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 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

29th July 2025

Abstract

Business school teaching using the famous case method requires a large amount of information about latest business events to keep the course relevant. Digital technologies like web search engines and large languages models can offer substantial assistance to this process, yet a few challenges are still present: 1) search engines struggle to process unstructured search queries. 2) while LLMs excel in understanding unstructured input, they often hallucinate and cannot ensure the retrieval of 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: content discovery, information retrieval, LLM, search engines, case method, Business school,

1 Introduction

The project aims to improve personal information retrieval tools for a specific usage: retrieving latest business news for business school teaching. In this introduction, section 1.1 explains why it is important in business school teaching; section 1.2 gives a broad review of the history and current methods of information retrieval. Then, section 1.3 describes how I aim to improve the current status of personal information retrieval systems.

1.1 Project background

Since the emergence of the first Business schools in the late 19th century [53, 45], several distinct pedagogical teaching strategies have been applied. First institutionalized at Harvard Business School [18, 6] 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 [57, 5, 28] than many traditional, for example, big lecture based, teaching methods. Thus, it has found wide adoption across the world [55, 11].

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 [49, 37]. In particular, Christensen identifies in his classical article that instructors have to conduct “extensive preparation” [7] for it. Another factor contributing to this constraint of case method 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 [8]; McFarlane emphasises the importance of updated cases, as otherwise students could be disengaged or discouraged [29].

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 [47, 26], notably with Morse's first telegraph message, "*What hath God wrought?*" in 1844 [31], marked a paradigm shift. This technological innovation was considerably improved by Bell's telephone in 1876 [59, 13], 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 [12] and the first successful transatlantic signal in 1901 [4] introduced electromagnetic wave-based wireless communication, eventually accumulating into the world's first voice broadcast by radio in 1906 [52]. These milestones collectively redefined the temporal and spatial boundaries of information gathering.

In parallel to improving the weaknesses of wireless communication [17, 61, 1], reseachers then worked on a theory of information. Hartley observed a logritham pattern of information capacity [19] and then Shannon expanded on it to first define *bits* and *entropy*, giving a formal mathematical theory of information [50] in 1948. Meanwhile, engineers were experimenting with alternative modulation techniques, notably frequency modulation (FM), and the concepts were formalized in the 1930s [20].

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 [60], which were made to repeat tedious operations fast. Actually, the term *information retrieval (IR)* was not invented until Mooers coined it in 1950 [30]. 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 [16]. 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 [16, 25]. Then commercial search engines emerged in the 1990s, opearting on the scale of the whole Internet [48, 32].

The aforementioned sorting, searching, and indexing algorithms were partial solutions to problem (2.1) and (2.2), with fundamental designs like inverted index [46, chap. 2], [64, sect. 2], boolean search [46, chap. 1], and various clever ideas such as spelling correction (stemming) [46, chap. 3.3], positional indexing [64, sect. 3]. Because the database can be extremely large, specific index construction [46, chap. 4], [64, sect. 5] compression [46, chap. 5] algorithms have appeared. Also, people saw ample opportunity to explore distributed computing, with the most notable model probably being MapReduce [9].

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, 39]. 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 [36].

One major breakthrough in NLP is Vaswani et al.'s transformer [56]. Building upon it, OpenAI created generative pre-trained transformers (GPTs) trained on massive amount of data [38],

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 [10]. 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 [63, 24]. However, because of the inherent limit of neural networks, some argue that they could not formally reason about what they output [44, 23, 43]. Indeed, hallucination [21, 41] 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 [62, 51, 58]. Specifically, Xiong et al. proposes to categorize them into "LLM4Search" and "Search4LLM", where "A4B" means using A to improve B [62]. 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.

1.2.3 Gaps I aim to fill

The contribution I aim make is not about combining LLMs and IR, a direction numerous studies have explored. I identify these gaps in the state-of-the-art IR tools, with or without LLMs:

1. They are still semi-automatic, be it search engines or LLMs. One would have to give a prompt first and wait for answers. I plan to make a mostly automatic system where the IR process will be run automatically based on some configuration.
2. LLM's training data is often outdated. The information retrieved cannot reflect the new events happened very recently. Even if existing tools like RAG can make LLM search the Internet with a search engine, users often lack control over the content the search engine indexes and returns, and may often end up with less relevant or outdated data.

