Exploring Large Language Models and Retrieval Augmented Generation for Automated Form Filling

MATEI BUCUR, University of Twente, Netherlands

Large language models (LLMs) such as the GPT family have shown remarkable natural language processing capabilities across a variety of tasks without requiring retraining or fine-tuning. However, leveraging their potential for use cases beyond the traditional chatbot paradigm remains an open challenge. One potential application is automated form completion, which enables users to fill out online forms using natural language and leverages available data about the user and the form completion guidelines. This can benefit a broad range of processes, such as applying for a loan or grant, filing a tax statement, or requesting a service. However, automated form filling faces challenges such as understanding form layout, guidelines, and user intent, as well as reasoning over data, in order to generate accurate and coherent text. In this paper, I propose a general method for adapting LLMs to different form-filling domains and tasks. The method consists of three steps: (1) creating a knowledge base that contains facts and rules related to the form-filling task; (2) augmenting the LLM with the knowledge base using retrieval-augmented generation; and (3) using prompt engineering techniques to improve the outputs. I evaluated the effectiveness of the method and the impact of the techniques on the task of completing request forms for various incentives and services.

Additional Key Words and Phrases: Large language models, gpt3.5, retrieval augmented generation, form filling

1 INTRODUCTION

1.1 Motivation

Forms are an essential way to extract information from people and have been used for a long time in various domains and contexts. They are a tool used to collect information in a structured and standardised way by means of a printed document with spaces in which answers to questions can be written. [3] Forms are needed for interacting with different institutions. Several industries, like healthcare, finance, government, law, and education, rely heavily on their usage. Every person faces a high likelihood of having to deal with form filling throughout their life, as it is a pervasive activity needed in common endeavours such as reporting taxes, applying for a credit card, or registering insurance. I have identified five types of forms based on their purpose and format: request, registration, consent, evaluation, application, report, and declaration. Form filling is typically not a creative task but a functional one in which accuracy, compliance with guidelines, and attention to detail are valued. However, it is common for people to make mistakes while completing them. According to an expert in grant applications, common mistakes include not following the instructions, not providing enough detail, and not explaining the significance of the project [9, 10]. Therefore, the task of form filling is considered to be time-consuming, bureaucratic, and tedious[4, 31].This could cause users to abandon the form or provide inaccurate or incomplete information. A survey found that 81% of people discontinued filling out an online form midway through the process, and out of those, 67% of them chose to terminate the process completely if they encountered any difficulties[2].

In addition to filling out forms for personal use, this process is common and crucial in the business world, 59% of US workers having to use forms in their jobs[35]. Forms are frequently employed as a tool to organise and standardise the data that organisations must gather, handle, and analyse from multiple sources and stakeholders. Businesses deal with a variety of unstructured, dynamic data sources that must be accessed and comprehended. Even though searching is now commonplace, the problem has not yet been resolved. An excellent illustration of this is enterprise search, which is, roughly speaking, the use of information retrieval technology to find information within organisations. Despite having great financial value, the topic has received little academic attention. [18] In order to be able to fill in forms accurately, users must obtain and grasp information from numerous sources, such as documents, databases, or web sites, about the form guidelines and requirements. This requires strong enterprise search capabilities. Enterprise search faces unique challenges and problems that make it notoriously difficult to achieve user satisfaction and have significant economic importance[13]. One of the challenges identified is that context-aware search is a challenging problem for enterprise search systems, where users often have diverse and dynamic information goals that depend on various aspects of their work context.

In the digital age, online forms are the most common method of data collection since they allow for inexpensive and effective storage, processing, and analysis. However, online forms also pose some challenges and limitations for users. To help improve the efficiency of form filling, web browsers introduced the autofill and autocomplete functions[1]. The autofill functionality fills in the fields of a web form based on previously entered data, while the autocomplete functions suggest or complete the fields based on what the user has typed. While those are successful in easing the completion of simple fields, they do not consider the broader context of the form and the user and are not able to assist in more complex ones.

