JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY

Semantic Similarity Search on Product Catalogues

Abstract Please provide an abstract of 100 to 250 words. The abstract should clearly state the nature and significance of the paper. It must not include undefined abbreviations, mathematical expressions or bibliographic references.

Keywords keyword, keyword, keyword, keyword, keyword [Keywords should closely reflect the topic and should optically characterize the paper. Please use about 3∼5 keywords or phrases in alphabetical order separated by commas.]

1 Introduction

2 Content

Regular Paper

3 Literature Review

In this section we will go over the literature related to our project, outlining the semantic search pipeline, as well as the document embedding and retrieval

3.1 Semantic Search

Semantic search aims to enhance information retrieval by interpreting the intent behind user queries and the contextual meaning of terms, rather than relying solely on lexical matching. Traditional semantic search systems often incorporate user-specific signals such as geographic location, browsing history, or long-term user behavior to personalize results. However, such contextual signals are absent in the present work, necessitating alternative approaches that are intrinsic to the corpus itself. This section reviews key strategies and literature related to corpus-centric semantic search, with particular attention to intelligent query understanding and semantic parsing.

3.1.1 Intelligent Query Understanding

In the absence of user-derived context, the burden of semantic interpretation shifts toward the structure and vocabulary of the underlying corpus. A critical area of innovation in this space is intelligent query expansion. Rather than employing generic linguistic resources (e.g., WordNet) for term substitution, a more effective strategy involves extracting expansion candidates directly from the corpus. This approach -herein referred to as *Intelligent Substitution* leverages co-occurrence patterns, domain-specific terminology, and observed naming conventions.

For example, the query “3 mm bolt” might be semantically expanded to include “0.118 inch bolt” if corpus data reveals frequent usage of imperial measurements. Similarly, terms such as “bolt” may be semantically linked to “fastener” or “screw” based on their syntagmatic relationships within the domain. This corpus-aware expansion strategy increases retrieval precision by aligning query representation with the semantic structure of the indexed data.

3.1.2 Semantic Parsing

Semantic parsing is a critical technique in natural language understanding that involves converting unstructured text into structured meaning representations. In the context of semantic search and information retrieval, it enables more precise query interpretation by extracting and formalizing key entities and relationships from user input. This process is particularly valuable in technical domains where queries often involve specific attributes such as materials, dimensions, or component types.

A foundational element of semantic parsing is Named Entity Recognition (NER), which identifies and classifies segments of text into predefined categories. In our application, NER is employed to detect structured elements relevant to engineering components, yielding tuples such as:

’material’ : ’aluminum’, ’part type’ : ’bolt’

This structured representation facilitates more accurate retrieval by aligning query semantics with indexed document content. For example, a query parsed into component categories and material types can be matched to domain-specific documents with similar structured annotations.

Incorporating semantic parsing into the retrieval pipeline strengthens the system’s ability to interpret user intent in a structured, machine-readable form, laying a foundation for more sophisticated matching and reasoning mechanisms.

3.2 Document Embedding

Document embedding techniques play a central role in semantic retrieval systems, providing a means to represent textual content in a high-dimensional continuous space. These embeddings facilitate comparison and ranking based on semantic similarity rather than purely lexical overlap.

3.2.1 Single Vector Embeddings

Traditionally, document retrieval pipelines have employed single-vector embeddings, wherein an entire document is compressed into a single fixed-length vector. While computationally efficient, this representation often sacrifices granularity and fails to capture localized semantic signals-particularly problematic in documents containing heterogeneous or multi-topic content.

3.2.2 Multivector embeddings

In contrast, multivector embeddings offer a more fine-grained alternative. Rather than encoding the document as a monolithic vector, the document is segmented into smaller units-such as sentences or clauses -each of which is embedded separately. This enables the retrieval system to assess the semantic relevance of sub-document components, allowing for more precise alignment with user queries. Emerging research has begun to explore this paradigm, often within hybrid retrieval frameworks that integrate sparse and dense indices. The use of multivector embeddings represents a natural evolution of this trend, offering a promising avenue for retrieval systems in technical and high-variance domains.

