UNIVERSITY OF ANTWERP -
MASTER OF COMPUTER SCIENCE: DATA SCIENCE AND ARTIFICIAL INTELLIGENCE

An Intelligent Semantic Search System for Digital Product Part Design Catalogues in Manufacturing

Pablo de Vicente Abad

Modeling Intelligent Complex Software and Systems - MICCS-Lab
University of Antwerp, Belgium

Promotor: Prof. Moharram Challenger
Co-Promotor: Alireza Khalilipour
Date: 14/06/2025

Master Thesis

Abstract This paper introduces a modular framework for semantic similarity search over product catalogues, designed to robustly extract and structure content from PDFs, including text, images, and tables. The framework supports a range of retrieval strategies, spanning traditional methods like Vector Space Model (VSM) and BM25, fusion-based approaches such as Reciprocal Rank Fusion (RRF), and hybrid pipelines with reranking modules. It accommodates customizable configurations, including document representation through single-vector or multi-vector embeddings, as well as optional normalization and semantic query expansion. Through systematic evaluation, we observe that retrieval performance varies significantly depending on the dataset characteristics and the intended use case; whether prioritizing precision, recall, interpretability, or scalability. Rather than prescribing a one-size-fits-all solution, the framework is designed to support side-by-side comparisons of diverse strategies under a unified experimental setup. This allows practitioners and researchers to empirically determine which configurations are most effective for their specific needs. By offering flexibility in method selection and parameterization, the framework not only supports rigorous benchmarking but also enables iterative refinement and adaptation to evolving data and retrieval requirements.

Keywords document embeddings, hybrid retrieval, multi-vector encoding, semantic query expansion, vector space models.

Contents

List of Tables

List of Figures

Thesis Summary

This thesis addresses the challenge of building effective retrieval systems for multi-modal documents, such as product catalogs that contain a mix of text, tables, and images. Traditional search engines struggle to handle this variety of content in a unified way. To tackle this, the research introduces and evaluates two complementary pipelines —TTI.txt, which transforms multi-modal content into a unified plain-text format, and SSE, a modular framework for segment-based embedding and retrieval. Together, they enable flexible and scalable search across richly structured documents.

The study is guided by four main research questions: how to represent multi-modal content in a single semantic space; whether single-vector or multi-vector document embeddings perform better; which retrieval strategy (sparse, dense, or hybrid) is most effective; and how dataset characteristics influence the choice of strategy. To explore these questions, the thesis combines practical system design with empirical evaluation using a real-world product catalog dataset.

Methodologically, the TTI.txt pipeline linearizes documents into a stream of text that preserves the structure and semantics of tables and figures. SSE, on the other hand, indexes each document (with different and diverse approaches) and allows for both unified and granular embedding strategies. Retrieval is performed using sparse search (BM25), dense embedding-based search, or a hybrid of both, incorporating rank fusion and re-ranking steps.

Results show that the TTI.txt representation retains important semantic cues, improving retrieval effectiveness compared to plain-text baselines. Single-vector and multi-vector embeddings perform similarly under default chunking settings, though tuning segment size and weighting may offer additional benefits. Hybrid retrieval strategies outperform individual methods in terms of mean average precision (MAP), albeit at the cost of added complexity and latency. Finally, the study confirms that no single approach fits all scenarios: structured, terminology-heavy datasets favor sparse methods, while dense embeddings are better suited to free-form descriptions.

This thesis contributes not only a working toolkit and experimental benchmarks but also a practical guide for developing multi-modal retrieval systems. It emphasizes the importance of aligning system design with data characteristics and use-case requirements. A roadmap for future work is also provided, including dynamic segmentation, integration of multi-modal encoders, and automated benchmarking tools. Overall, the work advances the goal of building robust, adaptable, and domain-aware search systems for complex document collections.

1 Introduction

Despite the widespread availability of PDF-to-text converters, existing tools often overlook or mishandle crucial elements embedded in technical documents, such as tables, charts, and images that carry experimental data or structured measurements. As a result, downstream semantic-search applications may operate on incomplete representations, losing valuable context and reducing retrieval accuracy. To bridge this gap, we have developed **Tables-Text-Images.txt (TTI.txt)**, a pipeline that converts every component of a PDF (all textual and graphical elements) into a single, unified raw-text format. Building on this pipeline, we introduce **Semantic Search Exploration (SSE)**, a flexible evaluation framework designed to compare and refine document retrieval strategies.