To overcome the limitations of existing solutions, we require a flexible system that is able to understand the context, requirements and generate text. In this context, Natural Language Generation (NLG), a sub-field of natural language processing, is relevant, as it aims to generate text using input data (prompts, tables, images, etc.). [11]. The transformer is a neural network architecture that leverages self-attention to encode and decode sequences without using recurrent or convolutional layers. It was first presented in [33] and has since been widely applied to NLP tasks. This innovative design

TScIT 37, July 7, 2023, Enschede, The Netherlands

laid the foundation for the large language models we have nowadays. By leveraging the transformer architecture and scaling up data and computation, large language models (LLMs) have emerged as powerful neural networks that can generate natural language for a wide range of domains and tasks.[36]

The growing interest in LLMs such as ChatGPT and GPT-4 is evident from the increasing number of papers on arXiv that explore their potential applications and ethical implications across diverse domains [23], making it a valuable technology that is still in the early stages of adoption.

In this research, I propose the Forms Copilot: a Large Language Model-based System for Form Autocomplete to provide context-aware completions for form fields that the user can review and modify, thereby enhancing efficiency and easing the challenge of filling out forms for users. Therefore, I address the following research question:

RQ: How can we design a large language model based system that leverages and adapts the model's pre-trained knowledge with retrieval-augmented generation of form completions in the context of enterprise search?

1.2 Related work

OpenAI's GPT (generative pretrained transformer) models have repeatedly outperformed other LLMs in a variety of natural language tasks, indicating their unrivalled ability to generate coherent and diversified output[39].By combining a massive amount of data, a Transformer-based architecture, and a post-training alignment process, GPT-4 achieves unprecedented natural language generation and understanding capabilities, surpassing previous language models and most state-of-the-art systems on a variety of tasks and exams. It exhibits human-level performance on various professional and academic benchmarks, such as passing a simulated bar exam with a score in the top 10% of test takers [26]. GPT-4 exhibits more general intelligence than previous AI models, as it can perform tasks that require complex and logical inference that involves knowledge and reasoning in addition to linguistic processing [6].

Large language models offer unprecedented opportunities for businesses to improve their goods and services. There have been investigations on their potential use to enhance process management, as they can assist users in various tasks, such as process discovery, analysis, redesign, implementation, and monitoring, by generating natural language responses, explanations, and suggestions. Even though there are challenges and limitations, the adoption of LLM in commercial products is expected in the future[34].

Knowledge-intensive tasks pose significant challenges for natural language processing as they require models to access, understand, and utilise a large and diverse amount of background knowledge, domain-specific expertise, or general real-world knowledge. ChatGPT and GPT-4 demonstrate remarkable capabilities in these tasks [39]. However, they rely on their knowledge, which is limited by the data they were trained on and may not cover the specific and dynamic information that is relevant for a given task or domain. Moreover, they may generate incorrect or inconsistent information, known as hallucinations, due to the lack of external verification or feedback[8, 25]. These issues can limit the applicability and reliability of LLMs for form filling, where the information needs are heavy and the data that needs to be inputted in the form is most probably outside the scope of the LLM's training data. To address these challenges of complex and creative tasks, AI copilots leverage generative models that can interact with users via natural language queries and dialogue, synthesising answers from multiple sources and domains[38].

One way to enhance LLM's knowledge and leverage various sources is retrieval-augmented generation (RAG). RAG is a language generation technique that combines pre-trained parametric and non-parametric memory[14, 21]. It combines a LLM with a retrieval mechanism that can access external documents to generate responses for knowledge-intensive NLP tasks. RAG consists of two main components: the retrieval mechanism and a generator. The retrieval mechanism is the way of finding relevant information from a collection of documents based on the user's query.

Several studies have proposed techniques to automate or assist the process of filling out forms, especially web forms.