3.3 Document Retrieval

Document retrieval forms the backbone of semantic search systems, enabling the identification of relevant documents in response to user queries.

3.3.1 Retrieval in Vector Space Models

Retrieval of documents in a vector space model has long been discused and analyzed. it is simple, fast and efficient. Multiple distance metrics can be employed depending on the specific task at hand. Some of them are Cosine similarity or Pairwise distance to name a few. Dense retrieval is implemented using Word2Vec-based embeddings, which allow for capturing semantic similarity beyond surface-level term overlap.

3.3.2 BM25-Based Retrieval

The BM25 algorithm remains a foundational component of sparse information retrieval systems due to its simplicity, effectiveness, and interpretability. Rooted in probabilistic retrieval theory, BM25 ranks documents based on the frequency of query terms, adjusted for term saturation and document length. Despite the rise of neural and embedding-based models, BM25 continues to offer significant benefits, especially in domains where lexical precision and performance efficiency are paramount.

One of BM25's key strengths lies in its ability to reward exact term matches, which is particularly advantageous in technical corpora where specific terminology carries high informational value. Unlike semantic models that generalize across terms, BM25 ensures that documents containing the exact query terms are given precedence, thus preserving lexical fidelity.

Additionally, BM25 incorporates a document length normalization factor, which mitigates biases toward longer documents that may match more terms simply by virtue of size. This normalization contributes to more balanced and fair rankings, ensuring that concise but highly relevant documents are not unfairly penalized.

From a practical standpoint, BM25 is computationally lightweight and highly scalable, making it suitable for large-scale indexing and real-time retrieval scenarios. Its deterministic nature also facilitates debugging, transparency, and reproducibility-qualities often lacking in dense or deep learning-based approaches.

3.3.3 Hybrid retrieval

By integrating these two paradigms, we can aim to leverage the complementary advantages of semantic generalization (via dense vectors) and lexical specificity (via sparse ranking). Moreover, BM25 serves as a robust baseline in hybrid retrieval frameworks, complementing dense retrieval methods by ensuring high recall for exact matches. When paired with semantic reranking modules, BM25 helps to anchor the retrieval process with high-precision candidates, providing a reliable first-pass filter before more nuanced processing is applied.

These benefits make BM25 not only a viable fallback strategy but also a strategic asset in hybrid retrieval pipelines, particularly in domains where query specificity and resource constraints play a significant role.

4 Methodology

This section outlines the core components of the methodology, structured across three primary workflows: query processing, document indexing, and retrieval with re-ranking.

4.1 Query Processing Pipeline

The query preprocessing and representation pipeline involves the following key steps:

Tokenization and Stemming: The query is tokenized and stemmed to normalize lexical variations, preparing it for both sparse (e.g., TF-IDF) and dense (e.g., neural embeddings) representations.

TF-IDF Weighting: A term-frequency inverse document frequency scheme is applied to highlight discriminative terms based on their distribution across the corpus.

Query Expansion: Semantically related terms are introduced through lexical databases or embedding similarity, improving recall by capturing synonyms and contextually similar variants.

Semantic Parsing and Named Entity Recognition (NER): Structured entities such as material types, dimensions, or component categories are extracted (e.g., `"material": "aluminum"`, `"part type": "bolt"`), facilitating more targeted retrieval.

This comprehensive processing pipeline ensures that queries are both linguistically normalized and semantically enriched before being used in retrieval tasks.

4.2 Document Indexing Pipeline

We assume that all documents have already been preprocessed and structured in the appropriate format. For details regarding the preparatory transformations applied, refer to the *Dataset* section.

The document indexing pipeline begins with the application of standard natural language processing (NLP) techniques, including tokenization. Following this, we employ a multi-vector embedding strategy to represent the documents semantically.

