Group 41 - Baseline Results 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

1 INTRODUCTION

In the current digital landscape, users face challenges in efficiently accessing relevant information and personalized assistance across different websites. The lack of a **unified assistant** leads to a fragmented user experience, impacting productivity and accessibility. To address this, our comprehensive browser extension acts as a co-pilot, seamlessly integrating features like a chatbot interface, multimodal content retrieval, versatile language support, semantic caching for performance optimization, and robust databases for chat and web page content storage.

2 PROBLEM IMPORTANCE

Addressing this 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, 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 TO BE 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. Possible Vector DBs that we are considering using include Pinecone and Milvus. The URL to each individual vector DB will be stored as a foreign key in a parent Mongo database.
- **Retrieval Algorithms for Ranking and Matching**: Similarity search algorithms include an Approximate Nearest Neighbor search. Similarity matching will be done using Cosine Similarity and Jaccard similarity scores, taking an average of the two.
- **Ranking**: We will aim to investigate probabilistic and vector space models for ranking and use algorithms like PageRank. Ranking can also be done on the metadata which can be generated synthetically.
- **Audio Processing (Audio inputs and for audio retrieval)** Whisper (OpenAI) can be used for ASR. Bhashini API is for ASR and TTS for Indian Languages.
- **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.

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 [1], 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 **[5]** 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 **[6]** 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 [4] 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 [2], 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 [3], 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.
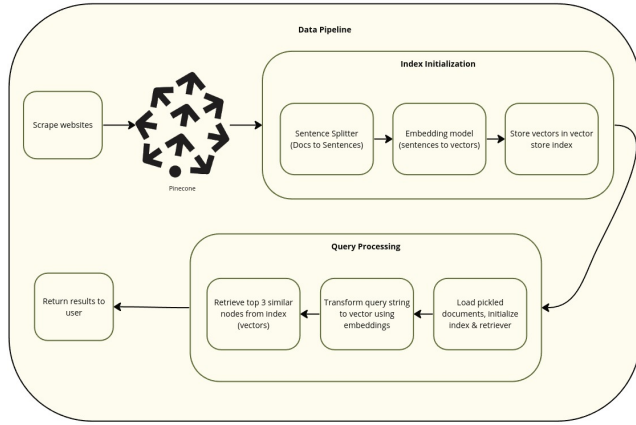
6 PIPELINE

Figure 1: Data 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, and extracts the text content. The text content is then cleaned to remove repeated blank lines. The cleaned text and the original HTML are saved to separate files in the local storage. 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, a Pinecone index named "detrieval" is created if it doesn't exist. This index 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 into sentences, and the embedding model transforms these sentences into high-dimensional vectors. These vectors are then stored in the vector store index.
- **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 index. This is done by transforming the query string into a vector using the same embedding model, and then retrieving

the vectors in the index that are most similar to the query vector.

7 EXPERIMENTS AND RESULT

7.1 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 and Taj 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.

7.2 Evaluation Metrics

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) **Recall@K**: Measures the proportion of relevant documents that are retrieved in the top K results out of all relevant documents. Since there is only one relevant document for each query, Recall@K is 1 if the relevant document is within the top K results and 0 if it is not.
(3) **F1@K**: The F1 score at K is the harmonic mean of Precision@K and Recall@K. Given that there's only one relevant document per query, F1@K essentially measures the balance between Precision@K and Recall@K. Because Precision@K and Recall@K will always be the same, F1@K will also mirror those values.
(4) 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

---

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

K	Avg Precision@K	Avg Recall@K	Avg F1@K
1	0.85	0.85	0.85
2	0.5	1	0.66
3	0.33	1	0.496
Avg MAP	0.925		

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

K	Avg Precision@K	Avg Recall@K	Avg F1@K
1	0.88	0.88	0.88
2	0.47	0.94	0.63
3	0.31	0.94	0.47
Avg MAP	0.916		

Table 2: Pipeline Results using Cosine Similarity (Taj)

the **3 most relevant documents**. We then evaluate against the ground truth for testing the robustness of our system.

7.3 Results

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.

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.

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] 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

[2] 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

[3] 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

[4] 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

[5] 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

[6] 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]