
12    loftq_config = None,  
13 )
```

**Figura A.14: Finetuning Llama 3. 3º Paso: Aplicamos LoRA**

```

1 from trl import SFTTrainer
2 from transformers import TrainingArguments
3
4 output_dir = # Directorio donde se guardarán los checkpoints y los logs
5
6 trainer = SFTTrainer(
7     model = model,
8     tokenizer = tokenizer,
9     train_dataset = tokenized_train_dataset,
10    eval_dataset=tokenized_val_dataset,
11    dataset_text_field = "text",
12    max_seq_length = max_seq_length,
13    dataset_num_proc = 2, # Número de procesos para la carga de datos
14    packing = False, # Hace que el entrenamiento sea 5x más rápido en secuencias cortas.
15    args = TrainingArguments(
16        per_device_train_batch_size = 2,
17        gradient_accumulation_steps = 4,
18        warmup_steps = 5,
19        learning_rate = 2e-4,
20        fp16 = not torch.cuda.is_bf16_supported(),
21        bf16 = torch.cuda.is_bf16_supported(),
22        logging_steps = 50,
23        optim = "adamw_8bit",
24        weight_decay = 0.01,
25        lr_scheduler_type = "linear",
26        seed = 3407,
27        output_dir = output_dir,
28        logging_dir="./logs",
29        save_strategy="epoch",
30        save_steps=50,
31        evaluation_strategy="epoch",
32        eval_steps=50,
33        do_eval=True,
34        num_train_epochs=4
35    ),
36 )

```

**Figura A.15: Finetuning Llama 3. 4º Paso: Entrenamiento final**





## Apéndice B

### Guiones

A continuación se muestra el guion original de la escena descrita en la sección 7.3 para que se pueda comparar con la escena generada.

**Tabla B.1: Escena original Bienvenido Mr. Marshall. Extracto obtenido de BerlangaFilm-Museum**

---

SALA CASONA DON LUIS, interior, noche

El retrato de don Gaspar de Arce Alonso de Quesada. PAN hasta recoger a don Luis que sentado bajo este retrato, ante la mesa, intenta leer un libro. El sueño le vence, los párpados se le cierran, la cabeza se le cae. Despierta un instante, alarmado. Se recompone e intenta sumirse en la lectura.

NARRADOR: (OFF) Caramba! Don Luis todavía despierta! No será por mucho tiempo. Ya falta poco...

Después de un par de intentos, el sueño lo vence y se queda dormido.

TRAV hacia don Luis.

NARRADOR: (OFF) ... Lo ven?

Empieza a oírse una dulce música tropical, sobre el fondo de un tam-tam.

ENCADENA A...

PLAYA DE AMÉRICA, interior, día

Una playa con arena de verdad, abierta a un mar de papel pintando. En el horizonte, pintados también, dos galeones. La playa está repleta de vegetación subtropical, toda de papel pintado. Una barca entra lateralmente, surcando las falsas olas. En su proa, va don Luis, con armadura del XVI, oteando el horizonte. Descubre tierra y desembarca, ni ritmo del tam-tam se acelera.

Una vez en la playa, cava orgullosamente un estandarte. Después mira a su derredor y cuando su vista se dirige hacia la CÁMARA, debo descubrir algo, porque empieza a hacer señas amistosas. Empiezan a entrar en cuadro, dirigiéndose hacia con Luis, indios; muy sumisos. Al llegar junto a él, le curiosean, le palpan. El ritmo del tam-tam crece todavía. De repente, con un gran griterío, los indios se apoderan de don Luis y lo traen en volandas hacia p.t. TRAV retroceso hasta descubrir un enorme caldero sobre un fuego crepitante. Con armadura y todo los indios sumergen en él a con Luis e inmediatamente inician a su alrededor una danza desenfadada.

Plano corto de Don Luis, sumergido en el caldero. El agua está hirviendo. Don Luis suda y el vapor y el humo nos lo ocultan poco a poco mientras el tam-tam y el griterío arrecian.

ENCADENA A...

SALA CASONA DON LUIS, interior, noche

Coselete, casco, manopla y demás elementos de la armadura sueltos y vacíos en el suelo. Entre el los un hueso mondo y lirondo. Entra un perro y husmea. En este momento cae sobre todas estas una manopla. El perro se asusta y ladra.

Los restos que quedan de pie de la armadura que había junto a la chimenea. Sobre la repisa de ésta, un gato, causante del derrumbe anterior, se eriza ante los ladridos del perro y salta.

Don Luis. El gato cae en su regazo. Se despierta sobresaltado. Se pone en pie. El perro continúa ladrando mientras persigue al gato. Don Luis mira la armadura. Inmediatamente se palpa asustado.

NARRADOR: (OFF) SÍ ... es un sueño. No lo dé importancia...

Don Luis, recupera su dignidad, y un poco quemado, echa a andar hacia la puerta.

NARRADOR: (OFF) Ya es un poco tarde. Por qué no se va a la cama?

Don Luis apaga la luz y da un portazo rotundo.

NARRADOR: (OFF) Buenas noches

Oscuridad absoluta.

---