IntelliSMART: Intelligent Semantic Machine-Assisted Research Tool

Aadyant Khatri[1][0009−0000−7574−7043], Nicolas Egierski[1][0009−0001−2738−1689], Ashutosh Pochamreddy[1][0009−0004−4542−1754], Abdulaziz Alhamadani[2][0009−0003−4732−5144], Shailik Sarkar[1][0000−0001−6544−2262], and Chang-Tien Lu[1][0000−0003−3675−0199]

[1] Department of Computer Science, Virginia Tech, Falls Church, VA 22043, USA
[2] Department of Data Science and Business Analytics, Florida Polytechnic University, Lakeland FL
{aadyant, nregierski, ashutoshpocham, shailik, clu}@vt.edu, aalhamadani@floridapoly.edu

Abstract. The exponential growth of academic literature presents significant challenges for researchers attempting to find relevant information. Traditional keyword-based retrieval systems often fail to address issues such as synonyms, homonyms, and semantic nuances, leading to suboptimal search results. This paper introduces a novel system called IntelliSMART (Intelligent Semantic Machine-Assisted Research Tool), which leverages large language models (LLMs) and advanced semantic processing techniques to improve the retrieval of academic literature. Our approach integrates query rewriting, embedding generation, efficient indexing, and complex article retrieval mechanisms to provide highly accurate and contextually relevant results that align with the user's intent. The IntelliSMART system features a user-friendly front end that facilitates intuitive query input, along with a robust back end for handling user queries, generating embeddings, indexing extensive collections of academic papers, and efficiently retrieving the most relevant documents. The proposed system shows significant improvements over conventional methods, highlighting its potential to transform the search experience in academic research.

Keywords: Data mining · Information retrieval · Semantic search · Large language models · Keyword search · Text Generation

1 Introduction

The rise of digital academic resources and scholarly literature necessitates efficient retrieval methods. Traditional keyword-based Information Retrieval (IR) systems struggle with synonyms, homonyms, and contextual understanding. While more complex keyword-based systems address some limitations, they require ongoing refinement, expertise, and cost. Major search engines like Google and Bing have adopted Large Language Models (LLMs) for improved IR, leveraging deep contextual understanding and semantic representation to enhance

information acquisition [1]. Similarly, LLMs offer academia promising avenues to improve traditional IR systems.

IR systems aim to quickly find and prioritize relevant information. Traditional keyword-based, or *Boolean models*, use static semantics and Boolean logic, which can miss linguistic nuances and hinder intuitive searches as digital repositories expand [2]. Binary keyword assessments fail to provide nuanced search results.

Semantic processing helps address these issues by considering textual intricacies and reducing word sense ambiguity. This involves extracting and understanding connotations and meanings within text using statistical models [3]. Accurate sense identification enhances all subsequent IR processes.

Our project applies LLMs in key areas:

Query Rewriter: This crucial stage processes user queries by analyzing keywords, phrases, and questions. It considers contextual nuances and semantic relationships, providing deeper insights into user intent. Techniques like tokenization and semantic analysis help extract relevant information from the queries.

Retriever: As the IR system's core, the retriever fetches relevant documents based on user queries. It involves preprocessing and indexing the document corpus with advanced LLM text embedders and indexers, facilitating efficient retrieval. This process considers textual information, metadata, and latent semantic meanings, ensuring precise and relevant results.

The rest of this proposal is structured as follows: Section 2 reviews related work and the inspiration behind our approach. Section 3 details the proposed approach and implementation. Sections 4 and 5 discuss the data used, evaluation methodology, and the results.

2 Related Work

2.1 Query Rewriting

Conventional strategies for improving retrieval performance involve enriching initial queries with insights from top-ranked documents. Methods like relevance feedback [4–7] and word embedding-based techniques [8, 9] are widely used, but limited by inadequate semantic comprehension and understanding of user intent.

