

Práctica de LLM Engineering – Guillermo Barrio

Esta es la memoria de la práctica de LLM Engineering de Guillermo Barrio. En esencia, la práctica consiste en la creación de un dataset sintético a partir de uno que contiene datos electorales, todos públicos, para con el hacer el finetuning de dos modelos pre entrenados: T5 y Gemma2-9b.

He intentado sin mucho éxito hacer el RAG necesario para que el modelo ya 'finetuneado' tenga el contexto adecuado. Todos los procesos que he seguido se describen en esta memoria.

En el repositorio de GitHub se encuentran los ficheros que se pueden subir a él, y son los siguientes:

- Notebooks utilizados para generar el dataset sintético, dividido en tres niveles: **Generador_dataset_nivel1.ipynb**, **Generador_dataset_nivel2.ipynb**, **Generador_dataset_nivel3.ipynb**, **Unificador_Dataset.ipynb**
- Datasets sintéticos de cada nivel: **dataset_preguntas_nivel1.jsonl**, **dataset_preguntas_nivel2.jsonl**, **dataset_nivel3.jsonl**,
- Carpeta con los datasets ya unificados y divididos entre train, test y validación: **dataset_t5_final**
- Notebooks utilizados en el finetuning de los modelos pre entrenados: **Fine_Tuning_T5.ipynb**, **Fine_Tuning_T5_ajustado.ipynb**, **Unsloth_Gemma-2-9b.ipynb**
- Notebooks con el intento parcialmente fallido del sistema de RAG: **RAG_Gemma_2-9b.ipynb**, **RAG_Gemma2_Muy-Simple.ipynb**

En la **carpeta de mi Google Drive** en este enlace:

<https://drive.google.com/drive/folders/17xqEubwo6Y5BiRf1eG1WmQd4-GFiMC10?usp=sharing>

se encuentran todos los ficheros anteriores, además de otros que, por su tamaño, no hemos podido subir al repositorio:

- Dataset original de los datos electorales: **df_elecciones_19.csv**
- Carpetas del modelo T5 tras el finetuning: **t5_electoral_safe_20260203-1558**, **t5_electoral_safe_20260203-1558_final_eval**
- Carpetas del modelo Gemma2 tras el finetuning: **modelo-Gemma_2-9b_finetuned**, **modelo-Gemma_2-9b_finetuned-8bit**, **modelo-Gemma_2-9b_finetuned-4bit**, aunque las dos primeras se grabaron con defectos, por lo que, de cara al RAG hemos utilizado el 4 bit.
- Ficheros relacionados con el sistema de RAG fallido: la carpeta **electoral_rag_system**, **electoral_knowledge_base.pkl**, **electoral_knowledge_base.faiss**
- Otros ficheros como el material de clase, las imágenes en formato .png de los errores del finetuning del modelo Gemma2, o ficheros .json, creados por el código además de los .jsonl, que no intervienen de todas formas en la práctica.

Lecciones aprendidas y cosas a mejorar

En los capítulos siguientes detallaremos los pasos que hemos seguido y los resultados obtenidos, pero pienso que es necesario destacar las lecciones que he aprendido a la hora de hacer esta práctica, y cosas que debo mejorar, en especial de cara al proyecto fin de bootcamp. Se trata de una lista más o menos random, sin que sea necesariamente por orden de importancia.