One related task that is similar to form filling is patent claims, legal statements that define the scope of a patent. A method to generate these by fine-tuning was proposed by [20], and has demonstrated capabilities of coherent generation. However, the work is considered a first step towards auto-completion functionality.

There have been machine learning approaches for automated form filling, one of which involves filling out categorical fields in data entry forms[4]. Their approach, called LAFF(Learning-based Automated Form Filling) uses Bayesian Networks (BNs) to learn field dependencies from historical input instances of forms. LAFF also applies local modelling to cluster input instances and builds additional local BNs that capture fine-grained field dependencies. During the form filling phase, LAFF uses the BNs to predict possible values for a target field based on the values in the already-filled fields and their dependencies. LAFF also includes a heuristic-based endorser that decides whether the predicted values are accurate enough to be suggested to the user. The authors showed that it can provide accurate and efficient suggestions. However, their approach is limited to categorical fields and does not handle textual or numerical fields. Moreover, their approach relies on the availability and quality of historical input instances. Furthermore, their approach does not consider the context or intent of the user or the form guidelines when generating suggestions.

1.3 Contribution

In this research, I make several contributions to the field of natural language generation and form filling.

- To the best of my knowledge, this is the first work that adapts LLMs for the task of form completion, which has many practical applications and challenges
- I design and implement a system that leverages and adapts the pre-trained knowledge of an LLM, namely GPT-3.5, for retrieval-augmented generation of form completions
- I develop a web extension that can scrape and fill fields on various websites, as well as a backend architecture that uses

FAISS indexing to retrieve relevant information from a knowledge base and generate structured text using GPT-3.5

- I demonstrate the capabilities of the LLM to reason over unstructured data and generate coherent and accurate text for different types of forms
- I apply prompt engineering techniques to improve the quality and diversity of the outputs and evaluate the impact of each prompt variation on the results
- I test the system on a dataset of complex request forms and show that it can provide a useful draft that is close to the ground truth, thus offering assistance and productivity improvement for users
- I propose a general method that can be adapted to any online form filling task with minimal changes, making the system flexible and scalable

2 METHOD

2.1 Retrieval augmented generation

The proposed system's RAG is composed of GPT 3.5 as the generator and a FAISS index as the retriever. Although GPT-4 would be the ideal model for the research, its availability is limited, and I could only obtain a small sample of its outputs that is suitable only for qualitative analysis. Therefore, I will mainly rely on its precursor, GPT-3.5, for text generation[5]. The retrieval mechanism will be based on embeddings[32], a technique used in NLP to represent words or phrases as vectors of numbers. FAISS (Facebook AI Similarity Search) is an open-source library for efficient similarity search and clustering of dense vectors [16, 17]. FAISS generates an index structure that organises the vectors in such a way that searching for related vectors is quick and efficient. Given a query vector, FAISS searches the index structure for the most similar vectors. The first step is to split all the documents into chunks (paragraphs), generate their corresponding embeddings, and store them together in a database. The form fields are encoded into a vector representation. Then, the retrieval mechanism is used to find the documents that are most similar to the query vector, based on Euclidean distance. The retrieved chunks are then fed into the generator, along with the rest of the prompt, to generate a response that incorporates the relevant information from the documents.

2.2 Prompt engineering

Prompts are the main means of communication with LLMs and are provided to guide the model's output generation[7]. Prompt engineering has emerged as one significant way to improve the output of the models, and is a reliable alternative to finetuning[5]. One of the most simple and effective techniques that has been shown to significantly improve the ability of large language models to elicit complex reasoning is chain of thought prompting(CoT)[37]. It involves breaking down a complex task into sub-tasks, a behaviour achieved by instructing the language model to think step by step. Also, few-shot prompting is a technique used to guide the models into generating more desirable responses by providing a small number of examples as part of the prompt[5, 29].