Ad-hoc retrieval, with brief and ambiguous queries, challenges traditional search engines. LLMs excel at understanding language semantics, providing two key benefits: mitigating the "vocabulary gap" and acting as "meaning interpreters" for vague queries [10–13]. LLMs address various ad-hoc retrieval challenges beyond vocabulary expansion and intent clarification [10–15]. In the legal domain, PromptCase [16] uses LLMs to bridge complex legal questions and computer-readable formats, simplifying legal research.

2.2 Retriever

Retrieval models have evolved from statistical algorithms [17] to neural models [18, 19], which better interpret complex user intents and offer improved semantic

understanding. Despite this advancement, challenges persist, such as the brevity and vagueness of user queries and the length and noise of documents, along with the time-consuming and costly process of collecting human-annotated relevance labels.

Large Language Models (LLMs) significantly enhance text processing capabilities, offering more accurate query and document understanding compared to smaller models [20]. Research into model scale's impact on retrieval performance has involved using LLMs as retrieval encoders [21–23]. For instance, OpenAI [21] employs adjacent text segments as positive pairs in the unsupervised pre-training of text embedding models, with parameter values ranging from 300M to 175B. Additionally, incorporating task-specific instructions, as demonstrated by TART [23], can enhance retrieval performance by aligning the model's capabilities with user search intentions across various tasks.

2.3 Text Embeddings

In NLP, word embeddings transform words into numerical vectors to capture semantic similarity and aid various language tasks. Techniques such as Word2Vec [24] and GloVe generate these vectors, reflecting semantic and relational information [25]. Trained on large datasets [26], embeddings streamline feature engineering in neural networks.

There are two main approaches: (1) Feature-based embeddings, where pre-trained networks generate static or dynamic embeddings, and (2) Fine-tuning embeddings, where models initially trained on general tasks are fine-tuned for specific NLP tasks. Static embeddings remain consistent across contexts, while dynamic embeddings adapt to context, addressing polysemy.

Word2Vec [24] uses techniques like Continuous Bag of Words (CBOW) and Skip-gram to predict target words from context or vice versa. These methods are efficient and reduce computational complexity.

GloVe differs by integrating local context windows with global matrix factorization, providing a broader context than Word2Vec.

ELMo [27] advances embeddings by using a bidirectional LSTM to generate context-sensitive representations influenced by the entire input sentence.

BERT [28] employs a bidirectional Transformer encoder to predict masked words using both left and right contexts, offering deep bidirectional representations pre-trained on unlabeled text.

3 Proposed Approach

Building the semantic processing LLM for academic literature retrieval requires the development and integration of the following essential components:

3.1 Query Rewriting

The query rewriter module enhances user queries through a structured One-shot Learning process.

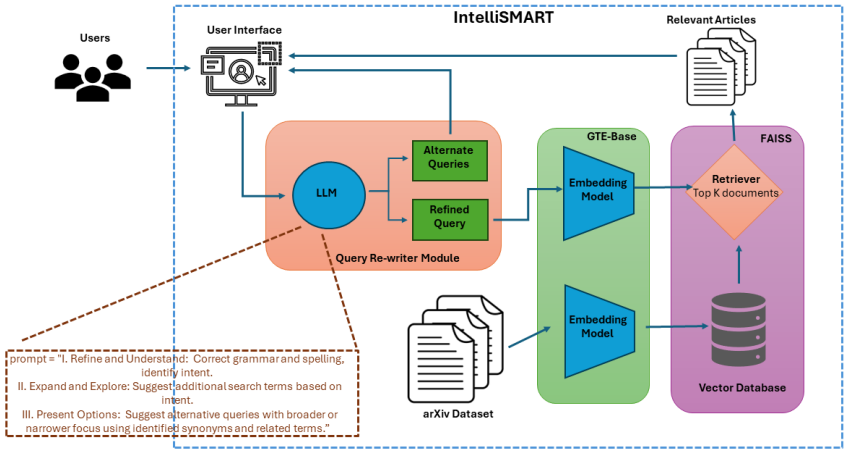

Fig. 1. Proposed Architecture

Notation: Let $A_Q = \{q^{(i)}\}_{i=1}^{N_Q}$ be the set containing user queries with N_Q samples, where $q^{(i)}$ is a textual user query. This scenario describes a query rewriter module that leverages a One-shot Learning approach, categorized into two stages: Refining and Understanding and Expanding and Exploring.

