Chatbot für das Deutsche Recht

Dogu Tamgac
Technical University of Munich
dogu.tamgac@tum.de

Max Fest
Technical University of Munich
ge34gub@mytum.de

Levente Tempfli
Technical University of Munich
lev.tempfli@tum.de

Abstract

This project introduces Chatbot für das Deutsche Recht, a tool designed to act as a "first responder" for addressing legal issues within the German legal domain, specifically aimed at aiding laypersons, including foreigners. Utilizing state of the art Large Language Models, the Chatbot allows direct interaction with legal information, bypassing intermediate steps, like online forums and or lawyers. It provides clear, accessible legal guidance in both German and English tailored to user queries and legal contexts.

1. Introduction

The progress in Natural Language Processing (NLP) changed the way we use the technology and how we gather information from it. In particular, ChatGPT [9] showed capabilities in understanding and generating human-like text. One can walk into a library of university and observe that most of the students are leveraging the capabilities of the new advancements. This expansion of the field of user-friendly applications has opened up new ways for applications in many domains, including but not limited to education, customer service and notably our topic the legal sector.

The legal domain, known for its complexity and the specialized language, often presents a significant challenge for laypersons trying to understand and navigate through legal matters. The need for clear, accessible and understandable legal assistance is important, particularly for those who may not have the means or access to professional legal advice. The Large Language Model based Chatbots offer a promising solution to bridge this gap, providing a direct, user friendly platform for legal questions and understanding.

This project's main goal is to leverage the advantages of Large Language Models (LLMs) specifically for the German legal domain. Our motivation can summarized as follows: To build a "first responder" for simple legal issues, without the need for lawyers, which is also leveraging the real legal information as its knowledge base.

By implementing this, Chatbot für das Deutsche Recht demonstrates how AI can simplify legal information, while making it more approachable, accessible and user oriented. In the upcoming sections we will delve into the Chatbot's design, functionalities and its

limitations.

2. Problem statement

The problem that we are trying to tackle, as already mentioned, is to bridge the gap between laypersons and the complex German legal system by building an intuitive tool that offers clear concise legal guidance in both German and English. As legal questions tend to be difficult to even formulate, let alone find answers to, we want to provide the layperson with a new option. LLMs and Retrieval Augmented Generation provide the fundamental building blocks for a candidate solution for this problem, which we wanted to evaluate.

2.1. Overview of the Pipeline

The pipeline can be divided into two segments: data preprocessing and legal document preparation, followed by application usage in production. The first segment (the initialisation process) can be seen in the figure 1, which is visualized as a box on the left hand side of the figure. This process, includes scraping the most recent German laws from the web, and preparing them for the usage in the application. This scraping method is detailed in the Section 3.

The laws then undergo a preprocessing step suitable for Referenced Lookup and Vector Database integration for semantic retrieval, elaborated in Sections 5.1 and 5.2, respectively.

These capabilities were then integrated into the UI, allowing the user to interact with the german law in a novel way, as detailed in 4.

3. Related Works

Due to computational constraints and convenience, we opted for OpenAI's API, both for generating embeddings and for the LLM. This limited us to improving our interaction with the model, rather than improving the model itself (e.g. via finetuning, though there is a simple interface for this). Our preliminary research therefore focused on prompting techniques, and Retrieval Augmented Generation (RAG). A survey of commercial LegalTech products showed that where LLM's are used, they are squeezed into narrow problem spaces.

3.1. Prompting techniques

The interaction with an LLM is governed by the context-limited prompt. All the information relevant to the response must be squeezed into this context. There is a wide array of possible approaches for this, including e.g. Chain-of-Thought [19] (prompt the LLM to make reasoning steps), Few-shot learning [8] (provide positive examples), or Prompt chaining [20] (use multiple prompts in response to a single user prompt. Due to the proliferation of the chatbot LLM interface (most popularly OpenAI's ChatGPT), there is a large lay corpus of *Prompt Engineering* techniques, though these are (like the published results mentioned) difficult to assess effectively.