TTI.txt delivers a complete, end-to-end conversion of every element in a technical PDF into plain text. Instead of extracting only body copy, it takes in captions, footnotes, tables and images, and outputs a single *.txt* file that contains all the information originally encoded in the document. Under the hood, TTI.txt reconstructs tables by generating line-by-line descriptions that capture row and column structure, cell values and labels. It likewise transforms images—whether charts, schematics or part diagrams—into structured text summaries that preserve axis labels, legends and annotations. All of these pieces are merged into one continuous text stream, eliminating gaps in the extracted data.

By preserving every detail in a flexible *.txt* format, TTI.txt lays the groundwork for downstream tasks—semantic search, data analysis or machine-learning pipelines—to operate on the document's complete informational content. No proprietary formats, no lost tables or figures, just raw text that fully represents the original PDF.

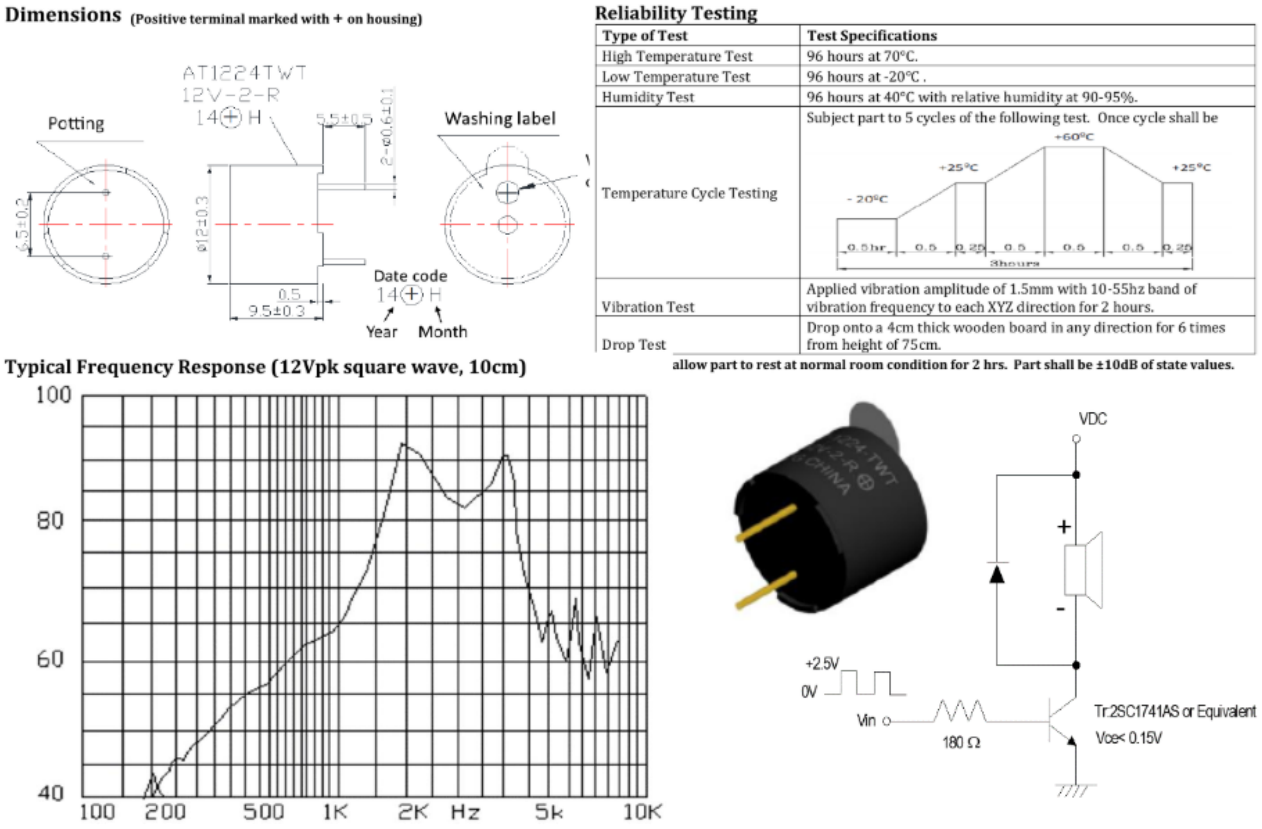
Once documents are fully converted, SSE serves as a research-grade evaluation toolkit, allowing users to systematically test and compare retrieval methods under a unified framework. Researchers can configure traditional lexical approaches such as BM25; dense-vector searches using pre-trained embeddings with both single-vector and multi-vector representations; hybrid strategies that apply score normalization to combine lexical and vector outputs; and ranking methods such as Reciprocal Rank Fusion (RRF). SSE further supports optional semantic expansion based on corpus analysis, enabling controlled studies of query expansion effects. To assess real-world performance, SSE offers cascaded re-ranking workflows, scalability features for growing document sets.

