

Chatbot médico

Josu Barrutia

Minería de Datos Textuales / Facultad de Informática, EHU

jbarrutia006@ikasle.ehu.eus

1 Introducción

La intención de este proyecto es crear un chatbot especializado en el dominio médico que sea capaz de responder a preguntas relacionadas con la salud de carácter general.

Concretamente, se pretende que el chatbot sea capaz de proporcionar un diagnóstico mediante una conversación y a partir de los síntomas o información que el usuario le proporcione, prácticamente lo que haría un médico de cabecera.

2 Afrontamiento del problema

Dadas las limitaciones de los LLMs especialmente en tareas específicas de dominio o que requieren conocimientos intensivos, cabe la posibilidad de la aparición de "alucinaciones" cuando manejan consultas que están más allá de sus datos de entrenamiento o que requieren información actualizada.

Para evitar esto, existen varias formas de optimizar los LLMs, como se muestra en la figura 1. Entre estas técnicas principales destacan: prompt engineering, Retrieval Augmented Generation y fine-tuning.

La ingeniería de prompts aprovecha las capacidades inherentes de un modelo con una mínima necesidad de conocimiento externo y adaptación del modelo. RAG puede compararse a proporcionar a un modelo un libro de texto a medida para la recuperación de información sobre la pregunta en cuestión. Mientras, FT es comparable a un estudiante que internaliza el conocimiento a lo largo del tiempo, adecuado para escenarios que requieren la replicación de estructuras, estilos o formatos específicos, [Gao et al. \(2024\)](#).

Por lo tanto, aunque la mejor opción para esta tarea sería tratar de realizar un fine-tuning de un LLM (añadiendo un RAG opcionalmente, ya que no son mutuamente excluyentes y pueden combinarse entre sí), dada la complejidad de realizar un

fine-tuning de un modelo de lenguaje, se optará por la opción de RAG, con la que se ha demostrado obtener buenos resultados en tareas de generación de texto en dominios específicos y no requiere la adaptación un modelo.

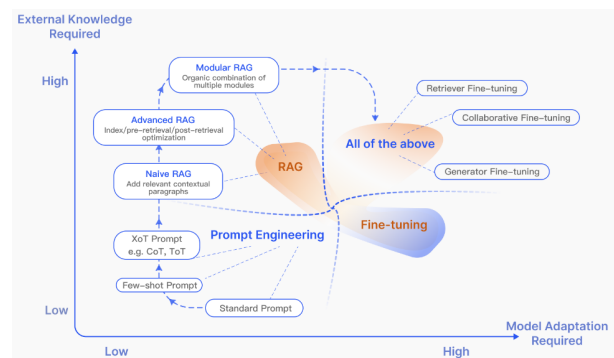


Figura 1: Métodos de optimización de LLMs representados en un espacio bidimensional dependiendo del conocimiento externo requerido y la adaptación del modelo.

3 Arquitectura RAG

Retrieval Augmented Generation es un conjunto de técnicas para enriquecer las salidas de un modelo de lenguaje mediante la recuperación de información contextual de una fuente de datos externa e incluyendo esa información como parte de las indicaciones de tu modelo de lenguaje.

La idea es que al aumentar el contexto de la query de un modelo de lenguaje, a través del prompt, con datos externos significativos, el modelo de lenguaje puede responder con un entendimiento más profundo y relevante.

Los componentes principales de una arquitectura RAG son el "Retrieval", "Augmentation" y "Generation".

El componente de "Retrieval" se encarga de recuperar información relevante de una fuente de datos externa, aquí hay varios problemas clave involu-

crados, como la fuente de recuperación, la granularidad de la recuperación, el preprocesamiento de la recuperación y la selección del modelo de embeddings correspondiente. Por esto, surgen infinitud de mejoras posibles, como la fuente de recuperación, granularidad de la recuperación, indexing optimization, query optimization o selección del modelo de embeddings.

Tras recuperar la información relevante, no suele ser buena idea añadir toda esta información al modelo, es por esto que surge el componente de "Generation", que utiliza técnicas como Context Curation (Reranking) o LLM Fine-tuning.

Aunque la práctica común es utilizar un único paso de retrieval y generation, se puede utilizar técnicas de "Augmentation" para mejorar la calidad de la respuesta, esto contempla técnicas como Iterative Retrieval, Recursive Retrieval o Adaptive Retrieval.

Una arquitectura Naive RAG sigue el siguiente proceso que incluye Indexing, Retrieval y Generation.

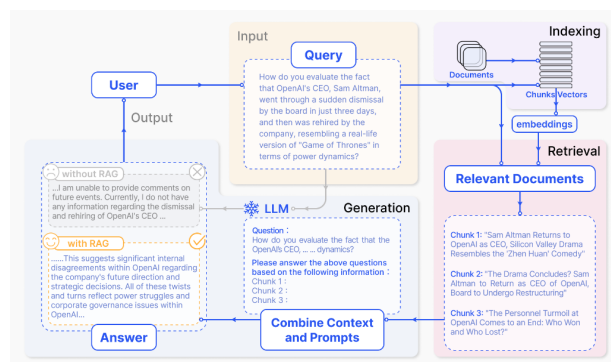


Figura 2: Funcionamiento de una arquitectura Naive RAG.