3.2. Retrieval Augmented Generation

We view RAG through the lens of *in-context learning*, i.e. purely as a means of providing more information in a prompt [17]. Important considerations include how the documents in the retrieval database are chunked, how the relevance of a chunk to a given piece of text is modelled, and how this chunk is then included in the prompt. Including RAG immediately prescribes two metrics:

1. Specificity: how well does the retrieved source match the question?

2. Factuality: how well does the response match the retrieved source?

3.3. LegalTech

There is a number of commercial LLM-based legal tech products (e.g. [1–4]). Most of them are based on OpenAI GPT4, though e.g. [3] uses Anthropics Claude because of its longer context length. These products target law firms, and aim

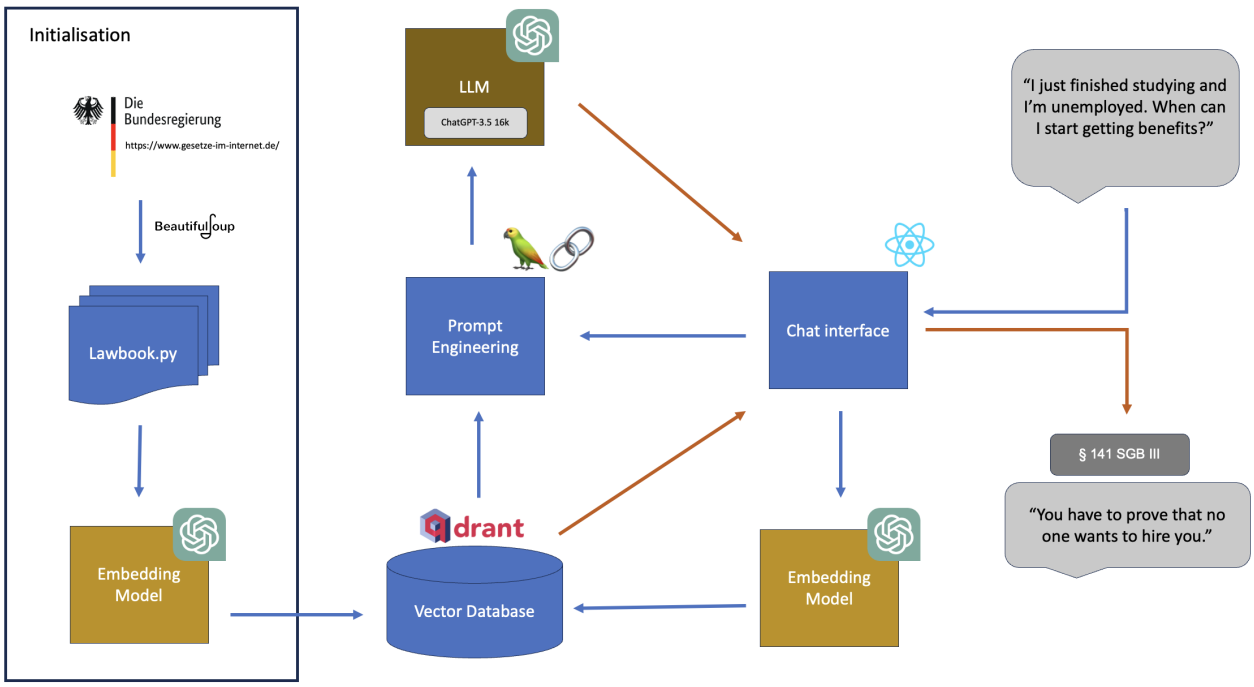

Figure 1. Overview of the pipeline for our Chatbot.

to make specific, repetitive use cases more efficient (e.g. drafting/reviewing contracts, summarisation, semantic search, ...). They are all aimed at legal professionals, that can verify LLM responses. CaseText ([2]) published a series of blog articles summarising the basic techniques used to combat GPT4 hallucinations, most of which we employ in our work. As far as we could tell, there are no commercial LLM-based legal tech products aimed at laypeople.