Refine and Understand The initial step involves correcting grammatical errors, spelling inaccuracies, and redundant words. For example, the query "The causes of global warming and what effect it has on environment" is refined to "The causes of global warming and what effects it has on the environment." This ensures queries adhere to linguistic norms, facilitating accurate semantic understanding and effective processing. The module then progresses to the Expanding and Exploring stage.

Expanding and Exploring In this step, the module interprets query intent by understanding subtle meanings and broader context. It enhances queries with synonyms and related terms, such as "climate change" and "greenhouse gases" for global warming, and suggests alternative queries to better capture user needs. This approach improves user experience and search result accuracy.

Prompt Construction: For each user query $q^{(i)} \in A_Q$, the system generates a prompt $p^{(i)}$ for the One-shot Learning process:

(1) **System Description** ($p^{(i)}_{\text{system}}$): Provides an overview of the query rewriter's functionalities. Example: *You are a query refinement system designed to enhance user search queries and suggest alternate queries. Example query: 'The causes of global warming and what effect it has on environment.'*

(2) **Task Description** ($p^{(i)}_{\text{task}}$): Outlines the One-shot Learning objective. Example:

I. Refine and Understand: Correct grammar and spelling: 'The causes of global warming and what effects it has on the environment.' Identify intent: User seeks articles on the causes and impacts of global warming. Add synonyms and related terms: 'global warming', 'climate change', 'greenhouse gases', 'anthropogenic factors'

II. Expand and Explore: Suggest additional terms: 'environmental impact', 'climate change effects', 'mitigation strategies'.

III. Present Options: Offer refined query: **'Climate change causes and environmental consequences.'** Suggest alternative queries: 'Anthropogenic factors in climate change' or 'Effects of climate change on specific ecosystems'.

(3) **Input** ($p^{(i)}_{\text{input}}$): Represents the user's original query, which the LLM processes without additional context.

The complete prompt for the i^{th} query is formulated by combining these elements:

$$p^{(i)} = p^{(i)}_{\text{system}} \,||\, p^{(i)}_{\text{task}} \,||\, p^{(i)}_{\text{input}}$$

3.2 Embedding Generation

We use SOTA embedding models like GTE-Base Sentence Transformer and Google's Gemini to encode academic articles and queries into high-dimensional vectors. These embeddings effectively measure semantic similarity, improving query-article matching and retrieval efficiency by positioning related concepts closer in vector space. Using both models ensures robust performance and enables comparative analysis.

Google's Gemini Embedding Model provides numerical representations for words, phrases, and sentences, supporting applications like semantic search and text classification with API customization and binary quantization for efficient storage. The **GTE-Base Sentence Transformer**, based on the BERT framework, generates dense representations for semantic similarity comparisons [29]. It features English proficiency, a 512-token maximum sequence length, a 768-dimensional embedding, and a model size of 0.22GB.

3.3 Storing and Retrieving Vector Embeddings

We use ANNOY and FAISS indexing systems to store academic article embeddings, which are computed during the initial data preparation phase. This setup ensures efficient storage and fast retrieval of relevant articles, improving overall retrieval process efficiency.

ANNOY excels in Approximate Nearest Neighbor Search, making it ideal for scenarios where efficiency is prioritized over exact matches. It uses randomized trees to build indexes, facilitating quick approximate queries while being memory-efficient and supporting disk-based indexes for large datasets through memory-mapped files. ANNOY handles various distance metrics, such as Euclidean and cosine, and is commonly used in image and document search tasks.

FAISS, on the other hand, offers efficient similarity search with algorithms like Index FlatL2, IVFFlat, and HNSW for high-dimensional vector spaces. It includes vector clustering, scales well with large datasets, and benefits from GPU acceleration for faster processing. FAISS's C++ core library, with Python bindings, integrates smoothly into scientific computing workflows.

