Group 41 - Final Project Review CSE508 - Winter 2024

Arnav Goel, Aditya Pratap Singh, Ashutosh Gera, Medha Hira, Nalish Jain, Shikhar Sharma
{arnav21519,aditya20016,ashutosh21026,medha21265,nalish21543,shikhar20121}@iiitd.ac.in
IIIT Delhi, New Delhi
India

ACM Reference Format:
Arnav Goel, Aditya Pratap Singh, Ashutosh Gera, Medha Hira, Nalish Jain, Shikhar Sharma. 2024. Group 41 - Final Project Review CSE508 - Winter 2024. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the contemporary digital landscape, users encounter significant challenges in accessing pertinent information and personalized assistance seamlessly across various online platforms. The absence of a **unified assistant** contributes to a disjointed user experience, adversely affecting productivity and accessibility. In response to this issue, our all-encompassing browser extension serves as an indispensable co-pilot. It seamlessly incorporates features such as a chat-bot interface, multi-modal content retrieval, extensive language support, semantic caching for optimal performance, and robust databases for efficient storage of chat and web-page content.

An inherent issue faced by chat-bots, namely the propensity to generate inaccurate information due to hallucination. While this is acceptable for generative purposes, our solution needs to go a step further due to the prioritization of correctness in the answers given. This is tackled in our model through the implementation of techniques such as controlled history exposure. This is achieved through intelligent aggregation of information from diverse resources, mitigating the impact of hallucinations that can significantly compromise the reliability of responses.

2 MOTIVATION

Addressing the above problem is not just a convenience but a critical enhancement to the overall user experience in web interactions. The absence of a unified assistant across all website tabs poses a significant challenge, impacting user productivity and satisfaction.

Due to the lack of a similar tool issues faced are:

- **Fragmented User Experience**: The lack of a unified intelligent assistant leads to disjointed interactions and a suboptimal user experience. This hampers productivity, causing challenges in understanding, retrieving information, accessing personalized assistance.
- **Accessibility Concerns**: The absence of a cohesive solution hampers accessibility, making it challenging for users to find needed information, especially across multiple tabs, affecting their ability to fully leverage resources on various websites.
- **Complex Online Landscape**: In the era of vast digital information, navigating the increasingly complex online landscape has become daunting for users. The absence of a unified assistant exacerbates this challenge. Current hardcoded chatbots contribute to user frustration by providing only default, pre-fed answers.
- **Need for Personalized Assistance**: Users desire personalized support during online interactions. The lack of an intelligent assistant hinders their ability to receive tailored information and assistance, impacting the relevance and specificity of support.
- **Impact on Multilingual Interactions**: In the realm of online interactions involving diverse languages, a versatile language model is crucial. Absence of such capability introduces language barriers, constraining effective communication for users.

Our solution integrates a chatbot interface, an information retrieval system, a sophisticated language model along with RAG pipeline, and key technologies to enhance user experience significantly. It addresses the limitations of current systems, especially those dependent on chatbots, by extending the scope of answers beyond the confines of a specific website's domain knowledge. This capability addresses a common user frustration: the reliance on guesswork or the expensive fallback of routing queries to human operators. Our browser extension eliminates these issues by offering quick, context-sensitive responses, streamlining the browsing experience without the usual delays or inaccuracies. Moreover, it reduces the need to escalate queries to human operators, thereby increasing operational efficiency and providing a more cost-effective solution. This strategy significantly diminishes the system's reliance on human intervention, making it more autonomous and efficient.