We build our system on top of product catalogues (PCs), a class of richly formatted technical documents that are extensively used in industrial settings. A product catalogue typically defines every item a manufacturer offers—ranging from simple fasteners (e.g., a stainless-steel bolt specified by material grade, thread size, tensile strength and corrosion resistance) to complex electromechanical assemblies (e.g., a loudspeaker driver characterized by impedance, power handling, frequency response and enclosure compatibility). PCs may take several forms, including

- **Printed PDF Catalogues**, prepared for distribution in manuals and brochures;
- **Interactive Web Catalogues**, where users navigate via hyperlinks or filters;
- **Embedded Database Catalogues**, which expose product tables and images through APIs or spreadsheets.

We focus on PDF-based PCs because of their sheer volume in engineering practice and the persistent challenge they pose for search and retrieval. By anchoring our corpus in these documents, we address a real-world niche: enabling engineers and procurement specialists to locate the exact PC —and therefore the precise part description— they need, without manual browsing. Our dataset builder accepts a URL to a web page or code repository, automatically harvests every linked PDF, verifies file integrity, and extracts both textual specifications and embedded assets (diagrams, tables, photographs).

Take, for example, the component labeled ”AT-1224-TWT-12V-2-R”. If you only saw its image—whether a photograph, schematic fragment, or data chart—you wouldn’t know it’s actually an alarm-buzzer-and-siren module. Only when you combine that visual with its accompanying text (operating voltage notes, mounting dimensions) and tables (performance curves under different frequencies) does its full function become clear. Figure 1 illustrates this point with four catalog excerpts, showing how part specifications, tabular data, frequency-response graphs, and full schematics all contribute essential context.



Reliability Testing

Type of Test	Test Specifications
High Temperature Test	96 hours at 70°C.
Low Temperature Test	96 hours at -20°C .
Humidity Test	96 hours at 40°C with relative humidity at 90-95%.
Temperature Cycle Testing	Subject part to 5 cycles of the following test. Once cycle shall be
Vibration Test	Applied vibration amplitude of 1.5mm with 10-55hz band of vibration frequency to each XYZ direction for 2 hours.
Drop Test	Drop onto a 4cm thick wooden board in any direction for 6 times from height of 75cm.

allow part to rest at normal room condition for 2 hrs. Part shall be ±10dB of state values.

Fig.1. Image extractions from Product Catalog

Central to our design is modularity and future-proofing. Each stage of the pipeline—document parsing, table and image transformation and search—is implemented as an independent component that can be swapped out as needed. If a particular parser, embedding model, or ranking algorithm becomes outdated or a superior alternative emerges, users can simply plug in the new module without altering the rest of the system. This plug-and-play architecture ensures that SSE evolves alongside advances in data processing and retrieval techniques, helping experimentation and minimizing maintenance overhead.