4 Datos

Para poder inyectar conocimiento sobre medicina en el modelo RAG se necesita un gran corpus de texto médico. Para ello, se ha utilizado un corpus de abstracts de PubMed <http://ixa2.si.ehu.es/~jibloleo/pubmedAbstraktak.txt.bz2>.

Este conjunto de datos consta de 4.639.522 de pares título-abstract de artículos médicos. La estructura que se sigue consta de un título y su correspondiente abstract en una nueva línea, y éstos seguidos por más pares título-abstract, separados por una nueva línea, y así sucesivamente.

Al realizar un análisis de los abstracts se ha observado que sus longitudes siguen la siguiente dis-

tribución:

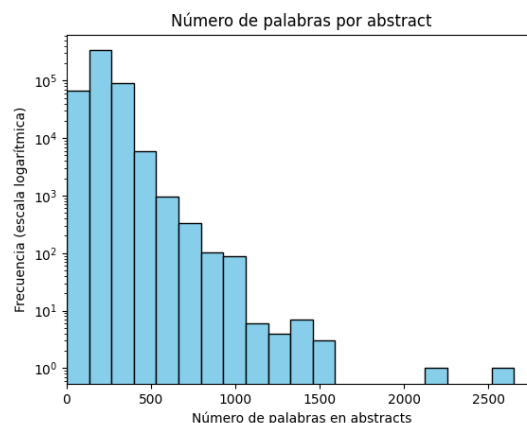


Figura 3: Distribución de la longitud de los abstracts sin tokenizar (número de palabras).

5 Implementación del chatbot

Para construir el chatbot tenemos que empezar consiguiendo un retriever, para ello tenemos que indexar los datos que tenemos en una vectorStore.

El primer paso será cargar los datos de nuestro corpus de abstracts de PubMed. Una vez cargados, tendremos que dividirlos, para ello, la aproximación más intuitiva sería dividir a nivel de abstract, descartando los títulos (realmente no aportan información, aunque podrían ser útiles para mejorar la recuperación y se añadirán como metadatos). Dadas las longitudes de los abstracts y teniendo en cuenta la longitud de tokens máxima del modelo de embeddings y el LLM que se vaya a utilizar, esta división es una muy buena aproximación.

Una vez divididos los abstracts, habrá que tokenizar y calcular los embeddings de cada palabra. Para ello, se utilizará un modelo de embeddings preentrenado *jina-embeddings-v2-base-es* (Günther et al., 2023), con una secuencia máxima de 8192 tokens, 137 millones de parámetros y, a diferencia de muchos otros, es un modelo bilingüe español-inglés, lo cual posibilitará la recuperación de información en ambos idiomas, aunque la información recuperada será en inglés.

Conseguidos los embeddings, se procederá a indexar los abstracts. Para ello, se utilizará FAISS, una biblioteca de indexación y búsqueda de vectores de alta eficiencia.

Con esto ya tendremos un retriever que nos permitirá recuperar información relevante a partir de una query.

Como modelo de lenguaje se utilizará *Mistral-7B-Instruct-v0.2* (Jiang et al., 2023), un modelo de lenguaje de 7B de parámetros, con una secuencia máxima de 32k tokens y que es la versión fine-tuneada instruida de *Mistral-7B-v0.2*. Para poder hacer inferencia con un modelo con tantos parámetros en una infraestructura limitada como Kaggle, es necesario cuantizarlo, en este caso se ha cuantizado a 4 bits.

Se ha probado a utilizar otros LLMs como *gemma-2b* (Team et al., 2024), en este caso sin éxito pues ha sido entrenado para evitar responder a preguntas comprometidas como las relacionadas con la salud.

En este punto, podemos utilizar un framework como langchain para montar el chatbot, hay que incorporar una memoria para poder referirnos a mensajes anteriores y poder mantener una conversación coherente.

Aquí pueden surgir problemas, como que la historia crezca demasiado, es por esto que por defecto se incorpora un paso adicional antes de la recuperación que combina el historial de chat y la pregunta en una "standalone question", esto se realiza con un LLM.

El funcionamiento es sencillo, se le pasa el historial de chat y la pregunta, y el modelo devuelve una pregunta que se le pasará al retriever para recuperar información relevante. Esta información se añadirá al prompt del LLM, junto con la pregunta actual para que pueda generar una respuesta más coherente.

5.1 Conversaciones con el chatbot

En este [enlace](#) pueden verse conversaciones en castellano y en inglés, además se puede analizar el funcionamiento de toda la arquitectura del chatbot, desde el retriever con los documentos seleccionados o la inferencia de la "standalone question", hasta la generación de la respuesta por parte del LLM.