3 TECHNOLOGIES & ALGORITHMS USED

- **Frontend for the ChatBot Interface**: ReactJS will be used to create an interactive user interface that contains a seamless and easy-to-use UI, giving the users a quality experience.
- **Backend**: The set of scraped/retrieved webpages will be stored across multiple vector databases. We have used Pinecone as the vector database.
- **Retrieval Algorithms for Ranking and Matching**: Cosine similarity has been used as the similiarity metric.
- **Creating a Chrome Extension**: This is the toolkit we will use for creating a Chrome extension. (Toolkit)
- **Framework for applications of LLM**: Langchain will be used to call LLM APIs and use them with our retrieval framework.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

4 NOVELTY

In comparison to the existing systems, our proposed idea offers a significantly enhanced user experience through a more robust context derived directly from the open tab. Our approach focuses on seamless integration and a stronger contextual understanding, ensuring better alignment with user preferences. A key feature is the personalization of user content, achieved through real-time data scraping on logged-in pages, ensuring that the information presented is both relevant and up-to-date. The system's capacity to tap into the entire webpage, as well as the specific section the user is viewing, offers a more nuanced understanding of the user's context. Additionally, our system supports multiple tab functionalities with a history that extends up to three previous tabs for enhanced pilot control.

Moreover, our idea stands out with the integration of modality in both input and output channels. The inclusion of multimodal retrieval further distinguishes our system, enabling us to answer user queries using a combination of various modes of information retrieval. Finally, the implementation of page-based question prompts ensures that users are presented with relevant inquiries tailored to each page, fostering a more interactive and user-centric browsing experience.

5 RELATED WORK

Current products in the market, such as `Microsoft Copilot` and `Perplexity AI`, serve as generalized co-pilots for users during browsing or searching.

In the initial phase of our pipeline, the accuracy of subsequent outputs heavily relies on effective data extraction. Previous works, exemplified by [2], have employed NLTK's punkt tokenizer to proficiently split sentences and tokenize them into Unicode word tokens. Additionally, the Body Text Extraction (BTE) algorithm, rooted in HTML tag distribution, enhances the extraction process. The paper introduces three ensembles as novel state-of-the-art extraction baselines. Moreover, the authors have demonstrated a thorough approach by amalgamating and refining various existing human-labeled web content extraction datasets. This meticulous compilation creates a comprehensive benchmark for evaluation, offering valuable insights instrumental in guiding our project forward.

Complex search tasks require more than support for rudimentary fact-finding or re-finding. The recent emergence of generative artificial intelligence (AI) and the arrival of assistive agents, or copilots, based on this technology has the potential to offer further assistance to searchers, especially those engaged in complex tasks. The authors **[11]** discuss the challenges and opportunities for researching, developing and deploying search copilots and ends by concluding that careful search interface design is required to help people quickly understand copilot capabilities, to unify search and copilots to simplify the search experience and preserve flow.

In the landscape of Multimodal Large Language Models (MM-LLM), the paper **[12]** introduces NExT-GPT, an innovative system designed for versatile interactions across diverse modalities, including text, images, videos, and audio. Unlike prior models, NExT-GPT achieves this by connecting a Large Language Model (LLM) with multimodal adaptors and distinct diffusion decoders. Notably, the model leverages well-trained encoders and decoders, requiring minimal parameter tuning, making it cost-effective and poised for potential expansion into more modalities. The introduction of modality-switching instruction tuning (MosIT) and a dedicated dataset for MosIT enhances NExT-GPT's cross-modal semantic understanding and content generation capabilities. This research makes a significant contribution by presenting NExT-GPT as a pioneering MM-LLM system, bridging modalities and paving the way for more human-like AI agents.

Conversational Information Seeking (CIS) has emerged as a novel paradigm for search engines, departing from the conventional query-Search Engine Results Page (SERP) paradigm. Unlike traditional methods, CIS empowers users to articulate their information needs through direct conversations with the search engine. The work by [8] presents a comprehensive approach to CIS, introducing a pipeline, a dataset, and a model to enhance information retrieval in conversational systems. The proposed pipeline consists of six integral sub-tasks: intent detection, key-phrase extraction, action prediction, query selection, passage selection, and response generation. This design aims to seamlessly integrate traditional search engine functionalities with the characteristics of conversational AI. The dataset, named WISE (Wizard of Search Engine), is constructed using a wizard-of-oz setup, simulating human-human CIS conversations. This approach captures a diverse range of information needs and conversational interactions. Furthermore, the paper proposes a modular end-to-end neural architecture tailored to each sub-task, allowing for both joint and separate training and evaluation.