Initially, each entire document is embedded using a Word2Vec model, with prior weighting via term frequency-inverse document frequency (TF-IDF) to emphasize the importance of informative terms. To enhance retrieval granularity, each document is further segmented into smaller subsections, and embeddings are generated for each segment. Depending on the document length, between two and five segment-level embeddings are created. This approach allows for a more nuanced semantic representation of the content, particularly useful for capturing detailed or context-specific information.

The complete set of embeddings-both the global (full-document) and local (segment-level)-are stored in parallel. This enables flexible and robust retrieval strategies by leveraging both holistic and fine-grained semantic cues. The multi-vector representation is especially beneficial in domains where documents contain dense and highly specialized technical content.

4.3 Retrieval and Re-ranking

The final stage involves a two-step retrieval strategy:

Initial retrieval is performed using a Vector Space Model (VSM), identifying candidate documents based on cosine similarity in the dense embedding space and Re-ranking of the top-K candidates is conducted using BM25, a sparse retrieval model that leverages exact term matching and document length normalization.

This hybrid approach balances the semantic generalization capabilities of dense embeddings with the lexical precision and computational efficiency of BM25. Additional re-ranking steps or GNN-based ranking models may be integrated in future iterations, as discussed in the Future Work section.

5 Dataset Construction

To address the scarcity of publicly accessible, text-centric industrial document datasets, we developed a custom Python-based web scraping pipeline. This pipeline processes specified DigiKey category URLs by downloading linked documents into a local cache and extracting any embedded PDF files. Local caching is a critical design choice to minimize redundant HTTP requests, thereby preventing server overload and mitigating the risk of IP-based rate limiting.

Following the download phase, a post-processing validation step is applied to ensure dataset integrity. This involves checking for and removing any corrupted or unreadable PDFs, thereby preserving the quality and usability of the final dataset.

5.1 Dataset Origin and Labeling

The dataset is constructed primarily from product documentation hosted on the DigiKey website. Each product category serves as a class label within the dataset. Although up to 400 PDFs can theoretically be retrieved per category URL, practical constraints, such as shared PDFs across multiple products or the use of non-embedded file links-reduce this number. On average, approximately 100 unique and valid PDFs were collected per label.

At present, the dataset contains 30 distinct labels, each with 100 validated PDF documents, totaling 3,000 PDFs. Descriptive statistics of the collected documents are as follows: the median PDF is 5 pages long and contains 1,371 words, 18 images, and 14 tables. These metrics highlight the richness and structural complexity of the dataset, making it well-suited for tasks involving information extraction, document classification and retrieval, or table/image recognition.

5.2 PDF Extraction Outline

The collected PDF documents were systematically converted into raw text files to facilitate downstream analysis. A critical objective of this process was to preserve the semantic content embedded in various formats-textual, tabular, and visual-while rendering it into a unified, text-based representation. To achieve this, the extraction workflow was divided into three distinct methods, each tailored to a specific type of content within the PDFs:

Text Extraction: All readable text elements, including paragraphs, headings, and annotations, were extracted and saved as plain text files. This step ensured that the primary narrative content of each document was preserved for further processing.

Table Extraction: Tabular data required a more nuanced approach due to its dependence on layout and context. To address this, table-to-text models were employed to convert tables into natural language descriptions, either summarizing entire tables or generating row-by-row textual explanations. This technique maintained the semantic integrity of the original tables.

Image-to-Text Extraction: For diagrams, charts, and other visual elements containing embedded text, image-to-text models were used to extract relevant textual information. These models enabled the integration of visual content into the text corpus, thus supporting a more comprehensive understanding of each document.

This multi-faceted extraction pipeline enabled a thorough and semantically rich transformation of PDF documents into a coherent text-based dataset, suitable for subsequent analysis and experimental workflows.