3.4. Case Study: Italian Legal Text Summarisation

The PRODIGIT project [11] provides a case study of using LLMs to support Italian tax judges and lawyers by generating summaries of legal texts, exemplifying the complexities of creating such a system while also ensuring some level of confidence in it. The authors find that LLMs are far superior to previous NLP approaches, and find that GPT4 outperforms a language specific model. The goal of this project is ultimately to deliver a product for the average Italian citizen.

4. Data

In order for our bot to have the knowledge necessary to answer legal questions, we needed to find a way to first find a digitized version of the German laws, then download it and transform it according to our needs. One possible option is [5], here we could access the German laws via API calls and get back the paragraphs in a structured format. Although very appealing because of the ease of use, we decided against it because the project has not been actively developed since 2019, so the laws might be quite outdated. And finally, we decided to use [6] because it's the official website of the German Federal Ministry of Justice, so it also contains the latest changes.

The difficulty with the site is that it does not provide an API or other way to retrieve the laws in a structured way, so we had to scrape it ourselves. We used the Beautiful Soup Python library to do this. Fortunately, the laws on the site are at least a little bit structured: there is a page with the title and a link to each law book; the pages of the law books contain their table of contents, but also a

link to a page that contains the whole book; this page with the whole book contains all the paragraphs; each paragraph title and paragraph has been assigned a different html class. Under these circumstances, it was straightforward to just get the links to all the lawbooks, iterate through them, download their full page one by one, and simply filter the components by HTML class. In the end, we had all the books in a structured format that we could easily save with Pickle and reuse later for the embedding generation and the retrieval.

5. Frontend

We decided to work with a fully-fledged UI framework early in the project, because we did not want to be limited by this choice later on. We also hoped to ultimately create something user-facing, for which an intuitive UI is crucial.

React was a sensible choice. The alternative were more limited, python-based frameworks, like Gradio or Streamlit, since we intended to write the backend using python. However, the scope of the project was not clear at this point, and some features we had in mind already would be difficult to achieve with such simpler UI frameworks.

For example, we retrieve relevant laws for every user query, which effectively determine how reliable the chatbot response is. A more *grounded* response leans more closely on the retrieved laws, offering a simple method for evaluation. Other features, like translations, questions about specific laws, or referenced law traversal all required a custom interface. We deemed this an important aspect of making LLM-based applications more powerful and more user-friendly.

We could likely have saved some time on the frontend by using a simpler UI framework, opting for faster development cycles and prototype iterations, and a tighter integration between front- and backend.

6. Laws Retrieved

Since law possesses high complexity, navigating through it and retrieving the related laws is a delicate task. Proper interpretation will be done using a Large Language Model backbone, however, it is also essential that the provided answers are grounded in the relevant laws. This is crucial within legal systems, as accuracy and interpretability of the answer is immensely important. Even if a LLM is state-of-the-art, its effectiveness is limited without access to specific, related legal information. Without this crucial data, the model cannot accurately interpret questions nor fully leverage its advanced capabilities. Therefore we proposed two solutions, “Referenced Lookup” and “Embedding Similarity Search”, which will be explained in the upcoming sections.

6.1. Referenced Lookup

An important component of the law retrieval system is the ability to detect when a user’s text contains references to the law itself. This can be simple direct questions such as ”What does § 288 BGB say?” when users ask directly about a law. In this case, the vector search would have absolutely no way of finding the referenced paragraph, so we need to take a different approach to detection. Another possibility is that users can feed the chatbot with legal texts, such as court decisions, and ask it to analyze them. Again, it’s critical that we search and resolve the references in the legal text, and then feed the results to the chatbot so it can respond more accurately.