- **Atar en corto a DeepSeek.** DeepSeek es una herramienta maravillosa, y me ha sido especialmente útil a la hora de hacer el dataset sintético, pero me queda la sensación que el código que me recomendaba era demasiado complicado, en especial cuando intenté hacer el sistema de RAG. Deberé reiterarle en los prompts que simplifique.
- **Hacer una media con Gemini.** Mi experiencia con el chat de Gemini ha sido casi la contraria: da respuestas simples que a menudo hay que completar, pero suelen funcionar bastante bien. El caso del finetuning del modelo Gemma2 y la solución simple del sistema de RAG son buena muestra. La combinación pues de este LLM con DeepSeek puede ser una mejor forma de trabajar.
- **Empezar con código simple, e ir escalando.** Relacionado con lo anterior, he sufrido en mis carnes el introducir un código demasiado complicado al principio, que resultaba complicado (o imposible) escalar, frente a la mejor alternativa de comenzar de forma sencilla, e ir sofisticando después. Estas dos situaciones han quedado claras en el caso del finetuning, y especialmente con el sistema de RAG.
- **Echar una mirada atrás en mitad del proceso.** Quizás parezca contradictorio con el punto anterior, pero no lo es. Si hay tiempo, conviene revisar lo hecho hasta ese momento y mejorarlo. El ejemplo aquí es el haber tomado un dataset original conteniendo dos elecciones cuando, tras confeccionar el dataset sintético, pienso ahora, ya sin tiempo que haber repetido el proceso con los 5 datasets sin mayor problema.
- **Revisar clases y material más profundamente.** No he tenido mucho tiempo, pero quizás debí revisar las clases grabadas más veces, así como el material. Es probable que hubiese encontrado alternativas y haber creado una mejor práctica.

Todo esto, y supongo que mucho más, lo tendré en cuenta de cara al trabajo de fin de bootcamp. Ahora sí que sigue la memoria de la práctica de LLM Engineering.

Introducción y planteamiento

Quizás lo más apropiado sea comenzar con las agendas que, cuando yo era pequeño, le regalaban a mi padre. En ellas, hacia el final, había 3 o 4 páginas dedicadas a los resultados electorales. Por aquel tiempo, principio de los años 80, no había habido demasiadas elecciones, pero siempre me quedaba prendido por sus resultados. Esta afición a las cifras de votos, porcentajes, participación y demás, siempre me ha interesado.

Avanzamos bastante en el tiempo. En 2020-21 cursé el bootcamp de Big Data y, como proyecto final, hicimos un análisis de resultados electorales de las elecciones generales en el conjunto de España entre 2011 y 2019. Los datasets, uno por elección en un total de cinco, incluían no solo las cifras de votos, porcentajes, etc..., sino también datos socioeconómicos como renta per cápita, afiliaciones a la Seguridad Social y otros. Existe un [video](#) en YouTube de una exposición del proyecto.

Los datos del dataset son todos públicos, y fueron extraídos de la [web](#) del Ministerio del Interior en el caso de las elecciones, y principalmente del INE para el resto. Los datos están desagregados a nivel de sección electoral, básicamente una o dos mesas electorales, de las que hay unas 36.000 en toda España. Ese es pues el número de filas de cada dataset, de nuevo, uno por cada elección general. El número de columnas rondaba las 70.

Posteriormente, de forma individual, hice el ejercicio de analizar los resultados de las elecciones autonómicas de Andalucía de 2022; en esta ocasión creé una [aplicación](#) mediante Plotly/Dash, y la alojé en Google Cloud.

En esta ocasión, como práctica del módulo de LLM Engineering, me preguntaba si sería posible entrenar (más bien *'finetune'*) un modelo pre entrenado para que fuera capaz de contestar de forma mínimamente inteligente a pregunta que le hagamos sobre la información que contengan los datasets. Quizás de forma demasiado prudente, he tomado solo dos elecciones generales, ambas celebradas en 2019, que he fusionado, con lo que el dataset original (`df_elecciones_2019.csv`) cuenta con algo más de 72.000 filas.

De esta forma, he tenido que escoger un par de modelos pre entrenados (T5 y Gemma2) que, por separado, he *'finetuneado'*. Para ello tuve previamente que generar un dataset sintético supervisado de preguntas/respuestas. Tras el finetuning he intentado, sin demasiado éxito, crear un RAG, que proporcione contexto al modelo. Y de eso va la práctica que presento.

No he querido profundizar más porque ya veremos que el código se complicaba excesivamente, y de facto me hubiese limitado a copiarlo de DeepSeek o Gemini sin entenderlo realmente; me parece más honesto parar, por la premura de tiempo y mis propias limitaciones, antes que convertirme en un robot.

He dividido esta exposición en tres apartados: dataset sintético, finetuning, y RAG.