5.3 Text Extraction

To identify the most effective approach for extracting textual content from PDFs, several Python-based libraries were evaluated based on their performance across various document layouts and content types:

PyPDF2: Demonstrated limited capability, particularly when processing documents with complex formatting or mixed content. It often failed to preserve structural coherence.

PyTesseract: Although proficient in optical character recognition (OCR), this library struggled with tabular and columnar content, frequently compromising layout fidelity and data integrity.

PDFPlumber: Offered the most consistent results across diverse content types, including both textual and tabular data. Despite some minor inaccuracies, it showed superior adaptability and was well-suited for large-scale extraction tasks.

MuPDF: Delivered strong performance with documents containing multi-column text but lacked the flexibility required for handling mixed or irregular structures as effectively as PDFPlumber.

Based on this evaluation, PDFPlumber was selected as the primary tool for text and table extraction due to its balance of accuracy, robustness, and versatility. While minor extraction errors were observed, these were effectively corrected through downstream post-processing using natural language processing (NLP) techniques. This ensured that the resulting text data maintained high fidelity to the original documents and met the quality standards required for subsequent analysis.

5.4 Table Extraction and Processing

Table extraction represented one of the most complex challenges in the dataset construction pipeline. Although various open-source Python libraries offer functionality for table extraction, they often fall short in capturing all relevant information accurately. These limitations are compounded by the inconsistent formatting of tables within PDF documents, which vary significantly in dimensions, structure, and content presentation. Additionally, some non-tabular elements are occasionally misclassified as tables, further complicating the extraction process. Such inconsistencies hinder the performance of table-to-text language models (LLMs), which are typically trained on well-structured, standardized tabular data.

To address these challenges, a multi-step workflow was developed for reliable table extraction and processing:

1. **Table Extraction**: Tables were extracted using the `pdfplumber` library, which offers robust capabilities for detecting and parsing tabular content within PDF files.

2. **Preprocessing**: Extracted tables underwent a preprocessing phase to standardize the structure. This included removing extraneous columns, correcting formatting issues, handling null values, and aligning inconsistent rows. The objective was to ensure that the resulting tables were clean and suitable for automated analysis.

3. **Filtering**: A set of heuristic rules, adapted from the methodology presented in the TableGPT2 paper[1], was applied to filter out poorly structured or irrelevant tables. Filtering conditions included a minimum threshold for rows and columns, the

[1]These filtering criteria ensure that only meaningful and structured tables are retained for downstream processing.

presence of meaningful headers, and a low level of noise.

4. **Model Selection**: Based on the structural complexity and content density of each table, one of two models was selected for table-to-text conversion: `TableGPT2-7B` or `Qwen2.5-7B-Instruct`.

5. **Table-to-Text Conversion**: Each selected table was formatted into a standardized CSV-like structure and passed to the respective language model with the prompt: *"Describe the contents of each row in a technical manner."* The models generated descriptive, row-level textual explanations that retained the semantic content of the original table.

6. **Integration**: The generated descriptions were appended to the raw text previously extracted from the corresponding PDF, ensuring that tabular data was semantically integrated into the final corpus.

Model Assignment and Performance: `TableGPT2-7B` was utilized for larger, information-dense tables. It demonstrated strong performance in generating rich contextual summaries, particularly for technical tables containing multiple columns and detailed numerical or categorical data

`Qwen2.5-7B-Instruct` was employed for simpler tables, typically comprising one or two columns. These often resembled structured lists rather than formal tables. Although not explicitly trained for table-to-text tasks, the model performed adequately due to its robust general-purpose instruction-following capabilities.

```
UL Temperature:  75C,
Operating:  -20C to +75C
```

The above is an example of a list-like structure, processed using `Qwen2.5-7B-Instruct` rather than a dedicated table-processing model.

This dual-model strategy ensured a balanced approach to table processing, capturing the full range of table complexities present in the dataset while maintaining the integrity and usability of the extracted information.