Since these law references usually follow a rule for how they should look, namely § paragraph_nr Abs. subsection_nr lawbook (e.g. § 256 Abs. 1 ZPO), it’s a perfect use case for regular expressions. So every time we get a request, we first run a regular expression on the query text to extract any references. Then, for each reference separately, we try to look up the corresponding law in our dataset based on the book name, paragraph number, and subsection number. Any laws found by this regex-based lookup are then fed into the

query itself, so that the LLM can answer the questions based on the references. In addition, the referenced laws are also sent to the frontend for the user to view.

6.2. Embedding Similarity Search

One of the main components of RAG systems is similarity search. The core idea in this concept is that chunks of text, in this case laws, are converted into embeddings. This embeddings can be thought of as features or representations, which describe the text inside the chunk. Leveraging this concept, our system can efficiently match user queries with the most relevant legal texts by comparing their respective embeddings, with the metrics such as cosine similarity, L2 and HNSW. This method ensures that the retrieval process is not just based on keyword matching but also on the semantic similarity between the user's questions and the potential legal information available.

To realize this idea, we used the "text-embedding-ada-002" embeddings from OpenAI [15], combined with a Vector Database called Qdrant [7]. However, creation of the embeddings and their subsequent storage in the Vector Database required approximately 90 minutes on a standard commercial laptop, posing a significant efficiency issue. To address this, we pre-generated the embeddings before application use, allowing immediate access without additional user input. One important remark about the new models, that we need to state, is that despite experimenting with OpenAI's newer and most powerful "text-embedding-3-large" embeddings, we did not empirically observe significant improvements in performance or accuracy in our context other than acquiring more costs.

6.3. Additional Features

In our goal of enhancing the usability and efficiency of our legal Chatbot, we have integrated several additional features designed to address specific challenges encountered by the users. These features aim to improve the user experience and also improving the relevance of the search results. Below, we will introduce two significant enhancements:

Translation and Flashrank, each developed to tackle different aspects of legal information accessibility and retrieval precision.

Translation

One of the additional features is targeted for the users who are not proficient in German language. As the German laws are complex and challenging to understand even for native speakers, we integrated a translation functionality for the retrieved laws to support individuals lacking sufficient German language skills. This feature is particularly beneficial for target groups such as foreign students and immigrants, making the laws more understandable and accessible to a broader audience. Implementation of this feature was at first done by a lightweight open-source model by Meta, known as "wmt19-de-en" [16]. However, due to the extended processing time of running this model locally, compared to utilizing the OpenAI API, we decided for the GPT models from OpenAI. This not only improved the efficiency but also allowed for translation through a structured prompt. Importantly, this approach enabled us to preserve the unique formatting of law citations within the text, ensuring that the Referenced Lookup functionality remained intact.

Flashrank

A critical decision in our project involved enhancing the retrieval of relevant laws from the vector database, as the relevancy of gathered laws had a huge impact on the behaviour of the model. Therefore, we introduced an additional feature called Flashrank, which is a highly efficient and lightweight Python library designed for re-ranking within our search and retrieval pipeline [12]. It is based on state of the art cross-encoders. To incorporate this re-ranking process, we ex-

panded the capabilities of the Langchain [10] library. It's important to note that the Langchain library does not directly have an integrated solution for re-ranking with Flashrank, driving us to make several design decisions. The idea that we came up with was to apply the re-ranking functionality on top of the relevant laws. By doing so, we integrated another layer of decision making in which we hoped to increase the accuracy of the retrieved laws. To make more accurate decisions, more laws were retrieved from the Vector DB. Then the final decision of which law to consider was made based on the re-ranking algorithm, where the laws were ranked by their relevance score to the query. The laws that were not similar enough were pruned with respect to a threshold, which is a hyperparameter, as they lacked sufficient similarity according to Flashrank. This allowed us to have a consequent decision mechanism for deciding which laws to take into account.