3 IMPLEMENTATION

I propose a cloud-based full-stack architecture inspired by [30] that leverages large language models and an information retrieval system to provide context-sensitive and smart autocompletion of web-based forms in a user-friendly interface facilitated by the means of a browser extension. One of the features of my architecture is its flexibility and adaptability to different use cases and scenarios. The system's usage frequency and document sizes influence the decision to index all documents or only the ones about guidelines and rules. All documents can be indexed for one-time, highly personalised use. For regular use by different users, only guideline documents can be indexed, and user-specific documents can be directly passed and added to the prompt.

3.1 Frontend

The frontend of the Forms Copilot is implemented as a chromium web extension, allowing users to interact with the system seamlessly and access and alter the contents of most of the websites that host forms. The extension triggers the execution of two major functions: the creation of the vector database of guidelines for form filling and the form completions based on the user's input and form fields. First, the user is able to upload all the documents that may be relevant to that specific task of form filling, such as FAQs, policies, guidelines, and rules. These documents are sent to the backend to create of the search index. Also, it encourages the user to provide a brief description of the intent and purpose of the form and add extra short-form documents to the system. For example, the research proposal can be relevant in the case of applying for a grant. The extension scans all the input fields and labels of the form and sends that, along with the user data, to the backend for form completion. The extension attempts to fill in the input fields, and in the case of forms that do not allow autofill, it displays the results in the extension window so the user can copy the answers.

3.2 Backend

The first component of the backend is an Azure function[12] which has a public endpoint that can receive POST requests with the documents needed for the creation of the search index. It first stores the files in their original form in an Azure blob storage container. The container can be reused later if additional files need to be added or modified. Each file is converted to plain text and split into chunks of maximum 1000 characters using the text spliter function of Langchain [19], a widely used framework that simplifies the development of LLM applications. For each chunk, a vector embedding is generated using OpenAI's text-embedding-ada-002 model [27] and stored in another Azure blob storage container. Finally, the address of the search index storage is shared with the frontend so it can be communicated with the second component of the backend.

The second component is an Azure prompt flow, a feature in Azure Machine Learning Studio that streamlines the experience of building LLM applications and facilitates the process of prompt engineering[30]. The deployed flow receives the user information in the form of documents, one form field label, and the path of the search index. It then generates the embedding for the label of the form and performs a lookup function in the FAISS search index in