Known Limitations

Several limitations were identified in the current extraction pipeline. One recurring issue is the duplication of content in the final text output. This typically arises when the same information is extracted both as plain text and as part of a table, resulting in redundant entries. Although efforts were made to resolve this overlap, a fully effective solution was not achieved. Nevertheless, this limitation is considered minor, as the duplicated content represents only a small fraction of the overall dataset and does not significantly impact downstream processing.

The most critical limitation stems from the table extraction stage. Since all subsequent steps rely on the accuracy of the initial extraction, any errors or inconsistencies introduced at this point propagate through the pipeline. Improving this foundational stage would have a significant impact on the overall quality of the dataset.

Potential avenues for addressing these limitations include the development of a proprietary, OCR-enhanced table extraction method or the integration of vision-enabled large language models (LLMs) capable of interpreting complex tabular and visual layouts more reliably. However, these enhancements fall outside the scope of this project and are proposed as directions for future work.

5.5 Image Extraction and Processing

Image Extraction Methods

To extract images embedded within PDF documents, three Python libraries were evaluated: `PyMuPDF` (`fitz`), `pdfplumber`, and `PyPDF`. Among these, `PyMuPDF` consistently yielded the most reliable results, successfully identifying and extracting the majority of embedded images. However, it occasionally fragmented larger images into smaller segments, which introduced minor inconsistencies.

Although `pdfplumber` lacks a dedicated image extraction function, it proved useful for identifying the bounding coordinates of image-like regions. These coordinates could be used to programmatically capture the visual content within those bounds, effectively mimicking screenshot-based extraction. `PyPDF`, by comparison, did not offer any significant advantages over `PyMuPDF` and was therefore not selected for further use.

Advanced visual extraction techniques involving image-based models were intentionally excluded from this pipeline. Such approaches are generally more complex, prone to inaccuracies, and risk interfering with the clarity and coherence of the textual content, an essential characteristic of the dataset.

Image Preprocessing

To ensure the quality and relevance of extracted images, a pre-filtering step was implemented based on two main criteria:

Pixel Count: Images with fewer than 1,000 total pixels (for example, a resolution of 100×100 or smaller) were considered low-quality artifacts or decorative elements such as logos, and were consequently discarded. Although a higher threshold might have further improved quality, it also posed the risk of excluding smaller but potentially relevant images. Therefore, a balanced threshold was selected to optimize both precision and recall.

Aspect Ratio: Images with an aspect ratio exceeding 5:1 or falling below 1:5 were also removed, as they typically represented non-informative elements like horizontal banners or vertical separators. Given the median image resolution of 300×270 pixels, this filtering strategy proved to be both effective and reliable.

Quality Assessment Using a Classifier

To further enhance the relevance and quality of the extracted images, a pretrained `MobileNet` classifier was employed. This classifier was tasked with identifying and discarding images that, although technically valid, were irrelevant to the dataset objectives, such as corporate logos, stock imagery, or decorative visuals unrelated to the technical documentation.

A preliminary classification model was developed using a curated dataset of 792 labeled images. Through data augmentation techniques, this dataset was expanded to 2,376 samples. It consisted of 445 images labeled as *Product* and 349 as *Not Product*. The initial model achieved a classification accuracy of 95.6 percent, demonstrating strong discriminative capabilities.

Following these results, a custom classifier was trained and scaled to process the full image corpus, consisting of approximately 50,000 entries. This model served as an additional filtering mechanism, substantially improving the overall quality and focus of the image dataset.

Image Processing

After irrelevant images were filtered out, a vision-language model was employed to generate technical descriptions of the remaining visual content. The `llama-3.1-8B-vision-378` model was selected for this task. Each image was paired with the following prompt:

"Describe the image in a technical manner." The resulting outputs were saved as individual text files and prepared for later integration with the rest of the extracted document content.