By combining comprehensive content extraction through TTI.txt with a configurable evaluation suite in SSE, our framework lowers the barrier to building, testing, and benchmarking document-centric search systems. Crucially, SSE acknowledges that each corpus is unique—part specifications, research papers, and legal reports demand tailored retrieval strategies based on their structure and end-user goals. Whether the target audience is engineers seeking technical measurements or analysts mining financial tables, SSE's modular design and broad retrieval options enable practitioners to evaluate and select the approaches best suited to their data and use case.

We anticipate that this toolkit will serve as a practical reference for anyone seeking to deploy robust semantic search solutions over complex PDF collections.

To guide the reader through this work, the remainder of the thesis is structured as follows. Chapter 2 offers a review of the literature, situating our study within existing research and pinpointing the gaps we address. Chapter 3 details our research methodology, from experimental design and analytical techniques employed in the SSE workflow. Chapter 4 details the data-collection and transformation procedures related to the TTI.txt tool.This Chapter also includes details about **Research Internship 2:** *Categorization of Textual Industrial (Technical) Reports*, conducted at the MICCS Lab, University of Antwerp—the same research group under which this thesis is presented. We outline the Experiment section in Chapter 5—detailing our methodology and its importance—and then, in Chapter 6, analyze the results in the context of existing research, highlighting their significance and broader implications. Finally, Chapter 7 concludes with a summary of our key contributions and avenues for future investigation are suggested in Chapter 8

2 Literature Review

The growing complexity of document retrieval pipelines has prompted extensive research into techniques that enhance query quality, optimize document representation, and improve ranking effectiveness. In the first strand of literature on query processing, scholars have investigated methods such as query expansion, term re-weighting, and semantic normalization to bridge the vocabulary gap between user queries and document content. Early work on pseudo-relevance feedback demonstrated that iteratively enriching queries with terms extracted from top-ranked documents can substantially improve retrieval performance [**Rocchio, 1971**, **Lavrenko and Croft, 2001**]. More recent studies leverage distributional word embeddings and contextual language models—such as Word2Vec and BERT—for semantic expansion, suggesting contextually relevant synonyms and related phrases to boost recall and precision [**Mikolov et al., 2013**, **Devlin et al., 2019**].

The second body of work addresses document indexing and representation. Traditional sparse, bag-of-words models like TF-IDF and BM25 represent documents as weighted term vectors and have formed the backbone of search systems for decades [**Järvelin and Kekäinen, 2002**, **Robertson and Zaragoza, 2009**]. Advances in neural language modeling introduced dense-vector embeddings, capturing semantic relationships in continuous space [**Le and Mikolov, 2014**]. Researchers have compared single-vector representations—which summarize each document with a single embedding—to multi-vector approaches that partition text into topical segments, encoding each separately for finer-grained matching [**Karpukhin et al., 2020**]. These studies highlight trade-offs between indexing efficiency, storage overhead, and retrieval accuracy, as well as the benefits of hybrid architectures that integrate sparse and dense representations [**Zhou et al, 2021**].

Finally, the retrieval and ranking literature examines how to select and combine these representations to produce optimal results. Sparse retrievers like BM25 excel on exact-match queries over frequent terms, while dense retrievers using cosine similarity in embedding spaces capture semantic similarity between diverse phrasings [**Gao et al., 2021**]. Hybrid methods—from simple linear score combinations [**Wang et al., 2021**] to rank-fusion algorithms such as Reciprocal Rank Fusion [**Cormack et al., 2009**]—seek to harness the strengths of both paradigms. Additionally, cascaded re-ranking frameworks employ lightweight first-pass retrieval followed by more computationally intensive neural re-rankers, balancing throughput and accuracy [**Nogueira and Cho, 2019**,**Cao et al., 2020**]. Collectively, these works underscore the need for flexible retrieval architectures that can adapt to the characteristics of specific corpora and user information needs.