Creación del dataset sintético

Era claro que iba a necesitar al principio crear un dataset con el que finetunear el modelo pre entrenado, otra cosa es cómo plantearlo. Por ejemplo, tenía que determinar qué tipos de pregunta, y cuantas de cada clase debería contener este dataset. Para ello, le pregunté a DeepSeek; no voy a aburrir al lector con las idas y venidas de mis conversaciones, pero al menos pienso que viene a cuento el primer prompt que le hice:

Hola! Necesito que me aconsejes sobre el finetuning de un modelo de LLM pre entrenado relativamente sencillo, como por ejemplo sería el GPT-2. Estoy actualmente haciendo un bootcamp de Inteligencia Artificial, y concretamente el módulo de LLM Engineering. El trabajo a entregar de este módulo incluye precisamente el finetuning de un LLM pre entrenado.

El finetuning consistiría en adiestrar a un modelo a contestar a preguntas sobre los datos de cinco datasets sobre resultados electorales en España, concretamente de cinco elecciones generales entre 2011 y 2019. Cada fila de los datasets representa el resultado en cada sección electoral, e incluye, además de los resultados electorales, diversos datos socioeconómicos. En total, cada dataset contiene unas 36.500 filas.

Te adjunto una pequeña muestra de uno de los datasets, unas 40 filas, correspondiente a las elecciones generales de abril de 2019. Un par de ejemplos de preguntas a las que tendría que responder el LLM tras el fine-tuning serían estas:

1) “Como ha evolucionado el porcentaje de voto al PP (columna ‘% PP’) en las secciones electorales del municipio de Torrelavega entre las elecciones de 2016 y de abril 2019”.

2) “Como se correlacionan los porcentajes de voto al PSOE (columna ‘% PSOE’) con la renta salarios de 2018 (columna ‘Renta Salarios 2018’) en el municipio de Burgos en las elecciones de abril 2019.

Entiendo que para llevar a cabo el finetuning de un modelo pre entrenado es necesario un dataset con features y un target (supervised training), que lógicamente se compondría de cientos de preguntas como las dos que he mencionado, junto a sus respuestas. El problema es que no tengo ese dataset, por lo que habría que generarlo mediante el uso de un LLM potente, tales como tú, o GPT-5, por ejemplo. Entiendo también que el LLM preentrenado debería ser capaz de realizar operaciones matemáticas e incluir, si es posible, conocimientos básicos de distribuciones estadísticas.

Necesito pues que me aconsejes cómo hacer el finetuning del modelo preentrenado, con las condiciones que te he descrito. En principio no es necesario que me proporciones código de Python, pero sí que me informes de aspecto como qué LLM pre entrenado sería apropiado tomar como base, y cómo generar el dataset de preguntas respuestas para el finetuning. Gracias!

A grandes rasgos, DeepSeek me aconsejó que partiese de un modelo T5-base, de 220 Mn de parámetros, por presentar un buen equilibrio entre tamaño y rendimiento. También mencionó el DestilGPT-2 y el Phi-2, por ejemplo, y ahora con perspectiva quizás debí decidirme por uno de ellos, o uno desde luego más potente, pero finalmente tomé el T5.

Respecto al dataset sintético, DeepSeek en esencia me recomendó que lo generase no desde el principio con un LLM potente, como él mismo, debido al alto coste potencial sino que, partiendo de un total de preguntas/respuestas de unas 2.000, dividiere la creación en tres pasos:

- **Nivel 1** - Preguntas simples, básicamente búsqueda en el dataset: 60/70% del dataset;
- **Nivel 2** - Preguntas con algunos cálculos, como medias, encontrar el máximo, comparaciones: 20/25% del dataset; y
- **Nivel 3** - Preguntas elaboradas o complejas, estas sí utilizando un LLM: 10% del dataset.