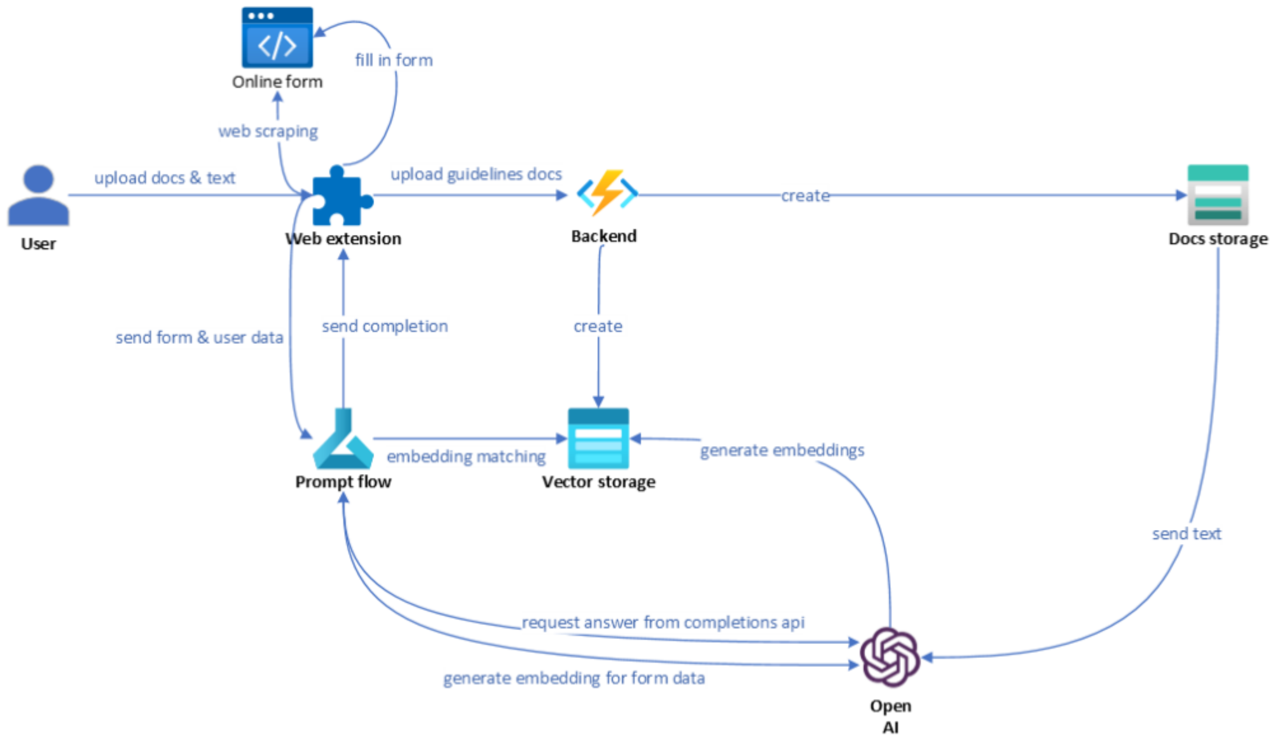

Fig. 1. Architecture of the system: given the user data X (form guidelines), the extension sends it to the backend for the creation of the FAISS index storage; the search index is later used in the generation of the form completion, together with user data Y(documents relevant for the form) and the scraped labels of the web form

order to find the top 3 most relevant chunks to the label. The name of the document for each chunk is appended, and all the formatted chunks are added as context to the prompt. In addition, the text from the user documents is extracted and added to the prompt. Finally, the prompt adds the description of the general form-filling task and the generation instruction that asks the gpt3.5 model to produce the result in the form of a JSON object with three fields: reasoning, sources, and value. After the response is generated, the object is parsed, and the value of the field is returned.

4 EXPERIMENTAL SETUP

The proprietary dataset consists of 50 web page files of request forms for various IT projects, submitted in the last 2 years by different employees. Each form is accompanied by a short document describing the implementation and impact, as well as a collection of documents with general guidelines and rules related to the process. Each form has approximately 30 fields of various complexity, spanning from filling out the name of the requester to justifications and inquiries about the project plan. The fields are mainly text and number inputs but also multiple choice, with slight variations in the field labels depending on the version of the form. Each form and document were processed as a list of JSON objects, with each object representing one form field. The objects contain key-value pairs that represent the id of the form, the field label, the ground-truth value, and the text of the document. Due to the high costs of generating and evaluating the outputs of the generative model, I follow previous works approaches [15] to subsample the dataset, in this case to 287 fields for the evaluation. To execute the evaluation, I ran the experiments on a cloud environment using a Standard-DS11-v2 Virtual Machine (2 cores, 14 GB RAM, 28 GB disc) with a memory-optimized CPU.

5 EVALUATION

Evaluation is a great challenge in this research, and it is in general for LLMs because they can generate diverse and fluent responses that may not be easily comparable or verifiable by existing methods. It is acknowledged that there is a need for metrics that reflect the system's human-like cognition rather than its constrained AI counterparts to assess their performance and intellect [6]. Traditional metrics, such as ROUGE and BLEU[22, 28], focus on lexical overlap with reference texts, which may miss the semantics and complexity of created texts. An alternative is G-EVAL, a methodology for assessing the similarity of NLG outputs that combines large language models (LLMs) with chain-of-thought (CoT)[?]. G-EVAL requests the LLM create comprehensive evaluation stages based on the task and criteria and then score the results in a form-filling paradigm. On two NLG tasks, text summarization and dialogue production, G-EVAL has been shown to outperform the state-of-the-art evaluators and achieve higher human correspondence [24]. However,