An alternative experimental setup was also explored to evaluate potential improvements in description quality. In this configuration, three vision-capable models, `LLaMA`, `LLaVA`, and `Florence`, were applied to the same set of images. Their generated outputs were concatenated with those from `LLaMA` to create multi-perspective descriptions. However, this approach did not yield a measurable improvement in the technical accuracy or informativeness of the results and was therefore not adopted in the final pipeline.

6 Future Work

We will outline here some of the stuff that we could have investigated but have no time -¿ future work then

6.1 Corpus-Guided Semantic Models

Although not implemented in the current work, an emerging and promising direction involves the use of corpus-steered query expansion powered by Large Language Models (LLMs). Rather than relying solely on general-purpose pretrained models, this approach advocates for fine-tuning or prompting LLMs using the linguistic and structural characteristics of the target corpus. This facilitates more accurate, context-sensitive query interpretations that are grounded in the data distribution of the specific domain. While our system does not incorporate this method, its potential merits suggest an intriguing avenue for future research.

6.2 Semantic Parsing

Although we initially considered implementing semantic parsing in the current project, constraints related to computational resources and time prevented us from pursuing this approach. The goal of semantic parsing is to extract relevant information from a user query in a structured format. By using a Named Entity Recognition (NER) tool, we could have conducted more precise and noise-free searches within the system. However, due to the domain-specific nature of our project, no pre-existing models were available to extract the required information. While training a custom model was a viable option, it necessitated the creation of our own labeled dataset. Given the aforementioned time constraints, we were unable to undertake this additional step.

6.3 Enhancing Document Ranking: VSM, GNN, and Hybrid Approaches

An important direction for future work involves a systematic evaluation and optimization of document ranking strategies within the retrieval pipeline. While the current system primarily relies on semantic matching for initial document retrieval, significant performance improvements may be achievable through more advanced re-ranking mechanisms. Three ranking strategies merit particular attention: traditional Vector Space Model (VSM)-based ranking, Graph Neural Network (GNN)-based re-ranking, and a hybrid framework that combines both methodologies.

6.4 VSM-Based Ranking.

The baseline approach uses cosine similarity to score the relevance between the query vector and the vector representations of documents. This method benefits from computational efficiency and a strong theoretical foundation in information retrieval. However, it inherently treats documents as isolated units in a high-dimensional space, without modeling inter-document dependencies or contextual semantics beyond individual term overlaps.

6.5 GNN-Based Re-ranking.

To capture higher-order relational information and contextual interactions among documents, a two-stage retrieval architecture can be employed. In the first stage, the top K candidate documents are retrieved using VSM similarity. In the second stage, these candidates are represented as nodes in a document graph, where edges denote semantic or structural relationships such as shared terminology, co-occurrence in domain-specific contexts, or hierarchical component-part associations.

A Graph Neural Network (GNN) is then applied to this subgraph to refine document embeddings or to generate context-aware relevance scores. This approach leverages graph-based inductive biases to model the interaction between documents, potentially improving the ranking of relevant but lexically dissimilar results.

6.6 Hybrid Approaches.

Combining the strengths of both VSM and GNN frameworks offers a promising hybrid strategy. One avenue involves weighted ensembling, where initial VSM scores are adjusted using GNN-derived relevance signals. Another involves iterative refinement, wherein GNN-informed rankings are used to update the query representation or guide a secondary retrieval phase.

Future research should investigate the comparative efficacy of these strategies using both intrinsic (e.g., ranking accuracy, nDCG) and extrinsic (e.g., task performance, user satisfaction) evaluation metrics. Further exploration into graph construction techniques, GNN architectures, and integration protocols is also essential for realizing the full potential of hybrid retrieval models.

6.7 Additional Proposals

Beyond core re-classification strategies, several additional proposals can enhance the semantic understanding and precision of the system's retrieval. These avenues involve improvements in ontology alignment, document representation, and advanced parsing techniques.