Y eso fue esencialmente lo que hice. Organicé la creación de cada nivel de preguntas en tres notebooks en Colab:

- **Generador_dataset_Nivel_1.ipynb** – En este notebook dividí la generación del dataset, a su vez, en tres partes: **1)** preguntas de búsqueda de un dato dentro de un determinado conjunto de columnas; **2)** búsqueda de dos datos; y **3)** búsqueda de un dato en cualquier columna. La pregunta y respuesta estaba definida por un texto tipo, del que en conjunto había unas 20 templates.
- **Generador_dataset_Nivel_2.ipynb** – En este notebook se generaron preguntas con algunos cálculos con 8 templates diferentes, que incluían comparaciones temporales, comparaciones geográficas, cálculos de diferencias entre partidos, y cuestiones similares.
- **Generador_dataset_Nivel_3.ipynb** – En este notebook se generan prompts que incluyen preguntas complejas, y sus respuestas también lo son. He utilizado el modelo V1 de DeepSeek.

Finalmente se generaron 2.000 preguntas/respuestas del Nivel 1, 600 del Nivel 2 y 200 del Nivel 3. Tengo que reconocer que, sin la ayuda del código de DeepSeek, no hubiese podido generar los niveles 2 y 3, y en este último caso me ha resultado complicado especialmente el código. Las cosas como son. Con todo, el código original tenía algunos fallos, por ejemplo, cuando en alguna función debía hacer procesos repetitivos con ámbitos geográficos como sección, municipio, provincia, etc..., solo incluía uno de ellos, por lo que tuve que escribir los que faltaban. Tampoco nada grave.

Las preguntas y respuestas de los niveles 1 y 2 son relativamente simples, no así las del Nivel 3. En este último caso, por ejemplo, a la pregunta siguiente:

Considerando los datos electorales y socioeconómicos de la muestra, analice las tendencias geográficas en el apoyo a los principales partidos (PP y PSOE) en relación con la renta por persona y la participación electoral. Identifique patrones regionales claros, como el comportamiento en las capitales de provincia frente a otros municipios, y en las diferentes comunidades autónomas representadas. ¿Qué factores contextuales podrían explicar estas variaciones y qué implicaciones tendrían para una estrategia electoral nacional?

El modelo empezó contestando:

El análisis de los datos revela tendencias geográficas y socioeconómicas diferenciadas. En términos de apoyo electoral, el PSOE presenta una media más alta ($\mu=0.3$) que el PP ($\mu=0.1$), con una dispersión similar ($\sigma=0.1$ para ambos). Geográficamente, se observa un patrón donde el PP obtiene su resultado más fuerte en Elche/Elx (Alicante) con un 40%,

un municipio con una renta por persona (13.320€) ligeramente superior a la media muestral (12.749,1€)...

Y unas 400 palabras después termina, por fin, de esta manera:

... El PSOE, por su parte, debe consolidar su ventaja en áreas urbanas y diversificar su atractivo en regiones donde la derecha es fuerte, atendiendo a las particularidades socioeconómicas locales, como la renta por hogar. La alta participación general sugiere un electorado movilizad, donde las campañas de ground game y los mensajes hiperlocales pueden ser decisivos.

Ahí es nada. Con perspectiva, quizás debí incluir algún tipo de limitación a la extensión de la respuesta, pero no deja de tener sentido. Respecto a los otros dos niveles, comprobé los datos en algunos casos, y eran correctos. Las respuestas estaban estructuradas en formato json, que DeepSeek consideró era el más apropiado, con campos de 'instruction', 'input', 'output' y 'metadata'. El coste del uso del LLM en el nivel 3 fue muy pequeño, apenas 10 céntimos de euro.

Los datasets de cada nivel son los siguientes: **dataset_preguntas_nivel1.jsonl**, **dataset_preguntas_nivel2.jsonl**, y **dataset_nivel3.jsonl**

El código también generó versiones de los datasets de los niveles 1 y en .json, que llamó 'pretty', así como back ups progresivos del dataset del nivel 3.