En estas conversaciones se puede ver como el LLM confunde el contexto que aporta el retriever con los síntomas del paciente. Esto es, no sabe diferenciar que es lo que dice el usuario y la información complementaria que aporta el retriever. Esto puede ser debido al prompt template que se está utilizando.

6 Evaluación

Para la evaluación del chatbot se ha optado por una evaluación en Q&A, pues una evaluación del chatbot es algo bastante complejo, ya que habría que realizar una evaluación en una conversación real y comparar de alguna manera cuanto se parece la respuesta que esperamos y la generada por el LLM, además realmente no existe una respuesta mejor que otra.

Para la evaluación en Q&A, se ha utilizado el dataset MedMCQA Multi-Subject Multi-Choice Dataset for Medical domain (Pal et al., 2022)

Se ha realizado la evaluación en la partición de validación completa, pues el conjunto de test no está etiquetado. Consta de alrededor de 4000 instancias de preguntas y respuestas, con 4 opciones de respuesta. No obstante, algunas preguntas tenían varias respuestas correctas, por lo que se han eliminado del dataset, reduciendolo así a 2000 instancias.

Para conseguir que el LLM responda una de las cuatro opciones, se ha utilizado un prompt template específico con un format instruction StructuredOutputParser, que obligaba al modelo a responder con un json con la estructura {"answer": "a,b,c o d"}.

Se han tratado de comparar el LLM solo con el LLM con RAG. El LLM solo ha obtenido un accuracy 0.140625, mientras que el LLM con RAG ha obtenido un accuracy de 0.182888, lo que supone una mejora del 30% por usar RAG.

Aunque puedan parecer unos resultados muy malos, y verdaderamente lo son, pues un clasificador aleatorio lo haría mejor, hay que tener en cuenta que en muchas ocasiones el LLM decide no responder a la pregunta, lo que se considera una respuesta incorrecta. También hay que tener en cuenta la dificultad de la tarea, en la Figura 4 se puede ver el leaderboard de la competición MedMCQA usando contexto, se puede ver que los resultados tampoco son excepcionales y teniendo en cuenta que nuestro chatbot no ha sido entrenado específicamente para esta tarea (de hecho no se ha hecho ningun ajuste de hiperparámetros ni entrenamiento sobre el conjunto de test) y que aún existe un margen de mejora, se puede considerar un resultado aceptable.

7 Conclusiones y trabajo futuro

Existen muchas formas de seguir mejorando el chatbot, como aumentar la cantidad de conocimiento externo, mejorar la arquitectura del RAG con alguna de las técnicas mencionadas, realizar un fine-







	Model	Code	Test Set	Dev Set
			Acc (%)	Acc (%)
1 March 10, 2022	BERT (Devlin et al., 2019) Base		0.37	0.35
1 March 10, 2022	BioBERT (Lee et al., 2020)		0.42	0.39
1 March 10, 2022	SciBERT (Beltagy et al., 2019)		0.43	0.41
1 March 10, 2022	PubmedBERT (Gu et al., 2022)		0.47	0.43
1 July 17, 2022	InstructGPT zero-shot CoT (Liévin et al., 2022)		0.49	0.49
1 September 23, 2022	VOD BioLinkBERT (Liévin et al., 2022)		0.58	0.63

Figura 4: Leaderboard de la competición MedMCQA usando contexto.

tuning del LLM para conseguir el tipo de respuesta específica que buscamos para una conversación natural, usar modelos aún más grandes ...

En estos momentos, el chatbot solo es capaz de responder en inglés, aunque se puede decir que entiende el español. Para conseguir que el chatbot funcione completamente en ambos idiomas, castellano e inglés, habría que contemplar la traducción de la información recuperada y cambiar el prompt template o realizar una traducción de la respuesta del LLM.

En cuanto a los resultados de la evaluación, sería interesante contemplar varias plantillas de prompts para ver si se consigue mejorar la respuesta del LLM.

Se ha realizado una evaluación en Q&A, pero sería interesante evaluar el modelo como chatbot, que es el objetivo real de este trabajo, para ello habría que realizar una evaluación en una conversación real y comparar de alguna manera cuanto se parece la respuesta que esperamos y la generada por el LLM. No obstante, esto sigue siendo algo bastante complejo porque realmente no existe una respuesta mejor que otra.

Para seguir mejorando la calidad de las respuestas del chatbot, habría que encontrar una manera de que el chatbot sea capaz de diferenciar entre la información que aporta el retriever y la información que aporta el usuario, ya que muchas veces asocia la información recuperada con síntomas que realmente el paciente no tiene.

8 Material suplementario

- [VectorStore](#)
- [Notebook](#)

- [Conversaciones](#)
- [Kernels Kaggle](#)

Referencias

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey.](#)

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. [Jina embeddings 2: 8192-token general-purpose text embeddings for long documents.](#)

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b.](#)

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. [Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering.](#) In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikula, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech

Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. [Gemma: Open models based on gemini research and technology](#).