Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Current Content Discovery for Business Module Teaching

by

Guanyuming He

Email: guanyuming.he24@imperial.ac.uk
GitHub username: esemsc-gh124
Repository: https://github.com/ese-ada-lovelace-2024/irp-gh124

Supervisors:

Sean O'Grady
Rhodri Nelson

27th August 2025

Abstract

Business school teaching using the well-known case method requires a substantial amount of information about latest business events to keep the course relevant. Digital technologies like web search engines and LLMS can offer substantial assistance to this process. However, several challenges remain: 1) search engines struggle to process unstructured search queries. 2) while LLMs excel in understanding unstructured input, they often hallucinate and cannot reliably retrieve very recent information. 3) these tools still require manual invocation. In this project, I aim to address these challenges by developing a software system which combines search engines and LLMs in clever ways and also support automatic execution of the whole process and the delivery of the results to users, requiring minimal user configuration.

Keywords: information retrieval, LLM, search engines, case method, Business school

1 Introduction

1.1 Project background

Since the emergence of the first Business schools in the late 19th century [68, 60], several distinct pedagogical teaching strategies have been applied. First institutionalized at Harvard Business School [25, 9] in the early 20th century, a method about teaching students with real world business cases (will be called *case method* in the rest of the thesis) has been found more effective and engaging [73, 7, 37] than many traditional, for example, big lecture based, teaching methods. Thus, it has found wide adoption across the world [71, 14].

Despite its aforementioned adoption and performance in business school teaching, the case method faces a significant constraint: the availability and collection of timely and relevant case material [64, 50]. In particular, Christensen identifies in his classical article that instructors have to conduct "extensive preparation" [10] for it. Another reason is the ever-evolving business world and the necessity of the latest information: Clark argues that learned skill will lose value quickly in five years, and then it is critical for students to be up to date to remain relevant in the business world [11]; McFarlane emphasises the importance of updated cases, as otherwise students could be disengaged or discouraged [39].

1.2 Past advancements in information retrieval

The thesis summarizes the challenges in information retrieval into two general problems: (1) collect/gather information (2) identify/select desired information from gathered sources.

1.2.1 Problem (1)

During the past two centuries, a number of key developments have profoundly expanded one's capacity to gather information about the world. In the early 19th century, the transmission of information was still traditional — carried by person on paper or simply remembered. The invention of telegraph in the 1840s by Morse, Cornell, and Henry [62, 35], notably with Morse's first telegraph message, "*What hath God wrought?*" in 1844 [42], marked a paradigm shift. This technological innovation was considerably improved by Bell's telephone in 1876 [75, 17], enabling communication directly by human voice, instead of encoded Morse code.

Wired communication is critically constrained by geographical features on where the wires were laid. Around the late 19th century, Marconi's experiments with wireless telegraphy [15] and the first successful transatlantic signal in 1901 [6] introduced electromagnetic wave-based wireless communication, eventually accumulating into the world's first voice broadcast by radio in 1906

[67]. These milestones collectively redefined the temporal and spatial boundaries of information gathering.

In parallel to improving the weaknesses of wireless communication [24, 77, 1], reseachers then worked on a theory of information. Hartley observed a logritham pattern of information capacity [26] and then Shannon expanded on it to first define *bits* and *entropy*, giving a formal mathematical theory of information [65] in 1948. Meanwhile, engineers were experimenting with alternative modulation techniques, notably frequency modulation (FM), and the concepts were formalized in the 1930s [27].

1.2.2 Problem (2)

Although solutions to problem (1) had enormously improved during these times, substantial progress in problem (2) would have to wait until digital computers were invented in the 1940s [76], which were made to repeat tedious operations fast. Actually, the term *information retrieval (IR)* was not invented until Mooers coined it in 1950 [41]. The nature of information reveals two subproblems of problem (2): (2.1) how to process (extract useful information from) potentially unstructured data. (2.2) how to related human queries to the content a user actually wants.

According to Griffiths and King, early IR systems (in the USA) were mainly transferring human computing activity (bookkeeping in a library or database) into digital forms, by various sorting, indexing, and searching algorithms [21]. The form slowly went from “offline, batch” computing to “online, real-time” computing, and finally became “distributed, networked, and mass computing”, thanks to the Internet and various of its applications [21, 33]. Then commercial search engines emerged in the 1990s, opearting on the scale of the whole Internet [63, 45].

The aforementioned sorting, searching, and indexing algorithms were partial solutions to problem (2.1) and (2.2), with fundamental designs like inverted index [61, chap. 2], [81, sect. 2], boolean search [61, chap. 1], and various clever ideas such as spelling correction (stemming) [61, chap. 3.3], positional indexing [81, sect. 3]. Because the database can be extremely large, specific index construction [61, chap. 4], [81, sect. 5] compression [61, chap. 5] algorithms have appeared. Also, people saw ample opportunity to explore distributed computing, with the most notable model probably being MapReduce [12].

Nevertheless, these inventions were still mostly doing textual processing, and machines struggled to “understand” information. Thus, advanced and complex search queries are often needed to achieve desired performance [2, 53]. The research on making computers understand natural language (NLP), images and visuals (computer vision), etc. were consequently and gradually being integrated into search engines, since 2000, notably with Google’s Panda, Penguin, and Hummingbird algorithms that aim to understand web content better to rank them in searches [49].

One major breakthrough in NLP is Vaswani et al.’s transformer [72]. Building upon it, OpenAI created generative pre-trained transformers (GPTs) trained on massive amount of data [52], which, after a few iterations, evolved into a chatbot facing the general public, ChatGPT, and it was perhaps then when LLMs started to recieve massive amount of public attention, where LLM stands for large language model, language models that are *large* in trained data and parameters.