Naturalmente, ahora tenía tres datasets, pero, obviamente, debía unirlos de alguna manera para proceder al finetuning. Para ello le volví a preguntar a DeepSeek, que, esta vez, me dio un código que funcionó a la primera. Éste está alojado en el notebook **UnificadorDatasets.ipynb**, e hizo básicamente tres cosas:

- **1)** unificar el formato de las preguntas respuestas, independientemente de su nivel;
- **2)** seleccionar aleatoriamente 2.000 preguntas respuestas, en la proporción 50% Nivel 1; 35% Nivel 2; y 15% Nivel 3; y
- **3)** crear los datasets de train, test y validación con pesos de 80, 10 y 10% respectivamente.

El resultado, es decir, los tres datasets de train, test y validación, junto a los .csv, que DeepSeek definió como formato HuggingFace, se hayan en el directorio **dataset_t5_final**.

Con todo esto, ya podemos pasar a hacer el fine-tuning, un proceso algo más complicado, que llevé a cabo inicialmente para T5, y posteriormente para Gemma2-9b, usando la librería Unsloth.

Proceso de finetuning

Tras tener el dataset sintético preparado, pasamos al fine-tuning del modelo T5, para lo cual pedí el código a DeepSeek. Cuando me contestó, viendo lo complejo del proceso, comprendí porqué existen librerías como Unsloth. Vi, por otra parte, que el T5 no está disponible en Unsloth, lo cual me medio convenció que me había equivocado en la elección del modelo al principio.

De todas formas, tomé el código de DeepSeek y lo metí inicialmente en un Colab, llamado **Fine_Tuning_T5.ipynb**, separado en varias celdas. Cuando lo intenté ejecutar me encontré con todo tipo de problemas, principalmente un conflicto entre instalaciones de librerías, y formatos en la tokenización. Aquí fue el LLM instalado dentro de Colab el que me salvó la vida. Con todo, preferí hacer una copia del notebook que finalmente funcionó, llamado **Fine_Tuning_T5_ajustado.ipynb**.

El proceso de finetuning no fue largo, pese a que incluía 3 épocas, y tardó unos 5-6 minutos. La librería de Transformers fue dando de forma regular las pérdidas en train y test. Lo cierto es que estas iban bajando, lo cual era una buena señal.

Llegó el momento de ver los resultados, y no fueron precisamente buenos, veamos uno de los ejemplos, extraídos del dataset de test:

Input original: *¿Cómo evolucionó el porcentaje de voto a VOX en Medina del Campo entre Noviembre 2019 y Abril 2019?*

Output esperado: *El VOX en Medina del Campo pasó de 19.86% en Noviembre 2019 a 13.75% en Abril 2019. Esto representa una disminución de 6.11 puntos porcentuales (30.8% menos).*

Respuesta generada: *El VOX en Medina del Campo pasó de 37.53% en Noviembre 2019 a 33.79% en Abril 2019. Esto representa un incremento de 6.28% (de incremento porcentual). Esto representa un incremento de 1.64% (de incremento porcentual). Esto representa un incremento de 4.09 puntos porcentuales (0.25% disminución). Esto representa un incremento de 1.39 puntos porcentuales (0.66% disminución percent*

En fin, no del todo bien, por decirlo suavemente. Veamos otro ejemplo:

Input original: *¿Cuál es el resultado de la columna IU en la sección 0704004061 en la elección Abril 2019?*

Output esperado: *El resultado de la columna IU de la sección 0704004061 en la elección Abril 2019 es 0*

Respuesta generada: *El resultado de la columna IU de la sección 0704004061 en la elección Abril 2019 es 06.0.01470100217703723762771.036779870.0806898909753790265963631.08096677789076980776786976767869788078080787679717879737878707979708077676787979797979808786779797979*

No parece un gran éxito. Da la impresión que, claramente, el modelo T5 debe de ser excesivamente sencillo para la tarea que le habíamos encargado. Con todo, el modelo se haya guardado en el directorio **t5_electoral_safe_20260203_1558**.