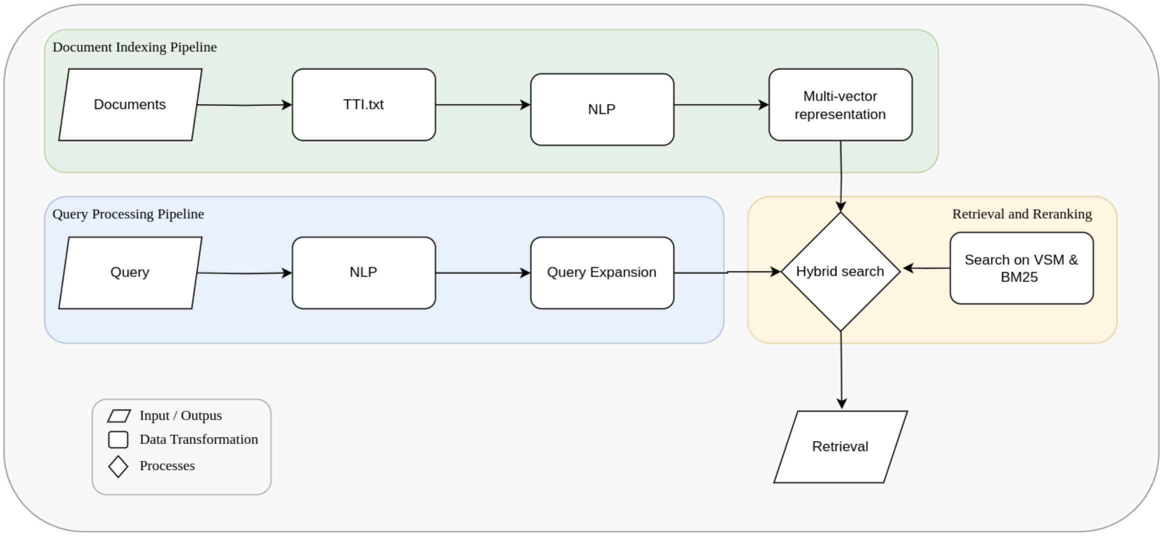


Fig.2. Semantic Search Evaluation (SSE) overview

2.1 Query Processing Pipeline - 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 [**Manning et al., 2008**, **Guo et al., 2016**]. Traditional semantic search systems often incorporate user-specific signals such as geographic location, browsing history, or long-term user behavior to personalize results [**Bennett et al., 2012**, **Yao et al., 2020**]. 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.

2.1.1 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 [**Voorhees, 1994**], 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 [**Carpineto and Romano, 2012**, **Xu and Croft, 2009**].

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” based on their syntagmatic relationships within the domain [**Fang et al., 2006**]. This corpus-aware expansion strategy increases retrieval precision by aligning query representation with the semantic structure of the indexed data.

2.1.2 Semantic Parsing

Semantic parsing is a critical technique in natural language understanding that involves converting unstructured text into structured meaning representations [**Liang, 2016**, **Zettlemoyer and Collins, 2005**]. 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 [**Zhong et al., 2020**].

A foundational element of semantic parsing is Named Entity Recognition (NER), which identifies and classifies segments of text into predefined categories [**Nadeau and Sekine, 2007**]. In our application, NER would be 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, laying a foundation for more sophisticated matching and reasoning mechanisms [**Dong et al., 2015**].

2.2 Document Indexing Pipeline

Document embedding techniques play a central role in semantic retrieval systems, providing a means to represent textual content in a high-dimensional continuous space [**Manning et al., 2008**, **Mikolov et al., 2013**]. These embeddings facilitate comparison and ranking based on semantic similarity rather than purely lexical overlap [**Devlin et al., 2019**, **Karpukhin et al., 2020**].

2.2.1 Single Vector Embeddings

Traditionally, document retrieval pipelines have employed single-vector embeddings, wherein an entire document is compressed into a single fixed-length vector [**Le and Mikolov, 2014**, **Guo et al., 2016**]. 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 [**Fang et al., 2006**].

2.2.2 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 [**Nogueira and Cho, 2019**, **Cao et al., 2020**]. Emerging research has begun to explore this paradigm, often within hybrid retrieval frameworks that integrate sparse and dense indices [**Karpukhin et al., 2020**]. 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 [**Zhong et al., 2020**, **Liang, 2016**].