Table 1. Evaluation results for different variants of prompts

Variant	f1 score	BERTscore P	BERTscore R	BERTscore F1	G-Eval
0 (vanilla)	0.30	0.43	0.44	0.43	2.37
1 (CoT + cite sources + optimized prompt)	0.31	0.71	0.72	0.71	2.41
2 (CoT only)	0.34	0.53	0.54	0.54	2.38

Table 2. Evaluation results for different variants of contexts

Variant	Evaluation	f1	BERTscore	ROUGE-1	ROUGE-2	ROUGE-L
0	Form guidelines	0.09	0.65	0.11	0.01	0.1
1	Project description	0.23	0.74	0.29	0.07	0.28
2	No supporting docs	0.13	0.72	0.14	0	0.14

that is still in the early stages of development and has shown a bias towards LLM-generated texts. One alternative is BERTScore, an embedding-based metric that compares the contextual embeddings of the generated text and the ground truth[40]. I evaluated the Forms Copilot on the proprietary dataset with the metrics described above.

6 DISCUSSION

In 1, it can be seen the impact of various prompt engineering techniques. The results of the experiment show that the version of the Forms Copilot that leverages all data sources and responses, uses chain of thought prompting, asks the model to cite the sources, and provides a description of the form has achieved an average of 2.41 / 5 G-EVAL score and 0.71 F1 BERTscore, indicating that the suggestions are similar to the ground truth. This indicates that an LLM based approach may be able to generate suitable responses for the form-filling task that can later be reviewed and edited, thus indicating a high probability of being more time-efficient. Furthermore, this shows that the model can benefit from a more detailed description of the task and from breaking it down into sub-tasks. Finally, the model can generate more accurate and relevant information by citing the sources, which also greatly benefits the users by increasing the transparency and explainability of the AI system.

The results in2 show that the documents directly related to the form-filling task have the greatest impact on the quality of the generated result. Also, the form guidelines and rules decrease the performance of the model if they are not accompanied by specific data that needs to be in the form. A qualitative analysis of the results shows that in that case, the model tends to generate more guidelines instead of values, which is not desirable and explains the low score. This shows the importance of providing the right data for the model.

6.1 Limitations

This research presents an architecture and conducts a preliminary experiment with LLMs for the task of form filling. However, I acknowledge that the evaluation is not comprehensive and only covers one domain-specific use case, which may affect the applicability of this approach to other scenarios. Further experiments are needed to ensure the generalizability of the method in other contexts. Also, the generation of accurate results relies on the presence of relevant external documents, which may not be available in the broader context of form filling. The frontend's functionality of extraction of labels and autofilling is contingent on the webpage's implementation and faces many obstacles, such as complex and inconsistent HTML code. Therefore, a more robust frontend component would be desirable for a production-level application. Finally, the system does not address the ethical and social implications of using an AI system for form filling. Further research is needed in order to provide guidelines regarding appropriate use cases as well as potential ethical risks.

6.2 Future work

In the future, I would envision form-filling software that would be adaptable to any form-filling scenario with no code changes. Further research would need to be conducted on a user testing study in order to test if this approach improves productivity, form filling accuracy, and efficiency. Finally, more thorough experimentation is needed to test the impact of different configurations of the retrieval and generation parts of the system. Despite the encouraging results of this research, there is a lot of room for development. Therefore, it is important that academia continue to be involved in the field of generative AI and ensure that LLM applications are developed in a conscious and responsible manner.

RQ: How can we design a large language model based system that leverages and adapts the model's pre-trained knowledge with retrieval-augmented generation of form completions in the context of enterprise search?