Although public attention is not the same as the importance of an idea, LLMs have undoubtedly entered and transformed many areas, including daily life, business and the industry, and research [13]. One strong appeal of LLMs is that they could process unstructured natural language input and generate natural language output in return with remarkable resemblence to

what a human would say [79, 32]. However, because of the inherent limit of neural networks, some argue that they could not formally reason about what they output [59, 31, 58]. Indeed, hallucination [28, 55] and other forms of distortion of facts, is a big problem of LLMs.

Despite the drawbacks of LLMs, they are currently the most successful tool in NLP, and naturally various attempts have been made to integrate LLMs with search engines [78, 66, 74]. Specifically, Xiong et al. proposes to categorize them into "LLM4Search" and "Search4LLM", where "A4B" means using A to improve B [78]. Here is where my thesis will build upon. Now, with the help with LLMs that can easily process frivolous natural language input, I plan to integrate them together to boost an individual's information retrieval further, especially in the area of business school teaching content discovery.

1.3 Contemporary related work

1.3.1 Retrieval augmented generation

During training, LLMs turn training data into their parameters (also parametric-memory). Its immutability after training is a weakness for information/knowledge based tasks, as information cannot be easily added without retraining. RAG is a general approach to fine-tune a trained model with non-parametric memory [34].

RAG was designed to enrich a trained LLM with a pre-trained retriever. Addressing hallucination is its major concern. Because the retriever also needs to be pre-trained, it cannot inherently solve the outdated training data problem. In fact, their work used a retriever trained on a 2018 Wikipedia dump [34, Section 3]. Additionally, the retriever is also a neural network. These are the major differences from my work.

1.3.2 Other hybrid methods

Beside RAG, numerous studies were conducted to augment pre-trained LLMs with external knowledge [23, 29, 8, 80, 43]. Some worked on the retriever, some on how information is integrated into the LLM, and some on both. Unlike my work, they focus on question-answering tasks, and none of them had getting the latest information as a requirement.

Among them, Nakano et al.'s WebGPT is perhaps the closest to my work: their retriever is a text-based web-browser [43]. Nevertheless, WebGPT was designed to answer long-form questions, trained primarily to answer questions from a specific dataset. Moreover, it used human feedback to improve quality. These are the differences from my work.

1.3.3 Search engine centric

While the aforementioned studies work on improving LLMs, various works are also committed to improve search engines.

Some of them rewrite and improve search queries with LLMs [40, 3]; some turn to assist ranking of search results [69, 69]. My work has both elements: in my project LLM will both be used on search queries and on search result synthesis.

Furthermore, LLMs can potentially help document indexing with their excellent NLP abilities in term extraction [38, 18] and document filtering [57], although the first two cited work do not primarily consider search engine applications.

1.4 Goals

1. By combining LLMs and search engines, compensate each other's weakeness in information retrieval.
   (a) The system shall improve search engines on prompt engineering to the extent that a user would only need to give a rough and vague natural language description of the target information to the system.
   (b) The system shall improve LLMs on result reliability and accuracy. If pure LLMs lack the two properties, then combination with deterministic search engines, using deterministic and accurate algorithms, will compensate for that.
2. Enhance the automation of the current general public information retrieval tools, e.g., Google search, ChatGPT, to the extent that a user would only need to occasionally configure the system, instead of giving search prompts each time.
3. Specially tailor the system to business teaching related information, aiming to gather information from authentic and reliable sources.
4. The system shall support up-to-date information retrieval. Particularly, it should prevent outdated LLM training data from polluting the output.
5. (Optional) The system could support user feedback and learning from it.

1.4.1 Not in the goals

1. Providing a method to rank business information. That is rather subjective.
2. Design, build, and train a LLM. That is too much work.

2 Methods

2.1 Design philosophy

The core of this project is a big software system designed to achieve the goals. Considering the scale and complexity of the software system, its development will greatly benefit from a clear design philosophy.

The design philosophy is a set of the following philosophical approaches to develop the system. Approaches 1–3 are from MIT Software Construction [19]. Approach 4 is part of the Unix philosophy [30].

1. *Correct today and in the unknown future*: By developing the system using design concepts like abstract datatypes [36], immutability, and software testing, its current correctness is defended; by defensive programming (type safety, robust checks, etc.), its correctness is defended from my future mistakes and stupidity.
2. *Easy to understand*: Via thorough documentation, static type hints, and again immutability, the code can communicate well with the future me who may need to fix bugs or extend the system.
3. *Ready for change*: The system should be designed to make extensions easy; it would not be ideal to discard and rewrite a lot of code for extensions.
4. *Do one thing and do it well*: The system is composed of a collection of programs, each of which is designed to do one particular job, and all of which are designed to work together.

2.2 Architecture

As clarified in the previous section, the system is a collection of programs working together. What threads them together is a pipeline, which takes the user's configuration as input and produces the current information about the business topics mentioned in the config as output. The workflow of the pipeline is:

1. LLMs generate a list of search engine prompts based on the business topics in the configuration.
2. The prompts are fed to search engines.
3. The search results are gathered and filtered. In particular, filter by date.
4. The processed results are synthesized by LLMs.
5. The results are sent to the users via the configured ways.
6. Optionally, the user gives feedback to the results.

The pipeline needs to retrieve current information from somewhere. Thus, another big module of the system is a minimal web search engine that can index business news websites for latest news. The purpose of this self-made search engine is to provide explicit control of the indexed contents and to avoid commerical limitations on the usage. See section 2.2.1 for the complete discussion. Still, third-party search engines are combined with the custom engine.

Surrounding the pipeline and the search engine are the supporting programs which handle automation (schedulling), output delivery (by email), and user-friendly configuration management.