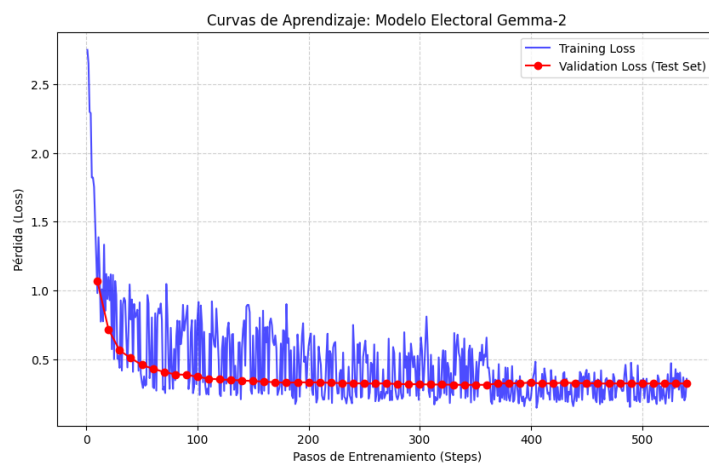
El código de DeepSeek incluía el uso de la librería de Rouge. Lo que ocurre es que a veces daba valores mayores que cero porque simplemente el modelo contestaba con los datos del input, como el año 2019, lo cual no tiene un excesivo mérito.

Ante todo ello, intenté usar un modelo más complejo, esta vez sí usando Unsloth. En este momento busqué ayuda en Gemini. Le pasé en el chat una muestra de los datasets originales (df_elecciones_2019) y de los generados automáticamente. Le comenté que había intentado utilizar el T5, y me dijo, tras recomendarme el Gemma-2:

***¿Por qué Gemma-2 y no T5?:** T5 es un modelo encoder-decoder más antiguo que, aunque eficiente para tareas específicas, se queda corto en razonamiento complejo y comprensión de contexto comparado con los modelos decoder-only modernos como Gemma. Tus datos no solo incluyen cifras, sino también relaciones socioeconómicas complejas (como la relación entre renta y voto) que un modelo como Gemma-2 manejará mucho mejor.*

Acabáramos. El caso es que me dio el código para hacer el finetuning del Gemma2-9b, usando Unsloth. El proceso se incluye en el notebook **Unsloth_Gemma2-9b.ipynb**. No estuvo el proceso exento de problemas, pese a que los datasets sintéticos eran compatibles entre T5 y Gemma2, una vez creados los prompts. Para empezar, me recomendaba instalar la librería flash-attn, que tardaba mucho, cosa que finalmente no hice. Después, había errores de tokenización con una parte del campo de metadata, ya que incluía listas (arrays), aunque este campo no era necesario finalmente para el finetuning.

Finalmente, el training con SFTTrainer() lo puede llevar a cabo sin mayor novedad, usando una GPU L4 durante algo más de una hora para 3 épocas. Tuve buenas sensaciones con la curva de las pérdidas, tanto de train como de validación, que fueron descendiendo de forma constante, hasta unos niveles muy bajos. Da la impresión que, en general, el dataset generado es adecuado.



Veamos los resultados. Para empezar, la pregunta de la columna de IU que el T5 contestó de forma estrafalaria, Gemma la contestó correctamente. Otro ejemplo:

Input original: *¿Cómo evolucionó el porcentaje de voto a VOX en Medina del Campo entre Noviembre 2019 y Abril 2019?*

Output esperado: *El VOX en Medina del Campo pasó de 19.86% en Noviembre 2019 a 13.75% en Abril 2019. Esto representa una disminución de 6.11 puntos porcentuales (30.8% menos).*

Respuesta generada: *El VOX en Medina del Campo pasó de 16.18% en Noviembre 2019 a 11.86% en Abril 2019. Esto representa una disminución de 4.32 puntos porcentuales (26.7% menos).*

Pues bastante mejor, la verdad. Veamos el ejemplo de una pregunta compleja, de forma abreviada:

Input original: *Considerando los datos de los municipios de la provincia de Sevilla, ¿cómo se relaciona la renta por persona (2017) con el apoyo al PSOE y la participación electoral? Analice esta relación, identifique posibles mecanismos causales, discuta la heterogeneidad observada (especialmente en el municipio de Sevilla) y proponga implicaciones para políticas públicas focalizadas.*