6.7.1 Leveraging ISO 10303/STEP Ontologies.

The ISO 10303 standard, commonly referred to as STEP, provides comprehensive ontologies for product and part modeling. While these structured classifications are valuable for representing mechanical components, their rigidity may limit their applicability in flexible or heterogeneous use cases. For instance, a component like a bolt may appear in various assembly contexts with different functional roles. Future work could explore selective or adaptive integration of STEP ontologies, potentially relaxing strict hierarchies in favor of more flexible semantic mappings.

6.7.2 Constructing Per-Document Graphs.

One promising direction involves parsing each document into a localized or micro-graph, where nodes represent entities such as materials, dimensions, and tolerances, and edges denote their interrelations. These per-document graphs can then be incrementally merged or aligned into a global knowledge graph, which serves as a substrate for GNN-based learning. This hierarchical graph construction strategy enables both document-level reasoning and corpus-wide generalization, offering a scalable method for semantic enrichment.

6.7.3 Advanced Document Parsing for Structured Layouts.

Technical documents, particularly PDFs, often include complex layouts with embedded tables, figures, and nested metadata. Traditional tools such as pdfplumber and PyMuPDF exhibit limitations in accurately capturing these structures. To overcome this, future iterations of the system may incorporate layout-aware deep learning models such as LayoutLM or Donut. These models integrate both visual and textual signals to extract structured data more effectively, thus improving downstream tasks such as entity recognition, relation extraction, and graph construction.

6.7.4 Transofrmer based embeddings – REWRITE

Due to the nature of our data (long documents), it was impossible to use different embedding models such as BERT (i mean, there are ways – link all relevant information). With a multivector approach, by cutting down pieces, we could embed said pieces into different vectors. This could also be interesting to explore.

7 References

References

[1] Bhatia S, Sharma P. Query expansion techniques: A survey. arXiv:1708.00247, 2017. https://arxiv.org/pdf/1708.00247, Aug. 2017

[2] Tang W, Jiang D, Zhang H. Corpus-steered query expansion with large language models. arXiv:2402.18031, 2023. https://arxiv.org/pdf/2402.18031, Feb. 2024

[3] Kaur P, Saini H. A review of ontology based query expansion. ScienceDirect, 2020. https://doi.org/10.1016/j.jksuci.2020.03.010, Mar. 2020

[4] Liu Y, Chen C, Li M. A survey of semantic parsing techniques. Symmetry, 2024;16(9):1201. https://www.mdpi.com/2073-8994/16/9/1201, Apr. 2024

[5] Chen J, Zhu X, Li Y, et al. GNNRank: Learning global rankings from pairwise comparisons via directed graph neural networks. arXiv:2202.00211, 2022. https://arxiv.org/pdf/2202.00211, Feb. 2022

[6] Zhang H, Wang K, Liu Y, et al. Semantic evidence-aware graph neural network (SE-GNN). arXiv:2109.11800, 2021. https://arxiv.org/pdf/2109.11800, Sep. 2021

[7] Guu K, Lee K, Tung Z, Pasupat P, Chang M. Open-domain question answering using early fusion of knowledge bases and text. arXiv:1809.00782, 2018. https://arxiv.org/pdf/1809.00782, Sep. 2018

[8] Zhou S, Wang S, Wang K, et al. AdapterGNN: Parameter-efficient fine-tuning improves generalization in GNNs. arXiv:2304.09595, 2023. https://arxiv.org/pdf/2304.09595, Apr. 2023

[9] Zhang Y, Hu B, Huang H. How can graph neural networks help document retrieval: A case study on CORD19 with concept map generation. arXiv:2201.04672, 2022. https://arxiv.org/pdf/2201.04672, Jan. 2022

[10] Park H, Kim J, Kim H. Hybrid recommendation system using graph neural network and BERT embeddings. arXiv:2310.04878, 2023. https://arxiv.org/abs/2310.04878, Oct. 2023