Table 1 lists all programs in the system and their functionality. The names are kept short to not overflow the table. The C++ programs can be found in `src/CMakeLists.txt`; the Python scripts can be found in `src/`.

`indexer`, `searcher`, `updater`	Build a search engine database from scratch, search a database, and update a database by adding new entries from RSS and removing old ones, respectively.
`rm_doc`, `upd_doc`	Tools intended for the developer (me) to fine-tune the database by removing and updating specific documents in it, respectively.
`llm_pipeline.py`	The main pipeline program.
`config_gui.py`	A user friendly GUI program to configure the system.
`search_combined.py`	Obtains the search result from my custom search engine as well as Google, filters them, and combines them.
`setup_schedules.py`	Schedule automatic executions for various programs in the system to make the system automatic.
`tests/*`	Unit tests
`misc/*`, `doc_dist`	Miscellaneous programs for analyzing the system in the results section.

Table 1: All programs in the software system

Figure 1 demonstrates my architecture design and the workflow of the system, where blue boxes form the pipeline, the yellow boxes are the other supporting programs, and the red box with the dashed lines are optional.

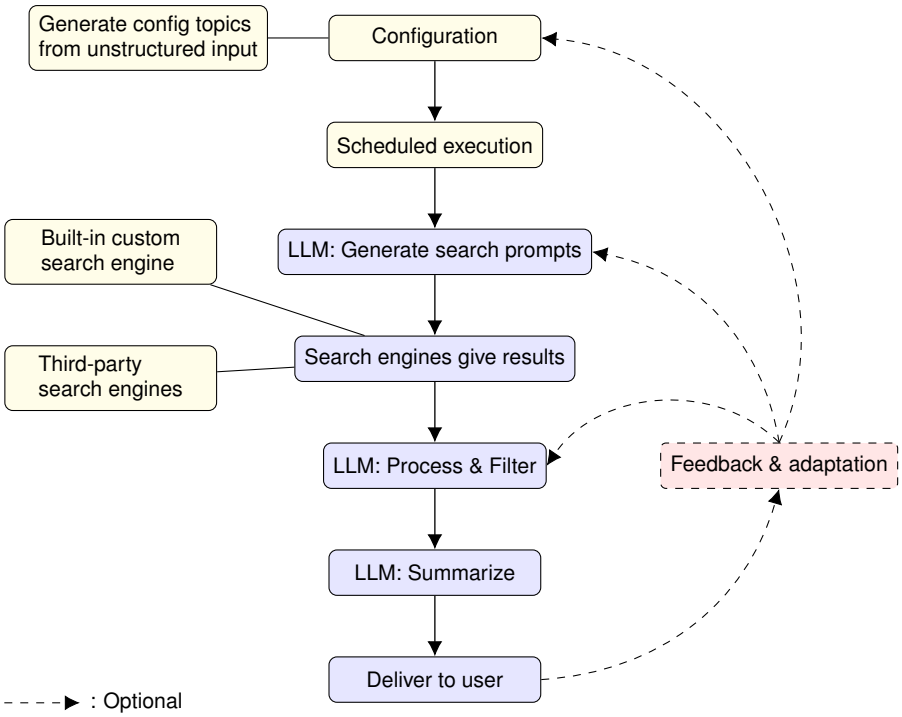

Figure 1: Architecture and operational flow of the software system

2.2.1 Search engines

The easiest way to integrate a web search engine is to use a commerical search API, building upon the large database the commerical company provides. However, this suffer from a few serious problems.

1. The free-tier APIs often have various ungenerous limits, e.g., number of queries, number of results returned, and filtering of search results. For example, see Google's [20] and
2. The algorithm of the search engines are proprietory; I cannot control what they give. This is a big problem, because my goal is not about retrieving general information but specific information of business teaching value.

Therefore, I plan to use commerical search APIs alongside my custom minimal search engine. The custom search engine will address the two problems with unlimited usage and specific information retrieval. To achieve the latter, I configure the search engine to ONLY index business news websites. Clearly, I will not have the resource to index most of the Internet and store them, and this coverage weakness will be compensated by the use of commercial search APIs.

2.3 Implementation detail

Following the design philosophy and architecture, here are the details of the implementation.

2.3.1 Configuration format

Most of the configuration is structured, e.g.,

1. Scheduled time of the executions.
2. A list of email addresses to deliver the results to.

The most important entry is the business topics used for information retrieval. Here, LLMs will be used to extract them from unstructured data, including both user natural language input and file input (e.g. business school lecture notes).

For all the configuration entries, see `config.py`.

2.3.2 LLM invocation

Considering the heavy use of LLMs and the strict limitation of commercial LLM APIs [48, 4], the project will use local LLMs powered by Ollama [46].

Ollama provides two main modes to communicate with LLM, `generate` via and `chat`; both access the Ollama local server at `localhost:11434` via HTTP POST. Because the system is not interactive, only the generate method is used.

Apparently, the system prompts are essential to our tasks. There is generally no fixed algorithm to arrive at the best prompts given to LLMs, as the results not only depend on a lot of aspects including how the input is given, but also probably are not deterministic, given that probalisitic processes are widely used in deep neural networks and complex machine learning models. Thus, the only way to obtain desirable prompts is to manually craft the prompts and test them.

Initially, each stage invoking LLMs was one-shot (using only one invocation of the LLM). Later tests revealed that this approach was extremely hard to tune for all situations. Therefore, I turned to have a multi-stage invocation for each step, enabling the LLM to focus on one specific task at one time.

The manual tests are done in two ways. The straight-forward way is to directly change the prompts in my system, which is easy, thanks to the fact that I put them in configuration. Still, this requires running a large part of the system and is inefficient. Another approach involves OpenWebUI [5], a GUI tool that allows me to interact directly with the LLM models via a web UI. Note that the models are in `chat` mode instead of `generate` mode. Nevertheless, practice proved that prompts good in chat mode is still relevant in the generate mode.