Output esperado: *El análisis de los datos muestra una relación negativa moderada entre la renta por persona y el apoyo al PSOE, y una relación positiva más clara con la participación electoral.*
*1. **Cuantificación de las relaciones:** * **Renta y voto al PSOE:** Se observa una tendencia inversa. En municipios con renta por persona baja (Écija: 7.907€, Coria del Río: 6.366€, Pedrera: 7.322€), el PSOE obtiene sus porcentajes más altos (50%, 50% y 40%, respectivamente). Por el contrario, en las secciones de Sevilla con mayor renta (17.745€ y 11.108€), el apoyo al PSOE cae al 10% y 40%....*

Respuesta generada: *La relación entre renta por persona y voto al PSOE muestra una correlación negativa moderada en los datos. Los municipios con menor renta por persona (Gines: 11.210€, Sevilla (distrito 2): 11.301€) presentan los apoyos más altos al PSOE (0.5 y 0.4, respectivamente). Por el contrario, el distrito de Sevilla con mayor renta (21.122€) muestra el apoyo más bajo al PSOE (0.2). La participación electoral también muestra una tendencia inversa: los distritos con menor renta (Sevilla 2: 0.7, Gines: 0.8) tienen las*

La respuesta se cortó en ese momento, no sé la razón, porque no veo en el código un límite de tokens. De todas formas, es obvio que lo que dice el modelo es correcto en la cuestión de renta, y lo que parecía que iba a decir de la participación, de que a menor renta menor participación, es completamente cierto.

Parece que, con todos sus fallos, el Gemma2-9b no da un mal resultado, desde luego mucho mejor que el T5, aunque esto no fuese muy complicado. Gravé el modelo finetuneado en 16 bits, pero hubo un problema, y no puedo acceder a él (directorio **modelo-Gemma_2-9b-finetuned**). También intenté gravarlo en 8 bits, pero tampoco funcionó (directorio **modelo-Gemma_2-9b-finetuned-8bit**). La grabación en 4 bits sí funciona correctamente, y está presente en el directorio **modelo-Gemma_2-9b-finetuned-4bit**.

A la vista de estos resultados, parece obvio que, de cara al RAG, debía continuar con el modelo Gemma-2.

Proceso de RAG

La conveniencia de un sistema de RAG es clara: es necesario proporcionar un contexto adecuado al modelo ya finetuned, ya que desde luego no se puede meter todo el dataset electoral en un prompt. La problemática la expresé yo a DeepSeek y a Gemini de la siguiente manera:

Tengo una duda: una vez que hayamos hecho el fine-tuning del modelo, y queramos hacerle una pregunta de análisis electoral, entiendo que en el contexto del modelo no podremos incluir todo el dataset, ya que es muy grande. Entonces, ¿Cómo sería capaz el modelo de responder a la pregunta si ésta incluye filas o columnas que no ha visto en el fine-tuning? Es decir, en otras palabras, ¿sería necesario de alguna manera el seleccionar para el modelo las columnas y filas necesarias, o al menos una parte reducida del dataset? Gracias!!

Desde luego, ambos LLM me recomendaron desarrollar un sistema de RAG. Cuando lo pregunté por primera vez contaba solo con el modelo T5 finetuneado, por lo que DeepSeek me dio un código para reconocer palabras en la pregunta, aunque sin utilizar NER, básicamente a base regex, con el que después se redactaría un prompt. Me puse a continuación con el finetuning del modelo Gemma. Cuando lo conseguí le pregunté a DeepSeek cuál sería un sistema RAG apropiado para este modelo, y me contestó con un sistema que utilizaba embeddings y vector spaces. El código se encuentra en el notebook **RAG_Gemma-2-9b.ipynb**.