7. Prompting

The effectiveness of a Chatbot, which is leveraging a LLM, is measured by its ability to understand and respond to user queries in a contextually relevant manner. This effectiveness is highly influenced by the Chatbot's initial prompting strategy [14]. For reference, our prompt for guiding the LLM, for the user queries is as follows:

You are a chatbot for the german law, your purpose is to simplify this complex domain for the user, who is likely to be a layperson. The user might ask questions that aren't strictly law-related, that is not your purpose. You can respond briefly, but remind the user that you are a legal chatbot. If the user persists, stop responding. Answer according to the earlier conversations: {Chat History}. The user may ask a question in german or english. Respond in whichever language the user chooses. You will also receive a set of laws which might be relevant to the users current query: {Relevant laws from the Vector Database}. They might not be relevant, so make your own judgement how strongly to lean on these for your answer. Ideally, the laws are very relevant, and you base your answer entirely on them, but this is unlikely to be the case. If the laws are irrelevant, or simply don't contain much information, just answer from your own knowledge to the best of your ability. The text may cite specific german laws, you may use these as a basis for your answer as well: {Referenced Lookup}

This prompt sets the tone for the interactions and guides the LLM towards effective communication. By refining our Chatbot's prompt, we aimed to reduce complexities of the law for the user and provide clear, concise and relevant legal information. Additionally, it is also important that, since we are building a legal Chatbot the nonlegal entries also should not be considered, to maintain the focus of the Chatbot on providing the legal assistance. Furthermore, as previously mentioned in the translation section, it is essential for the LLM to handle both English and German, therefore, we have incorporated this capability into the prompt as well. Moreover, the prompting strategy is built to adapt to the user's input. Meaning, it acknowledges the laws provided from the Referenced Lookup and Vector Database and utilizes them to guide its answer. However, it also assesses the relevance of the retrieved laws from the Vector Database according to the user's queries. In instances where the provided laws are not applicable, the Chatbot uses its broader knowledge base, which was acquired while training, for answering to the best of its ability.

By implementing this approach in our prompt, we not only improve the user experience but also ensure that the language model remains focused on the legal domain.

8. Experiments

This section will start by presenting and discussing the experiments we conducted. It will include an analysis of the user questions. Later in the section limitations and accuracy of our approach will be evaluated

8.1. Deployment

Before we did the poster presentation of this project, we deployed the app on a home server with the intention to make it easily available. Then, during the poster presentation, we attached a QR code next to our poster with the hope of gaining some test users and we could analyse their questions.

Overall, there were 7 users who interacted with our applications. Of these 7 users, 3 asked in German and 4 in English, but each time the bot answered in the language of the question. 5 of the interactions were just one question and one answer and the other 2 were a longer interaction with 4 question and answer pairs.

In general, all the questions were very short, some just short sentences. Most of the questions were fairly reasonable, such as "Marriage without German citizenship", "I've just finished my studies and I'm unemployed. When can I start receiving benefits?" or "Do you pay income tax as a self-employed person?" . However, some questions were a bit silly and may have been just to play with the bot, such as "Can I legally kill someone" or "I am unemployed, give me money".

8.2. Limitations and Accuracy

In our experiments, we tested a variety of prompts to understand the Chatbot's performance range. Despite multiple trials, we observed a plateau in the output quality after a certain point, indicating a limitation of our retrieved laws or the model's capability.

A significant challenge we encountered was the absence of an established benchmark for evaluating performance in the context of German law. This lack of a standardized measure for evaluating the Chatbot, made it extremely difficult for assessing if the answers were getting better with the changes we were making, particularly because we completely lacked legal expertise.

Moreover, our experiments were confined to OpenAI's language models due to hardware limitations, preventing us from testing alternative large language models such as Mixtral 8x7B [13] or LLama2 [18]. Even if there are quantized 4bit models of LLama2 which can be run on some of the commercially available laptops, its performance is not up to par with OpenAI's solutions.