Answer: We can design a large language model based system that leverages and adapts the model's pre-trained knowledge with retrieval-augmented generation of form completions in the context of enterprise search by following these steps:

(1) Augment the LMM by providing documents, elaborate descriptions, instructions and user input related to the specific form-filling task directly in the prompt
(2) Use prompt engineering techniques to improve the outputs, such as chain-of-thought prompting
(3) Create a knowledge base that contains facts and rules related to the form-filling task, such as guidelines, policies, FAQs, and user-specific documents.
(4) Augment the LLM with the knowledge base using retrieval-augmented generation

7 CONCLUSION

In this paper, I have proposed an architecture and a preliminary experiment for a form filling application that uses LLMs to generate suggestions for online forms. The software is composed of two parts: a frontend component that reads form labels and autofills forms, and a backend component that creates the form completions using an LLM and external data sources. The preliminary experiment reveals that the LLM-based approach may create appropriate responses for one form filling activity, which can then be reviewed and edited, showing a high probability of increasing the user's productivity.

REFERENCES

[1] 2012. Making form-filling faster, easier and smarter | Google Search Central Blog. https://developers.google.com/search/blog/2012/01/making-form-filling-faster-easier-and

[2] 2018. 6 Steps for Avoiding Online Form Abandonment. https://themanifest.com/web-design/blog/6-steps-avoid-online-form-abandonment

[3] 2023. form. https://dictionary.cambridge.org/dictionary/english/form

[4] Hichem Belgacem, Xiaochen Li, Domenico Bianculli, and Lionel C. Briand. 2023. A Machine Learning Approach for Automated Filling of Categorical Fields in Data Entry Forms. *ACM Transactions on Software Engineering and Methodology* 32, 2 (April 2023), 1–40. https://doi.org/10.1145/3533021 arXiv:2202.08572 [cs].

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 1877–1901. https://papers.nips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. http://arxiv.org/abs/2303.12712 arXiv:2303.12712 [cs].

[7] Hai Dang, Lukas Mecke, Florian Lehmann, Sven Goller, and Daniel Buschek. 2022. How to Prompt? Opportunities and Challenges of Zero- and Few-Shot Learning for Human-AI Interaction in Creative Applications of Generative Models. https://doi.org/10.48550/arXiv.2209.01390 arXiv:2209.01390 [cs].

[8] Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. 2022. On the Origin of Hallucinations in Conversational Models: Is it the Datasets or the Models? http://arxiv.org/abs/2204.07931 arXiv:2204.07931 [cs].

[9] John Ellery. 2013. Avoiding common mistakes on grant applications. *Headteacher Update* 2013, 6 (June 2013), htup.2013.13.6.98917. https://doi.org/10.12968/htup.2013.13.6.98917

[10] John Ellery. 2013. Get that funding – avoiding common grant application mistakes. In *SecEd*, Vol. 2013. sece.2013.6.1783. https://doi.org/10.12968/sece.2013.6.1783 ISSN: 1479-7704 Issue: 6 Journal Abbreviation: SecEd.

[11] Giacomo Frisoni, A. Carbonaro, G. Moro, Andrea Zammarchi, and Marco Avagnano. 2022. NLG-Metricverse: An End-to-End Library for Evaluating Natural Language Generation. https://www.semanticscholar.org/paper/NLG-Metricverse%3A-An-End-to-End-Library-for-Natural-Frisoni-Carbonaro/9f158e69eb2ccf5dce78cd50f4be3cff99b25ca8

[12] ggailey777. 2023. Getting started with Azure Functions. https://learn.microsoft.com/en-us/azure/azure-functions/functions-get-started

[13] David Hawking. 2004. Challenges in Enterprise Search. (2004).

[14] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot Learning with Retrieval Augmented Language Models. http://arxiv.org/abs/2208.03299 arXiv:2208.03299 [cs].

[15] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. http://arxiv.org/abs/2305.06983 arXiv:2305.06983 [cs].