Upon receiving a user query, it is processed by the Query Rewriting module and embedded. The embeddings are compared with stored article embeddings using FAISS and ANNOY for efficient nearest neighbor search. We then retrieve and rank the top k publications by similarity scores, presenting the results in a structured DataFrame. This DataFrame includes text, authors, DOIs, and similarity distances, providing a comprehensive view of the top k search results for an intuitive user experience.

3.4 User Interface

Users interact with IntelliSMART via an intuitive interface that efficiently retrieves relevant publications for queries like "articles about time travel." The system ranks results by relevance and features a user-friendly design with a consistent color scheme, clear typography, and visual feedback such as highlighted selections. Its responsive design ensures a smooth experience across different devices.

Our project integrates a React-based front end and a FastAPI-powered back end. The front end, using JavaScript and Tailwind CSS, delivers a dynamic, responsive user experience with efficient state management via 'useState' and 'useContext'. It fetches and displays data in JSON format, listing publications by similarity and providing details like title, authors, publication date, and abstracts, along with hyperlinks and filters for refined searches by author, date range, and view mode.

The back end, developed with FastAPI and hosted on Uvicorn, ensures high performance and scalability through asynchronous operations. It processes and optimizes user queries, generating related queries and converting them into embeddings for semantic similarity matching. This setup facilitates responsive search experiences by returning relevant results and suggestions to the front end, allowing seamless query navigation and refinement.

4 Evaluation Methodology

4.1 Dataset

The Cornell University arXiv dataset [30] is our main data source, containing over 2.4 million entries with scholarly article metadata, including titles, abstracts, author details, and categorization. It also provides access to some full-text PDFs through platforms like Kaggle, DataCite's API, arXiv's open API, and an AWS S3 bucket. This dataset supports trend analysis, literature search tools, and network construction, making it crucial for understanding and advancing STEM research.

4.2 Evaluation Metric

Information retrieval metrics fall into online and offline categories. Online metrics like session abandonment and click-through rates require real user data, which isn't feasible at our current stage. Offline metrics such as precision and recall need labeled datasets, which we don't have. Thus, we developed an alternative evaluation approach. Our method involves generating synthetic queries with Gemini LLM and Zero-shot Learning, simulating real user searches based on the provided articles to assess retrieval effectiveness.

Prompt Construction:

For each article, the system generates a prompt $p^{(i)}$ that guides the zero-shot learning process. The prompt is constructed as follows:

(1) Task Description: The task, denoted as $p^{(i)}_{\text{task}}$, outlines the objective of the zero-shot learning process. For instance: *Generate a short human-like user query that should return the following academic article on being searched upon.*

(2) Input: Denoted as $p^{(i)}_{\text{article}}$, this component represents the i^{th} article in the dataset. The LLM operates on $p^{(i)}_{\text{article}}$ without any additional processing or context added.

The complete prompt for the i^{th} article is formulated by combining these elements:

$$p^{(i)} = p^{(i)}_{\text{task}} \mid\mid p^{(i)}_{\text{article}}$$

Subsequently, with these generated synthetic queries, we deploy IntelliSMART to retrieve the most semantically similar articles from our database. Given our crafted queries, we possess a priori knowledge of the correct articles that should ideally be retrieved, facilitating meticulous evaluation against the ground truth—the correct articles identified during query creation.

	Top k Documents			
Retrieval System	**k = 1**	**k = 3**	**k = 5**	**k = 10**
Sklearn TF-IDF	64.03%	77.33%	81.73%	86.68%
BM25	63.83%	76.16%	79.96%	83.88%
Gemini Embedding, Annoy Indexing	79.78%	88.28%	89.43%	90.63%
GTE-BASE Embedding, Annoy Indexing	85.35%	92.60%	93.95%	95.20%
GTE-BASE Embedding, FAISS Indexing	87.58%	95.23%	96.65%	97.98%
Gemini Embedding, FAISS Indexing	85.78%	95.13%	96.50%	97.88%

Table 1. Top k document retrieval by Retrieval System combination

In the evaluation phase, we examine the top 1, 3, 5, and 10 retrieved articles for each synthetic query to understand performance across the different retrieval depths. We analyze the percentage of queries for which the correct corresponding article is present within the retrieved set, assessing our model's ability to capture semantic relevance. This approach enables assessment of the model's robustness and efficacy in delivering semantically aligned search results.