A crucial facet of our product involves enhancing large language models by incorporating retrieved passages to enhance domain-specific question answering. As highlighted by [4], generative models for question answering can significantly benefit from passage retrieval. The approach involves extracting supporting text passages from an external knowledge source, such as Wikipedia. Subsequently, a generative encoder-decoder model generates the answer, taking into account both the question and the retrieved passages. We draw upon the insights provided by this study to reinforce our rationale for utilizing retrieval mechanisms to effectively augment Large Language Models (LLMs) when addressing user queries related to a webpage.

The concept of Retrieval-Augmented Generative (RAG) models, as introduced by Lewis et al. in their seminal work [6], has significantly influenced the landscape of natural language understanding and question answering systems. RAG models leverage the power of large-scale pre-trained language models (LLMs) and enhance their performance by integrating retrieval mechanisms. This unique approach addresses the limitations of purely generative models by incorporating relevant passages retrieved from an external knowledge source.Lewis et al.'s work emphasizes the crucial role of passage retrieval in augmenting the capabilities of generative models for question answering. This retrieval-augmentation paradigm enables the model to access a broader spectrum of information, providing a contextually rich foundation for generating responses. In alignment with the principles laid out in the RAG paper, our work focuses on applying the RAG model framework to the domain of domain-specific question answering. We recognize the potential of RAG models to enhance large language models

(LLMs) with domain-specific information, thereby improving the accuracy and relevance of responses to user queries about queried content in the webpages.

6 PIPELINE

The *DataPipeline* class in our code is designed to scrape websites, initialize a Pinecone index, initialize a vector store index, and run queries on the index. Here is a detailed description of all the individual steps:

- **Web scraping:** The *scrape_websites* method is used to scrape each website in the list. It sends a GET request to the website, parses the HTML response using BeautifulSoup 4, and extracts the text content. A **domain-specific custom CSS-based selector** is used to locate useful content on the scraped page.
  The text content is then cleaned to remove repeated blank lines. The images and their repsective ALT text are stored in separate folders ïmgänd ïmg_txt∴Additionally, we also save the URL and title (after stopword removal and lemmatization) of the page as the **supporting metadata**. The cleaned text and the original HTML are saved to separate files in the local storage. The meta data is stored in a json file for each web-page and image respectively. The method also maintains a mapping of websites to their corresponding data files.
- **Pinecone Initialization:** Pinecone is a vector database service that allows you to store, search, and retrieve high-dimensional vectors. In the *initialize_pinecone* method we create two separate pinecone for page text and image ALT text respectively. These indices will be used to store and retrieve vectors.
- **Index Initialization:** The *initialize_index* method is used to initialize a vector store index. This index is created by running an ingestion pipeline on the documents. The pipeline includes a sentence splitter and an embedding model. The sentence splitter breaks the documents and ALT texts into sentences, and the embedding model transforms these sentences into high-dimensional vectors. These vectors are then stored in separate vector store indices.
  Fundamentally there are two filtering methods with vectors: pre and post filtering. Pinecone utilizes a Single-Stage Filtering algorithm giving benefits of pre-filtering accuracy without being restricted to small datasets. Additionally, we have implemented a function which is responsible to add metadata into vector store aiding in First Stage Filtering.
- **Running Queries:** The *run_query* method is used to run a query on the index. It first checks if the documents are pickled and loads them if they are. If not, it reads the documents from a directory and pickles them for future use. It then initializes the index and a retriever. The retriever is used to retrieve the top 3 similar nodes to the query string from the page text index and image ALT text index respectively. This is done by transforming the query string into a vector using the same embedding model, and then retrieving the vectors in the indices that are most similar to the query vector, for the image ALT text vectors we further check if they are relevant or not by using a threshold of 0.6 and only relevant ones are retrieved.
- **Output Generation:** The retrieved page text vectors are then forwarded to a Large Language Model (LLM) for generating replies and the generated text along with retrieved images are returned to the user.