1.3 Goals

Based on the gaps, I plan to develop a software system with these 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 credibility and interpretibility. If pure LLMs lack the two properties, then combination with deterministic and well-designed ranking and searching algorithms in search engines 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.

1.3.1 Not included in the goals

On the other hand, these are beyond the scope of my project:

1. Providing a method to rank business information. This is rather subjective and I believe is better left for the user to judge.
2. Build and train a LLM. That is a lot of work I cannot afford to do. This project will use trained LLMs.

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 [14]. Approach 4 is part of the Unix philosophy [22].

1. *Correct today and in the unknown future*: By reasoning of the system using design concepts like abstract datatypes [27], immutability, and software testing, its current correctness is defended; by defensive programming, 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 could also participate in the process.

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.

`indexer`, `searcher`, `updater`	Build a search enginedatabase 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.
`setup_schedules.py`	Schedule automatic executions for various programs in the system to make the system automatic.

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.

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, the commercial search engines 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 [15] 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.

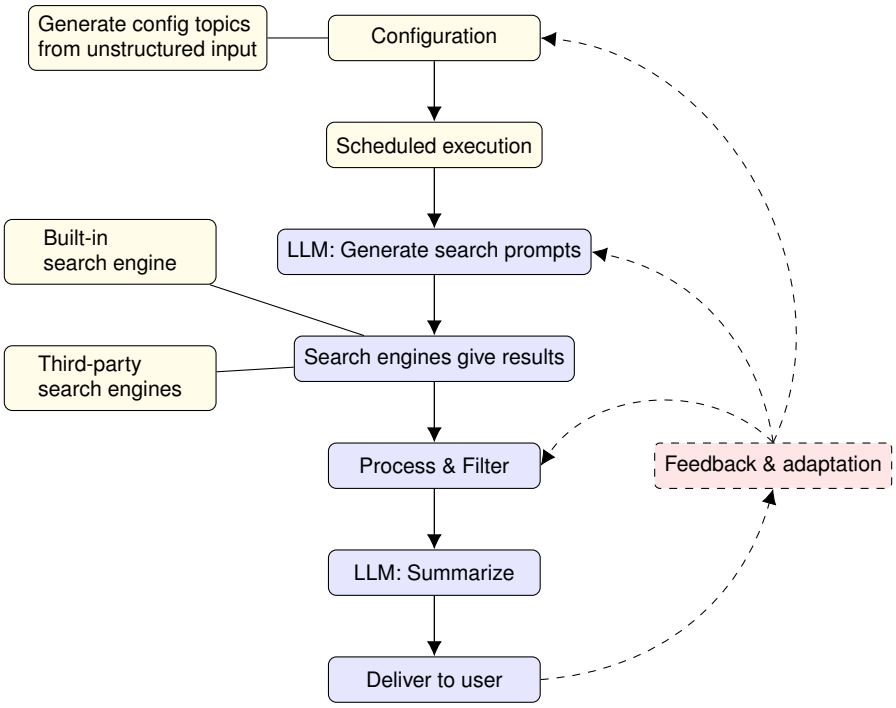

Figure 1: Architecture and operational flow of the software system

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 [35, 3], the project will use local LLMs powered by Ollama [33].

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

Both the search prompt generation and the results sythesis are invoked this way. For each, a different system prompt will be used, carefully crafted by my own interaction with Ollama. They are included in the default configuration (see `config.py`).

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 [34], I chose `llama3.1:8b` for its competitive ability on text generation and summarization, proven by several academic studies and benchmarks [42, 40, 54]. 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 Results delivery

The results are send 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.

3 Results

4 Discussion

5 Conclusion

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

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

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

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

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

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

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

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

[16] 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.

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

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

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

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

[21] 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.

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

[23] 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.

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

[25] 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.

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

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

[28] 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.

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

[30] 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.

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

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

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

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

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

[36] 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.

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

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

[39] 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.

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

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

[42] 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.

[43] 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.

[44] 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.

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

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

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

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

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

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

[51] 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.

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

[53] 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.

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

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

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

[62] 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.

[63] 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.

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