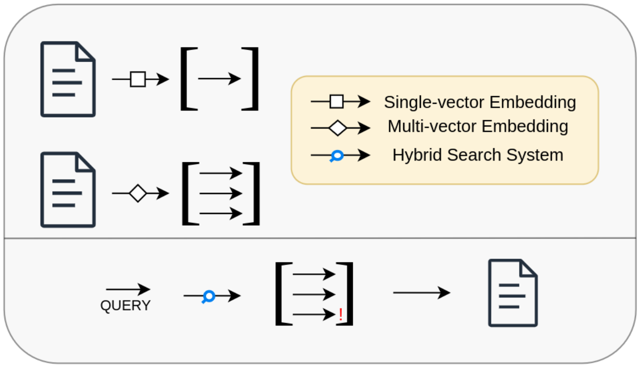


Fig.3. Single vs Multi vector embedding

2.3 Document Retrieval

At the heart of any semantic search system lies document retrieval, the process by which relevant documents are identified and ranked in response to a user's query. By projecting both queries and documents into comparable representations—whether sparse term vectors or dense embeddings—retrieval methods enable rapid filtering from large corpora, serving as the critical first step before any further re-ranking or fusion is applied.

2.3.1 Retrieval in Vector Space Models

Retrieval of documents in a vector space model has long been discussed and analyzed [**Salton et al., 1975**]. It is simple, fast and efficient. Multiple distance metrics can be employed depending on the specific task at hand—cosine similarity or pairwise distance to name a few [**Singhal, 2001**]. Dense retrieval is implemented using Word2Vec-based embeddings, which allow for capturing semantic similarity beyond surface-level term overlap [**Mikolov et al., 2013**].

2.3.2 BM25-Based Retrieval

The BM25 algorithm remains a foundational component of sparse information retrieval systems due to its simplicity, effectiveness, and interpretability [**Robertson and Zaragoza, 2009**]. Rooted in probabilistic retrieval theory, BM25 ranks documents based on the frequency of query terms, adjusted for term saturation and document length [**Robertson and Walker, 1994**]. 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 [**Lin, 2021**].

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 [**Guo et al., 2016**]. 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 [**Robertson and Zaragoza, 2009**].

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 [**Lin, 2021**].

2.3.3 Hybrid Retrieval

Hybrid retrieval combines the lexical precision of BM25 with the semantic generalization of dense embeddings. Formally, given a query q and document d, we compute

$$\mathrm{score}_{\mathrm{hybrid}}(q,d) = \lambda\,\mathrm{score}_{\mathrm{BM25}}(q,d) + (1-\lambda)\,\mathrm{sim}_{\mathrm{dense}}(q,d),$$

where $\lambda \in [0,1]$ balances the two components. BM25(q,d) is defined as

$$\sum_{t\in q} \frac{f(t,d)\,(k_1+1)}{f(t,d)+k_1\big(1-b+b\,|d|/L\big)} \cdot \log\frac{N-n_t+0.5}{n_t+0.5},$$

and $\mathrm{sim}_{\mathrm{dense}}(q,d) = \cos(\mathbf{v}_q, \mathbf{v}_d)$ uses pre-trained embeddings (single- or multi-vector) [**Robertson and Zaragoza, 2009**, **Mikolov et al., 2013**]. Tuning λ and normalizing both scores (e.g. min-max) is critical for stable performance across domains [**Cormack et al., 2009**].

2.3.4 Reciprocal Rank Fusion

Rather than combine raw scores, Reciprocal Rank Fusion (RRF) merges rank positions from two (or more) systems. For each document d, if its rank in system i is $r_i(d)$, RRF assigns

$$\mathrm{score}_{\mathrm{RRF}}(d) = \sum_i \frac{1}{k + r_i(d)},$$

where k (often set to 60) dampens the influence of top-ranked lists [**Cormack et al., 2009**]. Because it operates on ranks, RRF is robust to score-scale differences and consistently boosts documents that appear near the top in multiple runs.

2.3.5 Re-ranking Strategies

In a two-stage pipeline, a fast first pass (e.g. BM25 or RRF) retrieves the top-K candidates. These are then re-ranked by a more expensive model—often a transformer fine-tuned for relevance classification. Concretely:

$$\{d_1, \ldots, d_K\} = \mathrm{TopK}\big(\mathrm{score}_{\mathrm{first}}(q, D)\big),$$

then each d_j is assigned

$$\mathrm{score}_{\mathrm{re}}(q, d_j) = \mathrm{MLP}\big([\mathbf{v}_q; \mathbf{v}_{d_j}]\big)$$

or via pairwise cross-encoders [**Nogueira and Cho, 2019**, **Cao et al., 2020**].