The LLM models are selected prioritizing usability on personal computers. Thus, the maximum number of parameters should be about 10 billion. Among all the models Ollama offers [47], I chose `llama3.1:8b` for its competitive ability on text generation and summarization, proven by several academic studies and benchmarks [56, 54, 70]. Unfortunately, llama3.1:8b is text-only. When the user uses files to generate business topics, as mentioned earlier, another model fine-tuned with llama3 instructs, `llava-llama3`, is used.

2.3.3 Search engines

My custom search engine is implemented in C++ with `libcurl` for url handling, `lexbor` for HTML parsing, and `xapian` for database building and searching. Also, `pugixml`, `boost.url`, `boost.test`, and `amosnier/sha2` are used as support libraries. Finally, Python is linked to call Python functions inside C++.

The custom search engine's algorithm is similar to the mainstream ones: Starting from initial urls, scrap each. And for every url found in each, scrap them too. More precisely, the indexing of a webpage and the expansion of its urls included are controlled by four predicate filters. Two are on URLs and two are on the webpages. A webpage is indexed if both URL & webpage index predicates return true; each of its url is considered further if both URL & webpage recurse predicates return true. I separate the URL & webpage predicates because I can then delay scraping of a webpage only after its URL passes the conditions.

When search prompts generated by LLMs are given, they will be fed to both my custom search engine and Google's programmable search engine, with their results combined and filtered.

2.3.4 Task Schedulling

The scheduled tasks are installed on the target operating system via cron, a task scheduler on Unix-like systems.

Before the tasks are actually installed, the installer script will check if two jobs execute at the same time and abort the installation if they do, to avoid parallel access to the custom search engine database.

2.3.5 Report delivery

The results, finally summarized as a report, is sent to the users automatically via email. Because the code is visible in the repository along with the login credentials, I registered a burner Google account for email delivery. The configuration records a list of destination email addresses, to each of which the final results of running the pipeline will be sent from the burner account.

The protocol is SMTP + TLS to ensure that the email content is confidential.

3 Results

This section demonstrates the completed system. Next, using the same inputs, information retrieved by mainstream LLMs & search engines are also displayed for comparsion in the discussion section.

As a Business School teacher, my supervisor provides me with a list (LIST 1) of topics from his modules which will be used as inputs for demonstrating results.

1. Vertical integration
2. Diversification
3. Competitive advantage
4. Foreign direct investment
5. Mergers and acquisitions
6. Global supply network
7. Global value chain

3.1 Completed system

3.1.1 Configuration GUI

The GUI used for end-users to configure the pipeline is implemented in Tkinter [51], a Python GUI interface.

The source is in `config_gui.py`. By the principles in section 2.1, the configuration is divided into different separate sections, each of which is managed by a GUI class inheriting from base lass `ConfigSectionFrame`.

Figure 2 shows a screenshot of the GUI.

Figure 2: Configuration GUI

3.1.2 Custom search engine

Recall that the main goal of developing this custom search engine is to filter out non-concrete information sources, which mainstream search engines all include, as they index the whole Internet.

Figure 3 displays the host distribution of all the indexed data on my personal computer, as of 7 August. This figure is produced by `doc_dist.cpp` and `doc_dist_pie.py`. Unfortunately, because of the volatility of the Internet content, the exact data is probably not reproducible. Clearly, all domains from the database are reputable Business news publishers.

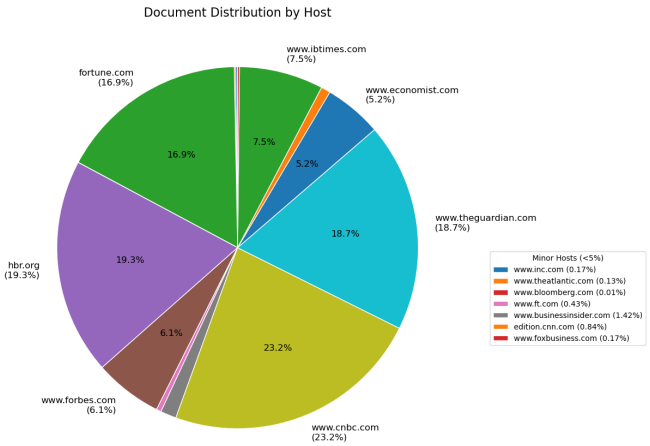

Figure 3: Indexed document host distribution

Since it is entirely local, its response times are anticipated to be much shorter than web-based

search engines. Nielsen claims that having a ≤ 0.1 second response time makes users experience instantaneous interaction [44, Chapter 5], a standard I expect it to achieve.

To verify this anticipation, a performance profiling is conducted (`search_eng_profiling.py`), using around 100 search queries (`100_queries.txt`) generated by LLM for representativeness. Figure 4 contains the result; all except one of average response times are below 0.1 second.

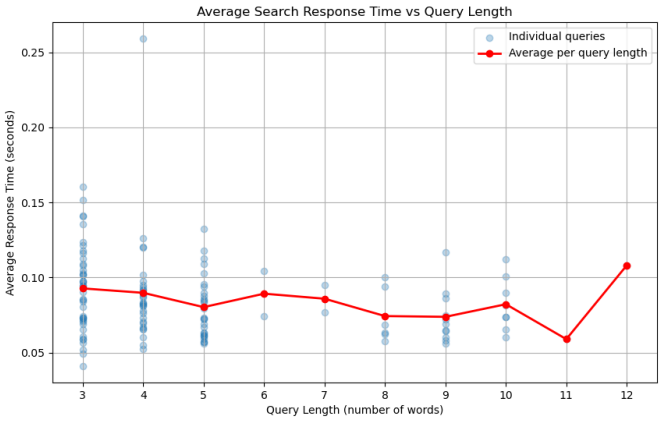

Figure 4: Average response time vs. query length

3.1.3 LLM invocation

LLMs are invoked with two main purposes in the project, search query generation from business topics, and summarization of search results.