Additionally, we benchmark our model against established content-based search benchmarks, specifically Sklearn's Tf-Idf and BM25.

Scikit-learn's Tf-Idf implementation measures term importance by combining term frequencies within documents with inverse document frequencies, creating a compact representation for efficient retrieval. **BM25** enhances term frequency scoring by using logarithmic functions to address saturation effects and adjusts inverse document frequency to counteract common term biases. It also normalizes document length and introduces saturation thresholds for both term frequency and inverse document frequency components.

Retrieval System	Data Preparation Time	Stored Data Size	Inferencing Time
Sklearn TF-IDF	27.4 s	90.94 Mb	11.46 mins
BM25	20 mins	841.1 Mb	60.93 mins
Gemini Embedding, Annoy Indexing	40 mins + 0.5 mins	934.3 Mb	27.63 mins
GTE-BASE Embedding, Annoy Indexing	7.5 mins + 0.5 mins	907.4 Mb	8.71 mins
GTE-BASE Embedding, FAISS Indexing	7.5 mins + 0.23 s	293.7 Mb	16.8 mins
Gemini Embedding, FAISS Indexing	40 mins + 0.25 s	294.7 Mb	37.1 mins

Table 2. Retrieval System Combination Statistics

5 Results

Evaluation results from approximately 1,000 queries reveal that Sklearn TF-IDF and BM25 models perform well, with accuracies ranging from 64% to 87%, peaking at 86.6% for Sklearn TF-IDF and 83.8% for BM25 at k=10. In contrast, IntelliSMART with Gemini Embedding and Annoy or FAISS indexing consistently outperforms these methods, with the GTE-BASE Embedding and FAISS indexing achieving the highest accuracy of 97.98% at k=10. These results highlight the superior effectiveness of embedding-based retrieval systems, particularly with advanced indexing like FAISS, in improving document relevance. Table 2 highlights the efficiency and scalability of different retrieval systems in terms of data preparation time, stored data size, and inference time for 1000 queries. Traditional models like Sklearn TF-IDF and BM25 have shorter data preparation times (27.4 seconds and 20 minutes, respectively) but longer inference times (11.46 and 60.93 minutes). In contrast, IntelliSMART with Gemini Embedding and advanced indexing techniques like FAISS, especially GTE-BASE Embedding with FAISS, excels in both data preparation and inference. Despite longer data preparation, these embedding-based models offer faster inference, with GTE-BASE Embedding and Annoy achieving the shortest inference time of 8.71 minutes. However, they require more storage compared to traditional methods, demonstrating the trade-offs between preparation time, storage, and inference speed.

6 Conclusion

In conclusion, traditional keyword-based search engines struggle with the growing volume of academic literature and lack nuance. Integrating large language models (LLMs) with semantic processing, as seen with IntelliSMART, significantly improves query precision and retrieval. Evaluation shows that embedding-based systems with advanced indexing, like FAISS, outperform traditional methods, positioning IntelliSMART as a potential game-changer in academic article retrieval.