7 EXPERIMENTS AND RESULT

7.1 Methodology

7.1.1 ***Query Generation.*** For testing our pipeline, we generated some queries from our corpora of documents. This was done to check the ability of our employed retrieval pipeline to retrieve accurate and relevant documents upon being prompted with a query. The following steps are followed for this process:

- Websites from the IIITD domain such as the ones about *B.Tech Project, Guest House, Hostels, Mess, Student Affairs* etc, Taj and Legal Law were scraped and stored as *.txt* files. These served as the document corpus for our experimentation. In total the corpus consisted of **10 such documents** for each website on which our pipeline will perform retrieval.
- Each *.txt* file was copied and was then input to OpenAI's ChatGPT chatbot [1] with the **zero-shot prompt**:
  You are a question setter. Go through the text given to you and give 5 question-answer pairs from it.
- This prompt generates 5 candidate question-answer pairs for each document. We select the top 2 queries from this and feed it into a pool of queries to test our retrieval system on. The total number of selected queries for retrieval is 50.

7.1.2 ***Benchmarking of LLMs:*** For benchmarking different large language models (LLMs), we leveraged the **WikiQA dataset** [13], a collection of question-answer pairs based on Wikipedia content. This dataset aids in assessing the models' capabilities in natural language understanding and generation. The evaluation process integrates the **retrieval-augmentation-generation (RAG)** technique, where models first retrieve relevant information before generating answers.

- **Dataset and Technique:** We use the **WikiQA dataset** [13] to evaluate the models' performance. This involves feeding the LLMs with a query, general instructions, and the retrieved context, employing the RAG technique to enhance relevance and accuracy.
- **Models Benchmarked:** The LLMs benchmarked include *Mixtral-7x8b* [5], *LLAMA-3-8b* [10], *Gemma-7b* [9], and *GPT-3.5-turbo* [3]. These models were selected based on their diverse architectures and capacities.
- **Procedure:** For each model, we retrieve the most relevant Wikipedia pages using the Hugging Face Wikipedia corpus. The retrieved pages then serve as context for the LLMs, which are tasked with generating an answer to the query.
- **Evaluation Metrics:** The effectiveness of each model is measured using BLEU-score, METEOR-score, and BERTscore metrics (precision, recall, and F1-score). These metrics help quantify the accuracy and relevance of the generated answers compared to the actual answers in the WikiQA dataset.