Another critical limitation is the context length. The problem with it is that if we gather too much information from the Vector DB and Referenced Lookup the LLM was not able to handle it as it was exceeding the context length. Therefore, we split some paragraphs in multiple parts so that they can fit in the context length.

Furthermore, we did not opt for the GPT-4 model as the response times are significantly slower compared to the GPT-3.5 based models. As the knowledge was already gathered from the Vector DB and Referenced Lookup, we did not observe huge response accuracy differences with respect to our knowledge of the law.

8.3. Retrieval

Our aim was to squeeze the LLM into a narrower space by *grounding* its answers in authentic german legal texts. However, we did this simply by computing cosine similarity between embeddings of the user prompt, and legal texts. This approach is fundamentally flawed, in that user prompts are typically sparse, and contain little legal information. We could likely have improved results by chaining multiple prompts, e.g. using the LLM to transform the user prompt into a plausible legal paragraph, and then retrieving articles similar to this hypothesis.

Similarly, when the user asked questions about a specific legal term, the LLM either provided the answer outright, or retrieved the answer via embedding search. Some form of hybrid search with more classical NLP algorithms would clearly have been more appropriate in this case.

When the retrieved legal texts were not relevant, the model answered poorly. However, it was clear from the beginning that letting it answer without retrieved sources was insufficient, primarily because this removes our only evalua-

tion metrics (see section 3.2).

9. Conclusion

Overall, while the Chatbot demonstrates potential in providing legal assistance, the limitations, that were mentioned underscore areas for future improvement and research, particularly in enhancing the Chatbot's retrieval system in the complex domain of law.

However, we would also like to emphasize some of the methods for future research. Here some of the relevant methods that would potentially help with the Chatbot's efficiency, accuracy and also data protection for users will be analyzed.

Starting with one of the obvious approaches for increasing the accuracy and aligning the domain for the LLM is finetuning or can be easily called as further pretraining in this context. This approach allows the LLM to have a more specific knowledge about the law, as one would use domain-specific data for training. So having this knowledge embedded in the weights of the LLM would serve as filling in the gaps for the retrieval system, or outright being able to answer the user queries. However, finetuning using a huge dataset such as law, would inquire a lot of costs, if one would decide to it on the OpenAI API. If one would decide to do the training on the cloud or locally while using an open-source model such as Mixtral 8x7B, a lot of computing resource will be needed, which we did not have. If one would be able host their own models, this would also mean that the data protection and security would be significantly improved, as it is a huge issue in law. One should not prompt his or her information, while using the Chatbot. If the user has delicate questions about breaking the law, committed crimes and its consequences, this information potentially can be used for training the model further and storing it. A further approach can be using a pseudonymization process on top of the prompt.

References

[1] https://contractpodai.com. 2

[2] https://casetext.com/cocounsel/. 2, 3

[3] https://www.robinai.com/post/copilot-your-legal-ai-assistant. 2

[4] https://www.lawgeex.com/platform/managed-ai/. 2

[5] https://de.openlegaldata.io/. 3

[6] https://www.gesetze-im-internet.de/. 3

[7] Qdrant, https://github.com/qdrant/qdrant, 2023. 5

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. 1

[10] Harrison Chase. LangChain, Oct. 2022. 6

[11] Thiago Dal Pont, Federico Galli, Andrea Loreggia, Giuseppe Pisano, Riccardo Rovatti, and Giovanni Sartor. Legal summarisation through llms: The prodigit project. *arXiv e-prints*, pages arXiv–2308, 2023. 3

[12] Prithiviraj Damodaran. FlashRank, Lightest and Fastest 2nd Stage Reranker for search pipelines., Dec. 2023. 5

[13] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. 7

[14] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021. 6

[15] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pretraining, 2022. 5

[16] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*, 2020. 5

[17] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023. 2

[18] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou,

Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 7

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 2

[20] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–10, 2022. 2