References

1. Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," 2024.
2. R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The concepts and technology behind search*, 2nd ed. USA: Addison-Wesley Publishing Company, 2011.
3. R. Mao, K. He, X. Zhang, G. Chen, J. Ni, Z. Yang, and E. Cambria, "A survey on semantic processing techniques," *Information Fusion*, vol. 101, p. 101988, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253523003044
4. N. Abdul-Jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, M. D. Smucker, and C. Wade, "Umass at trec 2004: Novelty and hard," *Computer Science Department Faculty Publication Series*, p. 189, 2004.
5. C. Zhai and J. Lafferty, "Model-based feedback in the language modeling approach to information retrieval," in *Proceedings of the tenth international conference on Information and knowledge management*, 2001, pp. 403–410.
6. D. Metzler and W. B. Croft, "A markov random field model for term dependencies," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 472–479.
7. Z. Zheng, K. Hui, B. He, X. Han, L. Sun, and A. Yates, "Bert-qe: contextualized query expansion for document re-ranking," *arXiv preprint arXiv:2009.07258*, 2020.
8. F. Diaz, B. Mitra, and N. Craswell, "Query expansion with locally-trained word embeddings," *arXiv preprint arXiv:1605.07891*, 2016.
9. S. Kuzi, A. Shtok, and O. Kurland, "Query expansion using word embeddings," in *Proceedings of the 25th ACM international on conference on information and knowledge management*, 2016, pp. 1929–1932.
10. I. Mackie, I. Sekulic, S. Chatterjee, J. Dalton, and F. Crestani, "Grm: Generative relevance modeling using relevance-aware sample estimation for document retrieval," *arXiv preprint arXiv:2306.09938*, 2023.
11. K. Srinivasan, K. Raman, A. Samanta, L. Liao, L. Bertelli, and M. Bendersky, "Quill: Query intent with large language models using retrieval augmentation and multi-stage distillation," *arXiv preprint arXiv:2210.15718*, 2022.
12. J. Feng, C. Tao, X. Geng, T. Shen, C. Xu, G. Long, D. Zhao, and D. Jiang, "Knowledge refinement via interaction between search engines and large language models," *arXiv preprint arXiv:2305.07402*, 2023.
13. I. Mackie, S. Chatterjee, and J. Dalton, "Generative and pseudo-relevant feedback for sparse, dense and learned sparse retrieval," *arXiv preprint arXiv:2305.07477*, 2023.

14. L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," *arXiv preprint arXiv:2212.10496*, 2022.
15. R. Jagerman, H. Zhuang, Z. Qin, X. Wang, and M. Bendersky, "Query expansion by prompting large language models," *arXiv preprint arXiv:2305.03653*, 2023.
16. Y. Tang, R. Qiu, and X. Li, "Prompt-based effective input reformulation for legal case retrieval," in *Australasian Database Conference*. Springer, 2023, pp. 87–100.
17. S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford *et al.*, "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.
18. V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.
19. L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," *arXiv preprint arXiv:2007.00808*, 2020.
20. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
21. A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy *et al.*, "Text and code embeddings by contrastive pre-training," *arXiv preprint arXiv:2201.10005*, 2022.
22. X. Ma, L. Wang, N. Yang, F. Wei, and J. Lin, "Fine-tuning llama for multi-stage text retrieval," *arXiv preprint arXiv:2310.08319*, 2023.
23. A. Asai, T. Schick, P. Lewis, X. Chen, G. Izacard, S. Riedel, H. Hajishirzi, and W.-t. Yih, "Task-aware retrieval with instructions," *arXiv preprint arXiv:2211.09260*, 2022.
24. J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.
25. M. Li, H. Zhuang, K. Hui, Z. Qin, J. Lin, R. Jagerman, X. Wang, and M. Bendersky, "Generate, filter, and fuse: Query expansion via multi-step keyword generation for zero-shot neural rankers," *arXiv preprint arXiv:2311.09175*, 2023.
26. A. Anand, V. Setty, A. Anand *et al.*, "Context aware query rewriting for text rankers using llm," *arXiv preprint arXiv:2308.16753*, 2023.
27. J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "Pretrained language models for text generation: A survey," *arXiv preprint arXiv:2201.05273*, 2022.
28. B. Mitra and N. Craswell, "Neural models for information retrieval," *arXiv preprint arXiv:1705.01509*, 2017.
29. Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, "Towards general text embeddings with multi-stage contrastive learning," *arXiv preprint arXiv:2308.03281*, 2023.
30. arXiv.org submitters, "arxiv dataset," 2024. [Online]. Available: https://www.kaggle.com/dsv/7548853