[1] https://openai.com/blog/chatgpt

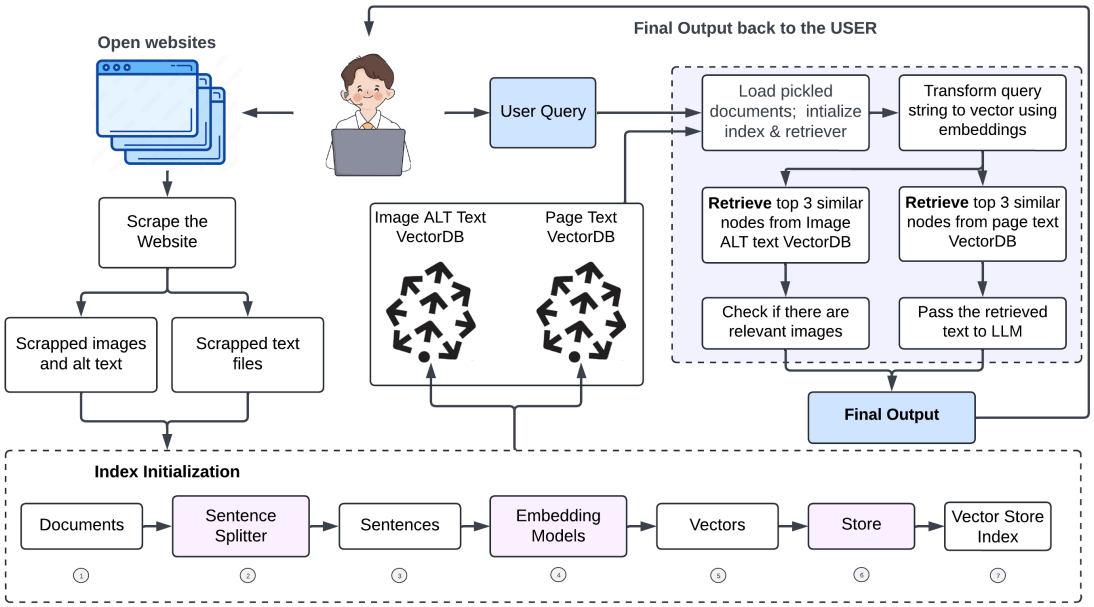

Figure 1: Data Pipeline

7.2 Evaluation Metrics

7.2.1 ***Query Retrieval:*** Since this report details our baseline experiments for the retrieval setup, we employ the following binary relevancy metrics for evaluating our system. Based on the query generation, there is one relevant document for each input query.

(1) **Precision@K**: Is the proportion of retrieved documents in the top K results that are relevant. Since there's only one relevant document per query, Precision@K will be 1 if the relevant document is within the top K results, and 0 otherwise.
(2) **Mean Average Precision (MAP):** Mean Average Precision (MAP) is a metric that calculates the average precision across all relevant documents for each query in information retrieval tasks, providing a comprehensive measure of system accuracy.

We have a total of 20 generated queries with their most relevant document (i.e. the ground truth) as described in the previous section. We prompt these 20 queries one by one to the pipeline and retrieve the **3 most relevant documents**. We then evaluate against the ground truth for testing the robustness of our system.

7.2.2 ***LLM Benchmarking:*** To benchmark large language models (LLMs) effectively on text generation tasks, a mix of qualitative and quantitative metrics is essential. These metrics evaluate the text's coherence, relevance, and fluency. Common automated metrics include BLEU, which measures n-gram overlap with reference texts; METEOR, which accounts for synonymy and stemming; and BERTScore, which assesses semantic similarity using contextual embeddings from models like BERT. Combining these metrics with human evaluations ensures a comprehensive assessment across diverse linguistic scenarios. In our analysis, we utilized BLEU, METEOR, and BERTScore for a detailed evaluation of LLM performance.

(1) **BLEU** [7]: BLEU measures the similarity between machine-generated text and one or more reference texts by counting the matching n-grams, providing a score that reflects how natural the translation or generated content appears. The reported BLEU score is an average of the BLEU-1, BLEU-2, BLEU-3 and BLEU-4 scores.
(2) **METEOR** [1]: METEOR improves upon BLEU by including synonyms and stemming in its evaluation, allowing for a more nuanced understanding of language use. It aligns better with human judgment by accounting for both precision and recall.
(3) **BERTScore** [14]: BERTScore leverages the deep contextual embeddings from pre-trained models like BERT to calculate the semantic similarity between the generated text and the reference text. This metric focuses on the contextual alignment of words, offering a sophisticated assessment of content quality.

7.3 Results

7.3.1 ***Query Retrieval:*** This section describes the results obtained by our system on the evaluation metrics and experiments described in the previous section. The results using cosine similarity for retrieval are summarised in the table 1 for IIIT Delhi website and in table 2 for Taj Mahal website.

We report the metrics Precision@K, Recall@K and F1@K averaged over the 20 queries used by us to test the system. Thus we refer to it as an average of those values. The values @1 are high

indicating our system's robust ability to retrieve the relevant document as the first one itself. Precision values @2 and @3 are lower because of there being only one relevant document in this current experiment. This is also reflected by our Recall@K values being 1 for $K = 2, K = 3$ which means all relevant documents are retrieved. The further robustness of our system is validated by the high Mean Average Precision (MAP) value reported in the end.