Table 1. Comparison of Retrieval Methods

Method	Lexical Fidelity	Semantic Matching	Computational Cost
Dense Embeddings	Low	High	Medium
BM25	High	Low	Low
Hybrid	Medium	Medium	Medium
RRF	Varies	Varies	Low
Re-ranking	Medium	High	High

3 Methodology

Having surveyed the key concepts in query processing, document representation, and retrieval & re-ranking, we now describe how these pieces fit together in our Semantic Search Exploration (SSE) framework. Figure 2 illustrates the end-to-end architecture, which we divide into three interconnected workflows:

1. **Query Processing Pipeline**, where raw user inputs are normalized and intelligently expanded to maximize alignment with the indexed corpus.

2. **Document Indexing Pipeline**, which transforms each PDF—via our TTI.txt converter—into a unified raw-text stream, generates both single-vector and multi-vector embeddings, and stores them in fast lookup indices.

3. **Retrieval and Re-ranking**, where candidate documents are first retrieved by sparse (BM25) and/or dense (cosine-based) methods, then fused or re-ranked through hybrid strategies (score combination, RRF) and neural re-rankers to produce a final, high-precision ranked list.

In the subsections that follow, we walk through each of these stages in detail—describing the algorithms, configurable parameters, and data flows—while demonstrating how our implementation reflects the literature surveyed above. Together, these components realize an end-to-end toolchain for rapid experimentation, comparative evaluation, and eventual deployment on diverse technical PDF collections.

3.1 Query Processing Pipeline

Because user queries are typically brief and may not match document wording exactly, our pipeline first *lexically normalizes* and then *semantically enriches* each query to maximize retrieval effectiveness.

Lexical Normalization. We begin by tokenizing the raw input into words and punctuation, then apply stemming to reduce each token to its root form. This step reduces surface-level variation (e.g. 'measuring', 'measured') and ensures consistency across downstream models.

Term Weighting & Expansion. Next, we compute TF-IDF weights for each normalized token, emphasizing terms that are rare in the corpus but frequent in the query. To broaden recall, we perform query expansion via corpus-centric *Intelligent Substitution*, which extracts high-cooccurrence terms directly from our specialized dataset [**Carpineto and Romano, 2012**]. By augmenting the query with these semantically related terms, we capture synonyms and measurement variants (e.g. '3 mm' == '0.118 inch') that may otherwise be missed. These processed and expanded queries then feed into our document indexing pipeline (Section 2.2), where both sparse and dense representations are constructed for use in retrieval.

3.2 Document Indexing Pipeline

Starting from the unified raw-text streams produced by TTI.txt—which merge body text, tables, and image annotations into a single '.txt' file—we apply standard NLP preprocessing (tokenization, stop-word removal) before embedding. Further details about TTI.txt can be found in the *Dataset* section

Global (Single-Vector) Embeddings. Each entire document is first converted into a single fixed-length vector. We compute TF-IDF weights to emphasize domain-specific terms, then use a pre-trained Word2Vec model to produce $\mathbf{v}_{\text{doc}}$. This representation is compact and efficient, serving as a strong baseline for semantic retrieval.

Segmented (Multi-Vector) Embeddings. To capture fine-grained semantics in technical content, we partition each document into n segments (typically $n = 2$–5, based on page or section breaks) and embed each segment independently, yielding vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. This multi-vector approach allows retrieval algorithms to match queries against localized parts of a document, improving precision for mixed-topic or densely formatted materials.

As discussed in Section 2.2 , multi-vector embeddings capture localized semantic signals that single-vector methods can miss—particularly important for technical documents with mixed content.