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. http://arxiv.org/abs/1702.08734 arXiv:1702.08734 [cs].

[17] Jeff Hervé Jegou Johnson, Matthijs Douze. 2017. Faiss: A library for efficient similarity search. https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/

[18] Udo Kruschwitz and Charlie Hull. 2017. Searching the Enterprise. *Foundations and Trends® in Information Retrieval* 11, 1 (July 2017), 1–142. https://doi.org/10.1561/1500000053 Publisher: Now Publishers, Inc..

[19] langchain. 2022. LangChain | LangChain. https://docs.langchain.com/docs/

[20] Jieh-Sheng Lee and Jieh Hsiang. 2020. Patent claim generation by fine-tuning OpenAI GPT-2. *World Patent Information* 62 (Sept. 2020), 101983. https://doi.org/10.1016/j.wpi.2020.101983

[21] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401v4

[22] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[23] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. http://arxiv.org/abs/2304.01852 arXiv:2304.01852 [cs].

[24] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. http://arxiv.org/abs/2303.16634 arXiv:2303.16634 [cs].

[25] Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. 2023. Sources of Hallucination by Large Language Models on Inference Tasks. http://arxiv.org/abs/2305.14552 arXiv:2305.14552 [cs].

[26] OpenAI. 2023. GPT-4 Technical Report. http://arxiv.org/abs/2303.08774 arXiv:2303.08774 [cs].

[27] doc openai. 2023. OpenAI Platform. https://platform.openai.com

[28] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[29] Shrimai Prabhumoye, Rafal Kocielnik, Mohammad Shoeybi, Anima Anandkumar, and Bryan Catanzaro. 2022. Few-shot Instruction Prompts for Pretrained Language Models to Detect Social Biases. http://arxiv.org/abs/2112.07868 arXiv:2112.07868 [cs].

[30] promptflow. 2023. Harness the power of Large Language Models with Azure Machine Learning prompt flow. https://techcommunity.microsoft.com/t5/ai-machine-learning-blog/harness-the-power-of-large-language-models-with-azure-machine/ba-p/3828459 Section: AI - Machine Learning Blog.

[31] Enrico Rukzio, Chie Noda, Alexander De Luca, John Hamard, and Fatih Coskun. 2008. Automatic form filling on mobile devices. *Pervasive and Mobile Computing* 4, 2 (April 2008), 161–181. https://doi.org/10.1016/j.pmcj.2007.09.001

[32] Tim Schopf, Daniel Braun, and Florian Matthes. 2021. Lbl2Vec: An Embedding-Based Approach for Unsupervised Document Retrieval on Predefined Topics. In *Proceedings of the 17th International Conference on Web Information Systems and Technologies*. 124–132. https://doi.org/10.5220/0010710300003058 arXiv:2210.06023 [cs].

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. http://arxiv.org/abs/1706.03762 arXiv:1706.03762 [cs].

[34] Maxim Vidgof, Stefan Bachhofner, and Jan Mendling. 2023. Large Language Models for Business Process Management: Opportunities and Challenges. (2023). https://doi.org/10.48550/ARXIV.2304.04309 Publisher: arXiv Version Number: 1.

[35] Paul Viola and Mukund Narasimhan. 2005. Learning to extract information from semi-structured text using a discriminative context free grammar. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 330–337. https://doi.org/10.1145/1076034.1076091

[36] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. http://arxiv.org/abs/2206.07682 arXiv:2206.07682 [cs].

[37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. https://doi.org/10.48550/arXiv.2201.11903 arXiv:2201.11903 [cs].

[38] Ryen W White. 2023. Tasks, Copilots, and the Future of Search. (2023).

[39] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. http://arxiv.org/abs/2304.13712 arXiv:2304.13712 [cs].

[40] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. https://doi.org/10.48550/arXiv.1904.09675 arXiv:1904.09675 [cs].