K	Avg Precision	
1	0.85	
2	0.5	
3	0.33	
	MAP	0.925

Table 1: Pipeline Results using Cosine Similarity (IIIT D)

K	Avg Precision	
1	0.88	
2	0.47	
3	0.31	
	MAP	0.916

Table 2: Pipeline Results using Cosine Similarity (Taj)

Combined Retrieval from 3 websites: Furthermore, we combined the data of 3 websites i.e IIIT Delhi, Taj mahal, and a law website (latestlaws.com) and tested the system by performing 47 queries on the system. The metrics obtained are summarised in table 4 as follows:

K	Avg Precision	
1	0.89	
2	0.56	
3	0.41	
	MAP	0.932

Table 3: Combined Pipeline Results for all 3 websites

K	Avg Precision	
1	0.93	
2	0.68	
3	0.52	
	MAP	0.96

Table 4: Updated Combined Pipeline Results for all 3 websites

7.3.2 ***LLM Benchmarking Results:*** Based on our LLM benchmarking evaluation metrics, we observe that GPT-3.5-Turbo is performing the best, closely followed by Mixtral-8x7b, LLAMA 3-8b and Gemma-7b respectively. Hence, in our final pipeline, we are using the GPT-3.5-Turbo as our LLM.

Model	BLEU	METEOR	B-P	B-R	B-F1
Mixtral-8x7b	0.013	0.183	0.55	0.675	0.605
LLAMA-3-8b	0.012	0.171	0.595	0.672	0.628
Gemma-7b	0.009	0.113	0.627	0.660	0.641
GPT-3.5-turbo	**0.021**	**0.232**	**0.654**	**0.713**	**0.687**

Table 5: Performance comparison of different LLMs on various metrics. B-P: BERT-Precision, B-R: BERT-Recall, B-F1: BERT-F1.

8 POTENTIAL CONTRIBUTIONS

Our contribution in this work will be developing a pipeline capable of chatting with the user and retrieving answers based on queries on pages they are surfing. This will be provided by an extension that exists as an overlay of the browser. We also aim to involve multimodal retrieval by allowing Q&A over images and audio data. This will be supported by multiple multimodal input and output channels according to the query. Our contributions will be vital in making open-source products which can be helpful for understanding focused context and improve grounding.

REFERENCES

[1] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[2] Janek Bevendorff, Sanket Gupta, Johannes Kiesel, and Benno Stein. 2023. An Empirical Comparison of Web Content Extraction Algorithms. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Taipei</city>, <country>Taiwan</country>, </conf-loc>) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 2594–2603. https://doi.org/10.1145/3539618.3591920

[3] 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. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[4] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, Online, 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74

[5] 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. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG]

[6] Patrick S. H. 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. *CoRR* abs/2005.11401 (2020). arXiv:2005.11401 https://arxiv.org/abs/2005.11401

[7] 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*. 311–318.

[8] Pengjie Ren, Zhongkun Liu, Xiaomeng Song, Hongtao Tian, Zhumin Chen, Zhaochun Ren, and Maarten de Rijke. 2021. Wizard of Search Engine: Access to Information Through Conversations with Search Engines. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Virtual Event</city>, <country>Canada</country>, </conf-loc>) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 533–543. https://doi.org/10.1145/3404835.3462897

[9] 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 Mikuła, 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. arXiv:2403.08295 [cs.CL]

[10] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[11] Ryen W. White. 2024. Tasks, Copilots, and the Future of Search: A Keynote at SIGIR 2023. *SIGIR Forum* 57, 2, Article 4 (jan 2024), 8 pages. https://doi.org/10.1145/3642979.3642985

[12] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2023. NExT-GPT: Any-to-Any Multimodal LLM. arXiv:2309.05519 [cs.AI]

[13] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lluís Màrquez, Chris Callison-Burch, and Jian Su (Eds.). Association for Computational Linguistics, Lisbon, Portugal, 2013–2018. https://doi.org/10.18653/v1/D15-1237

[14] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).