Table 2 lists the results of search query generation, using inputs from LIST 1. They are collected from the output of `llm_pipeline.py`. The results demonstrate very high relevance to the topic, although one of them cites an old year (... trends 2024 examples). A search query is judged as relevant, if it contains the exact phrase or synonyms, or express the same meaning as the topic.

Topic	Search Queries	Relevance
Competitive advantage	innovation strategies for business competitiveness intrapreneurship best practices for sustainable growth tech firms leveraging digital innovation to gain market edge sustainable competitive advantage through emerging technologies examples Amazon supply chain innovation Walmart logistics strategies	1, 2, 3, 4 (4/6)
Vertical integration	vertical integration advantages and disadvantages pros and cons of integrating supply chain vertically Vertical integration success stories Failed vertical integration examples in manufacturing industry	1, 2, 3, 4 (4/4)
Diversification strategies	Diversification strategies mergers acquisitions recent news M&A diversification trends 2024 examples geographical expansion impact on company performance case studies expansion into new markets and its effects on corporate success stories Product line diversification vs market expansion best practices	1, 2, 3, 4, 5 (5/5)
Foreign direct investment	Effects of foreign direct investment on local job market and labor laws in India and China Foreign investment impact on domestic employment and labor regulations worldwide Foreign direct investment emerging market economic growth Economic development in emerging markets foreign investment impact	1, 2, 3, 4 (4/4)

Table 2: Relevance of generated queries

The results of search summarization are even harder to qualify objectively. An objective merit is that the LLMs can sometimes uncover surprising angles, while an objective shortcoming is that the summarization quality sometimes cannot meet expectation. One novalty in the summarization, only discovered in late stage after several testing, is to make LLM output individually

for each search result a relevance number, which is used for filtering and reranking, instead of giving all results to the LLM, where the LLM often distort the search results.

3.1.4 Walkthrough

The full system starts with the user's configuring the business topics, which she can generate from unstructured text and images, the automatic scheduelling, and the accounts, like sender email address, destination email addresses, and Google search API key. All of them are configured on the GUI.

After the configuration, a setup script needs to be run to install the schedualling tasks on the operating system. The user should also run the indexer to initialize the custom search engine database. Then, at scheduled times, the LLM pipeline and database updater will run.

The LLM pipeline starts by reading the business topics, then

1. Invoke LLM to divide each topic into a diverse range of subtopics.
2. Invoke LLM to generate search queries for each subtopic.
3. Invoke both the custom search engine and Google's to produce search results and combine them.
4. Invoke LLM to filter and rerank the search results.
5. Invoke LLM to summarize all of them to a final report.
6. Send the final report to the configured destination email addresses.

The database updater pulls the latest business news from the RSS feeds of reputable Business sites, and then also removes the oldest documents from the database, if the total number of documents exceeds a limit.

The user is free to reconfigure the system, and the changes will be reflected the next time tasks are run.

The lengthy result includes both a final report and a directory of intermediate results (e.g. search queries generated and raw search results), and thus cannot be included in this word-count-limited thesis. Moreover, the use of LLM makes them non-deterministic. The reader can instead run the pipeline herself to examine output.

3.2 Results from mainstream tools

I feed the same input to mainstream tools. Later in the Discussion section, I will compare them with my system.

3.2.1 ChatGPT

When given the exact four topics used to produce Table 2 and asked to summarize current business news about them, GPT-4.1-mini produced remarkable result (link). For each topic, it summarizes three recent examples that are all relevant. However, from the sources it cites, a few issues can be identified:

1. Some of the search results of ChatGPT did not originate from reputable Business news publishers, including sources from "The Motley Fool" and "AInvest".
2. ChatGPT failed to find only sources from 2025, despite the presence of "current" in the prompt.

3.2.2 Google

The four topics are each separatedly given to the Google search engine, using the phrase "Current business news about [topic]". Results: 1, 2, 3, 4.

Compared with ChatGPT, Google performed worse for all four topics. Each results page contains some websites that describes/elaborates on the topic itself or websites that act as a collection of articles on the topic, instead of Business news about it.

However, sometimes the AI Overview would trigger for the search results, but that shares the same problems ChatGPT has. Worse than ChatGPT, Google's version will also include webpages that are not news articles at all. Figure 5 displays Google's top results with and without its AI Overview.

Note that the results are purely identified by the URIs with the search queries embedded in. As Google continues to update its service, there is no guarantee that the observed results will remain the same.