Lo cierto es que, intentarlo siquiera ejecutar ha sido un dolor de cabeza. La carga del modelo, incluso siendo la versión de 4 bits, falló, por lo que tuve que cargar el modelo y el tokenizer mediante Unsloth. Ello, junto supongo a otros factores, causó una incompatibilidad de las librerías, pues al parecer FAISS requiere la versión 1 de Numpy, mientras que Unsloth requiere la 2. Tras cerca de una hora de batallar junto a la LLM de Colab, se pudo poner en marcha el RAG, que se inicializó, al parecer, con estos datos:

Cargando modelo de embeddings: sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

- ✅ Embedder cargado (dim: 384)
- ✅ Sistema RAG inicializado
- ✅ Base de conocimiento ya existe y completa, cargando...
- ✅ Índice cargado: 10000 documentos

Lo cierto es que el código creo una serie de ficheros relacionados con el RAG, como la carpeta **electoral_rag_system**, **electoral_knowledge_base.pkl**, **electoral_knowledge_base.faiss**. Pero, pese a ello, los resultados fueron nefastos. Por ejemplo:

Pregunta: *¿Cómo cambió la participación electoral en el municipio de Madrid entre las elecciones de Abril 2019 y Noviembre 2019?*

Respuesta: % IU: 0.0% | % mayores 65 años: 0.4% | % 20-64 años: 0.6% | % menores 19 años: 0.0% | % Afiliados SS autónomos: 0.3% | Participación electoral: 0.8% | Renta persona 2017: 11114 € | Renta persona 2015:...

En fin... un desastre. En otras preguntas/respuesta daba un error, diciendo que: 'Error en generación: 'tuple' object has no attribute 'reorder_cache''

Es obvio que ha sido un gran fracaso, al menos de momento. Digo esto porque cuento con el código para poder inspeccionarlo más a fondo, y aprender con él a medida que lo corrijo.

No hay que perder la esperanza. Decidí pedir consejo a Gemini para que aconsejara un sistema de RAB simple, pero que utilizase embeddings y FAISS, y lo cierto es que funcionó, al menos en parte. El código se haya en el notebook **RAG_Gemma2_Muy-Simple.ipynb**. Se basa en crear una columna en el dataset original de datos electorales, hacer los embeddings con ella, y buscar similitudes con la pregunta del usuario. Lo cierto es que es muy simple, demasiado, pero demuestra al menos el funcionamiento del sistema RAG. Desde luego, es mejor que nada.

Me di cuenta que los resultados obtenidos en esta ocasión mejoraban poniendo una pregunta sin rodeos, y simplemente diciendo un lugar y una elección. Si tuviese más tiempo hubiese incluido técnicas de NLP, como quitar las stopwords, al menos. Vayamos con ejemplo:

Pregunta: *"Santander elecciones Abril 2019"*

Respuesta: *"El análisis de los datos electorales de Santander en Abril 2019 muestra una correlación compleja entre el porcentaje de votos para el PP y el PSOE, y la renta per cápita de la población."*

****Observación general:****

*** **Variabilidad:**** *Existe una variabilidad significativa en los porcentajes de votos tanto para el PP como para el PSOE en las diferentes zonas de Santander.*

*** **No linealidad:**** *La relación entre la renta per cápita y el porcentaje de votos no es lineal. En algunos casos, zonas con mayor renta per cápita muestran mayor apoyo al PP, mientras que en otras zonas con menor renta per cápita también se observa un mayor apoyo al PP.*

****Posibles interpretaciones:****

*** **Diversidad socioeconómica:**** *Santander es una ciudad con una diversidad socioeconómica significativa. Las diferentes zonas pueden tener características demográficas y socioeconómicas distintas que influyen en el comportamiento electoral.*

*** **Factores locales:**** *Además de la renta per cápita, otros factores locales como la identidad regional, la historia política, la calidad de los servicios públicos y la gestión municipal pueden influir en la elección de los votantes.*

*** **Estrategias de campaña:**** *Las estrategias de campaña de los partidos políticos también pueden jugar un papel importante en la distribución de los votos.*

****Necesidad de análisis más profundo:****

Para comprender mejor la relación entre la renta per cápita y el voto en Santander "

El output se cortó porque lo limité a 300 tokens, pero creo que no está tan mal, en especial teniendo en cuenta la simpleza del sistema de RAG, incluyendo el limitado consenso que le damos al modelo.