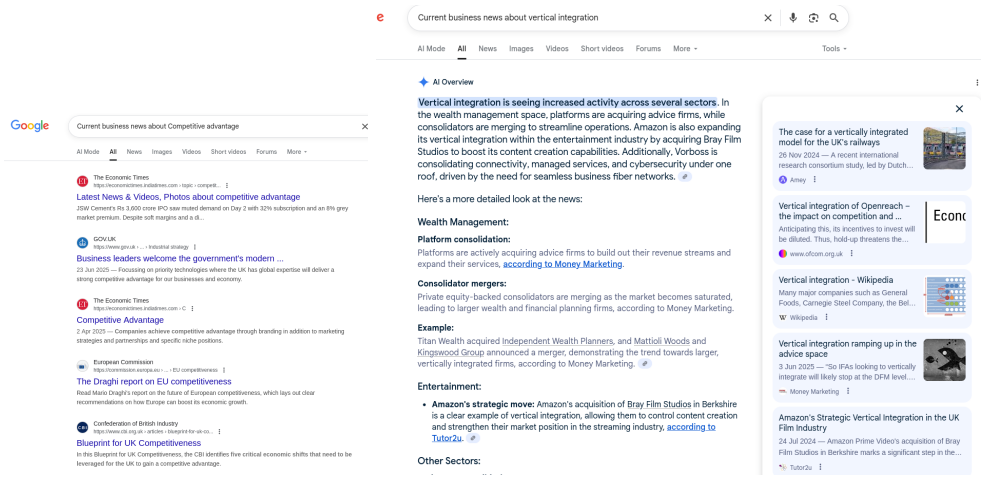

Figure 5: Google results on the topics

4 Discussion

4.1 Analysis of results

4.1.1 Alignment with design

The system conforms to my design philosophy and methodology strictly, except in a few places.

Carefully designed OOP class hirerachy and interfaces are used in both C++ programs and Python scripts; interfaces of classes are chosen to make them highly cohesive and loosely coupled. In C++, `const` is widely applied to ensure immutability of data classes after construction. Comments cover all functions, except for trivial ones whose name gives its meaning. Unit testing were conducted for important features, althought time constraints prevented me to test all programs. These together adhere to design philosophies 1–3.

Executables are separated to mostly doing one thing, often taking another's output as input. This adheres to design philosophy 4.

Except for the optional feature of accepting user feedback and self-improving, all features mentioned in methodology are implemented.

4.1.2 Fulfilment of goals

Goal 1(a) is fulfilled, since the user only needs to give unstructured text/images as input. One limitation of the system, however, is the lack of support for other different file types, like lecture notes in PDF.

Goal 1(b) is fulfilled, by having a custom search engine only indexing reputable Business news websites, and having both it and Google search only results in specific date ranges. Both of the measures are deterministic, providing reliability and accuracy to the system. Conversely, site-limiting indexing inherently is inflexible and lack coverage of good information from less reputable sources. A fixed range of date also excludes the possibility to include old discussions of concurrent events that spans a long time.

Goal 2 is fulfilled by the pipeline, which can be schedulled for execution, once the user configured the system. Still, the automation can be further improved by having a LLM generate all the configuration, with only a natural language description from the user. Alternatively, I could exchange some automation for more accuracy of the system, by having the pipeline interrupt the user in the middle, asking her to select from the generated subtopics/search queries.

Goal 3 is fulfilled by indexing only specific websites in the custom search engine, too. Beside the inflexibility and coverage problem mentioned earlier, the metric of reputability is inherently subjective. In this project, I took advice from my supervisor and consorted lists of ranking of Business news websites [22, 16].

Goal 4 is partially fulfilled by using LLM only on real data from the search engines. However, this cannot stop LLMs from hallucinating when generating search queries or summarizing them.

The optional Goal 5 is not fulfilled.

4.2 Comparsion with mainstream tools

Against ChatGPT, my system successfully eliminated search results of non-reputable sites. However, my system produced a few irrelavant search queries and could not compete with ChatGPT in summary quality.

Versus Google, my system retrieved more reputable and current information, but lacked in breadth. When Google's AI overview was triggered, its summary quality also appeared to be superior.

Compared with the surveyed studies combining search engine of LLMs which have a focus (either LLM4Search or Search4LLM) [78], my system is a more neutral combination with a more general framework for information retrieval. Orthognoal to their work aiming to address specific weakness of one of the two, my work aims primarily to solve the specific information retrieval problem.

4.3 Challenges and future plans

The biggest engineering challenge overcome was implementing the custom search engine. The indexing algorithm is simple, but a search engine spans many different areas of computer science, networking, concurrency, program design, database, searching and indexing. Many third-party libraries had to be picked and used. Moreover, information retrieval on the current Internet is messy. One particular problem to address was to evade blocking to my search engine indexer.

The most significant optimization challenge was to fine-tune my pipeline and LLMs. Tuning the prompts is complicated; changing a prompt to solve a current problem may trigger another problem. The multi-stage design was consequently implemented, to allow the LLM to at least work well for some substages.

The system is significantly limited by the time-span of the project and my resources. Particularly, I had no fund to purchase commercial LLM APIs and could only rely on much less powerful local LLMs, constrained by local computing resources. Additionally, the indexing database of my custom search engine has to be built from scratch. Conversely, the minimal nature of my system enables it to become a free personal solution running on PCs.

Most of the system's capabilities could be enhanced with more resources. Nevertheless, there are a few design limitations to improve next:

1. Currently, the LLMs and search engines are loosely coupled (design philosophy 4). But in many cases integrating them more deeply (e.g. at the LLM model level) could be beneficial.
2. Some parts are hard-coded to work for Business topics, damaging its potential to extend for general topics.
3. The system is not designed to integrate with teaching platforms.
4. The system is not designed to be cross-platform.

5 Conclusion

I successfully designed a pipeline and developed a corresponding system that combines LLMs with search engines to supply current content in business school teaching. The system complements each individual tool substantially with another, by limiting indexed websites, deterministic search result filters, and multi-stage LLM invocation. Moreover, the system achieves more automation than mainstream similar tools.

The methods worked considerably well in fulfilling my project goals. However, the system faces significant challenges to become a mature tool for general usage. While many of them could be mitigated by more resources, such as purchasing a more powerful LLM, or devoting more time to make it cross-platform, the system's design choices exhibit some inflexibility, leaning towards Business topics.

Although my system contributes as an engineering solution and an innovative framework to integrate search engine and LLMs, it is not a theoretical improvement of them; intrinsic problems like hallucination of LLMs and non-semantical understanding of queries of search engines are still present. Future work could also improve the two inherently and replace them in the framework.

References

[1] E. F. W. Alexanderson. Transatlantic radio communication. *Proceedings of the American Institute of Electrical Engineers*, 38(10):1077–1093, 1919.

[2] Muhammad Bello Aliyu. Efficiency of boolean search strings for information retrieval. *American Journal of Engineering Research*, 6(11):216–222, 2017.

[3] Abhijit Anand, Venktesh V, Vinay Setty, and Avishek Anand. Context aware query rewriting for text rankers using llm, 2023.

[4] Anthropic. Api rate limits. `https://docs.anthropic.com/en/api/rate-limits`, 2025. Accessed 29 July 2025.

[5] Tim Baek and The Open WebUI Team. Open webui. `https://docs.openwebui.com/`, 2025. Accessed 31 July 2025.

[6] W. J. G. Beynon. Marconi, radio waves, and the ionosphere. *Radio Science*, 10(7):657–664, 1975.

[7] Kevin M. Bonney. Case study teaching method improves student performance and perceptions of learning gains. *Journal of Microbiology & Biology Education*, 16(1):21–28, 2015.

[8] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 17–23 Jul 2022.

[9] Todd Bridgman, Stephen Cummings, and Colm McLaughlin. The case method as invented tradition: revisiting harvard's history to reorient management education. In *Academy of Management Proceedings*, volume 2015, page 11637. Academy of Management Briarcliff Manor, NY 10510, 2015.

[10] C Roland Christensen. Teaching and the case method harvard business school. `https://www.hbs.edu/teaching/case-method/Pages/default.aspx`, 1987.

[11] Tom Clark. Case method in the digital age: how might new technologies shape experiential learning and real-life story telling? LSE Impact of Social Sciences, 2016.

[12] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[13] Qinxu Ding, Ding Ding, Yue Wang, Chong Guan, and Bosheng Ding. Unraveling the landscape of large language models: a systematic review and future perspectives. *Journal of Electronic Business & Digital Economics*, 3(1):3–19, 2023.

[14] Rebekka Eckhaus. Supporting the adoption of business case studies in esp instruction through technology. *Asian ESP Journal*, 14:280–281, 2018.

[15] Gabriele Falciasecca. Marconi's early experiments in wireless telegraphy, 1895. *IEEE Antennas and Propagation Magazine*, 52(6):220–221, 2010.

[16] FeedSpot. Top 60 business news websites in 2025. https://news.feedspot.com/business_news_websites/, 2025. Accessed 12 August 2025.

[17] J.E. Flood. Alexander graham bell and the invention of the telephone. *Electronics and Power*, 22:159–162, 1976.

[18] Julie Giguere. Leveraging large language models to extract terminology. In Raquel Lázaro Gutiérrez, Antonio Pareja, and Ruslan Mitkov, editors, *Proceedings of the First Workshop on NLP Tools and Resources for Translation and Interpreting Applications*, pages 57–60, Varna, Bulgaria, September 2023. INCOMA Ltd., Shoumen, Bulgaria.

[19] Max Goldman and Rob Miller. 6.031: Software construction. https://web.mit.edu/6.031/www/sp22/, 2022.

[20] Google Developers. Usage limits. https://support.google.com/programmable-search/answer/9069107?hl=en, 2025. Accessed: 28 July 2025.

[21] J.-M. Griffiths and D.W. King. Us information retrieval system evolution and evaluation (1945-1975). *IEEE Annals of the History of Computing*, 24(3):35–55, 2002.

[22] Vinayak Guhanarayan. The 10 best business websites for the latest news and insights. https://www.makeuseof.com/best-business-news-websites, 2023. Accessed 12 August 2025.

[23] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020.

[24] Brian N. Hall. The british army and wireless communication, 1896–1918. *War in History*, 19(3):290–321, 2012.

[25] John S Hammond. *Learning by the case method*. Harvard Business School Boston, MA, 1980.

[26] R. V. L. Hartley. Transmission of information. *Bell System Technical Journal*, 7(3):535–563, 1928.

[27] Raymond A. Heising. Modulation methods. *Proceedings of the IRE*, 50(5):896–901, 1962.

[28] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2), January 2025.

[29] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering, 2021.

[30] Brian W. Kernighan and John R. Mashey. The unix™ programming environment. *Software: Practice and Experience*, 9(1):1–15, 1979.

[31] Andrei Kucharavy. Fundamental limitations of generative llms. In *Large Language Models in Cybersecurity: Threats, Exposure and Mitigation*, pages 55–64. Springer Nature Switzerland Cham, 2024.

[32] Pranjal Kumar. Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260, 2024.

[33] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, October 2009.

[34] 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. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[35] Kenneth B Lifshitz. *Makers of the Telegraph: Samuel Morse, Ezra Cornell and Joseph Henry*. McFarland, 2017.

[36] Barbara Liskov and Stephen Zilles. Programming with abstract data types. *SIGPLAN Not.*, 9(4):50–59, March 1974.

[37] Alberto Lusoli. Teaching business as business: The role of the case method in the constitution of management as a science-based profession. *Journal of Management History*, 26(2):277–290, 2020.

[38] Reza Yousefi Maragheh, Chenhao Fang, Charan Chand Irugu, Parth Parikh, Jason Cho, Jianpeng Xu, Saranyan Sukumar, Malay Patel, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Llm-take: Theme-aware keyword extraction using large language models. In *2023 IEEE International Conference on Big Data (BigData)*, pages 4318–4324, 2023.

[39] Donovan A McFarlane. Guidelines for using case studies in the teaching-learning process. *College Quarterly*, 18(1):n1, 2015.

[40] Fengran Mo, Abbas Ghaddar, Kelong Mao, Mehdi Rezagholizadeh, Boxing Chen, Qun Liu, and Jian-Yun Nie. Chiq: Contextual history enhancement for improving query rewriting in conversational search, 2024.

[41] Calvin Mooers. The theory of digital handing of non-numerical information and its implications to machine economics. In *Proceedings of the meeting of the Association for Computing Machinery at Rutgers University*, 1950.

[42] Samuel Finley Breese Morse. First telegraph message. Retrieved from the Library of Congress, `https://www.loc.gov/item/mcc.019/`, May 1844.

[43] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022.

[44] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[45] Rhoda Okunev. *History of the Internet, Search Engines, and More*, pages 9–16. Apress, Berkeley, CA, 2023.

[46] Ollama developers. Ollama. `https://ollama.com/`, 2025. Accessed 29 July 2025.

[47] Ollama developers. Search models. `https://ollama.com/search`, 2025. Accessed 29 July 2025.

[48] OpenAI. Api rate limits. `https://platform.openai.com/docs/guides/rate-limits`, 2025. Accessed 29 July 2025.

[49] Akshita Patil, Jayesh Pamnani, and Dipti Pawade. Comparative study of google search engine optimization algorithms: Panda, penguin and hummingbird. In *2021 6th International Conference for Convergence in Technology (I2CT)*, pages 1–5, 2021.

[50] Sandeep Puri. Effective learning through the case method. *Innovations in Education and Teaching International*, 59(2):161–171, 2022.

[51] Python Software Foundation. tkinter — python interface to tcl/tk. `https://docs.python.org/3/library/tkinter.html`, 2025. Accessed 7 August 2025.

[52] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018.

[53] Shivangi Raman, Vijay Kumar Chaurasiya, and S. Venkatesan. Performance comparison of various information retrieval models used in search engines. In *2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, pages 1–4, 2012.

[54] Ankush Raut, Xiaofeng Zhu, and Maria Leonor Pacheco. Can llms interpret and leverage structured linguistic representations? a case study with amrs, 2025.

[55] Vipula Rawte, Amit Sheth, and Amitava Das. A survey of hallucination in large foundation models, 2023.

[56] Tohida Rehman, Soumabha Ghosh, Kuntal Das, Souvik Bhattacharjee, Debarshi Kumar Sanyal, and Samiran Chattopadhyay. Evaluating llms and pre-trained models for text summarization across diverse datasets, 2025.

[57] Michael Reynolds. Research on the application of large language models in classification and indexing. *Preprints*, May 2025.

[58] Walid S Saba. Stochastic llms do not understand language: towards symbolic, explainable and ontologically based llms. In *International conference on conceptual modeling*, pages 3–19. Springer, 2023.

[59] Erin Sanu, T Keerthi Amudaa, Prasiddha Bhat, Guduru Dinesh, Apoorva Uday Kumar Chate, and Ramakanth Kumar P. Limitations of large language models. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–6, 2024.

[60] Steven A Sass. *The pragmatic imagination: A history of the Wharton School, 1881-1981*. University of Pennsylvania Press, 2016.

[61] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[62] Mischa Schwartz and David Hochfelder. Two controversies in the early history of the telegraph. *IEEE Communications Magazine*, 48(2):28–32, 2010.

[63] Tom Seymour, Dean Frantsvog, Satheesh Kumar, et al. History of search engines. *International Journal of Management & Information Systems (IJMIS)*, 15(4):47–58, 2011.

[64] Binod Shah. Case method of teaching in management education. *Research Gate*, 2019.

[65] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[66] Xiang Shi, Jiawei Liu, Yinpeng Liu, Qikai Cheng, and Wei Lu. Know where to go: Make llm a relevant, responsible, and trustworthy searchers. *Decision Support Systems*, 188:114354, 2025.

[67] Elliot N. Sivowitch. A technological survey of broadcasting's "pre-history," 1876–1920. *Journal of Broadcasting*, 15(1):1–20, 1970.

[68] John-Christopher Spender. The business school in america: a century goes by. *The future of business schools: Scenarios and strategies for*, pages 9–18, 2020.

[69] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore, December 2023. Association for Computational Linguistics.

[70] Munief Hassan Tahir, Sana Shams, Layba Fiaz, Farah Adeeba, and Sarmad Hussain. Benchmarking the performance of pre-trained llms across urdu nlp tasks, 2024.

[71] Carlos JO Trejo-Pech and Susan White. The use of case studies in undergraduate business administration. *Revista de administração de empresas*, 57:342–356, 2017.

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[73] Klára Vítečková, Tobias Cramer, Matthias Pilz, Janine Tögel, Sascha Albers, Steven van den Oord, and Tomasz Rachwał. Case studies in business education: an investigation of a learner-friendly approach. *Journal of International Education in Business*, 18(2):149–176, 2025.

[74] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. Freshllms: Refreshing large language models with search engine augmentation, 2023.

[75] Thomas A. Watson. How bell invented the telephone. *Proceedings of the American Institute of Electrical Engineers*, 34(8):1503–1513, 1915.

[76] Martin H. Weik. The eniac story. *Ordnance*, 45(244):571–575, 1961.

[77] JONATHAN REED WINKLER. Telecommunications in world war i. *Proceedings of the American Philosophical Society*, 159(2):162–168, 2015.

[78] Haoyi Xiong, Jiang Bian, Yuchen Li, Xuhong Li, Mengnan Du, Shuaiqiang Wang, Dawei Yin, and Sumi Helal. When search engine services meet large language models: Visions and challenges. *IEEE Transactions on Services Computing*, 17(6):4558–4577, 2024.

[79] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Trans. Knowl. Discov. Data*, 18(6), April 2024.

[80] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR)*, 2023.

[81] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6–es, July 2006.