3.3 Retrieval and Re-ranking

Several retrieval strategies were explored to improve the accuracy and robustness of the system. The initial approach used a simple cosine similarity search within a Vector Space Model (VSM), leveraging dense embeddings to capture semantic relationships between queries and documents. In parallel, BM25 was tested as a sparse retrieval method that relies on exact term matches and accounts for term frequency and document length.

A key part of this work involved experimenting with hybrid retrieval methods that combine the strengths of both VSM and BM25. In one setup, the scores from both retrieval models were combined to create a more balanced ranking that benefits from both semantic and lexical features. Another method, Reciprocal Rank Fusion (RRF), was used to merge the rankings produced by each model, prioritizing documents that consistently appear near the top across different ranking lists.

In a final variant, a two-stage approach was tested where candidate documents were first retrieved using BM25 and then re-ranked based on their similarity in the VSM embedding space. This strategy aims to refine the initial retrieval with a deeper semantic understanding.

Overall, these retrieval and re-ranking combinations aim to enhance performance by leveraging the complementary strengths of sparse and dense models. Future improvements may include more advanced ranking techniques, such as graph-based neural networks, which are outlined in the *Future Work* section.

4 Dataset Construction

This dataset was built as part of the course **Research Internship 2:** *Categorization of Textual Industrial (Technical) Reports*, conducted at the MICCS Lab, University of Antwerp—the same research group under which this thesis is presented.

During the internship, we investigated methods for categorizing technical textual documents from industrial contexts, with a particular focus on product catalogs. As a foundational step, we constructed a custom dataset, which also serves as the basis for this thesis project. In the following sections, we detail the methodology used for building this dataset, including key design decisions, processing steps, and the classification results obtained during the internship. While the classifier developed during that period is not directly used in this thesis, the insights gained from that task informed several of the design choices and conceptual modules applied throughout the project.

Note: The earlier course, **Research Internship 1**: *Immunosequencing Signatures of CMV Exposure and HLA Influence on TCR Diversity*, was unrelated to this thesis and was carried out under the Department of Computer Science.

4.1 Dataset Overview

Our dataset comprises text-centric product documentation sourced from the DigiKey platform, organized by product category as class labels. After an initial collection and validation process, we assembled roughly 100 unique, high-quality PDFs per category, yielding a diverse corpus of technical reports. Each document averages five pages in length and contains over a thousand words alongside multiple tables and figures, reflecting the structural complexity typical of industrial catalogs. In the sections that follow, we describe how these files are prepared for downstream tasks, how categories are assigned, and the key statistics that characterize the dataset's richness and suitability for information extraction and retrieval experiments.

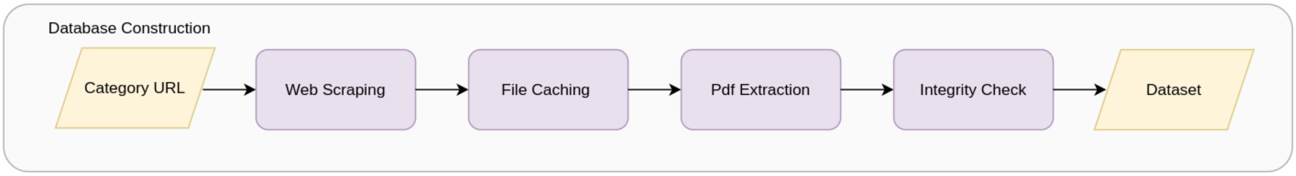


Fig.4. Database construction pipeline

4.2 Dataset Origin and Labeling

All documents were sourced from DigiKey's publicly accessible product pages, where each category URL, for example, *https://www.digikey.com/en/products/filter/printers-label-makers/887*, returns a paginated list of printer products. Rather than relying on browser automation, we exploited the site's URL-based paging: the first 100 items are shown by default, and successive batches are accessed simply by appending query parameters to the base URL, such as:

```
?s=N4IgrCBcoA5QjAGhDOl4AYMF9tA
?s=N4IgrCBcoA5QTAGhDOkCMAGTBfHQ
```

This approach allowed us to programmatically traverse all pages in each category without the need for interactive tools. Each retrieved PDF was automatically assigned the class label corresponding to its originating category URL. To maintain label purity, any PDF found under multiple categories was excluded, ensuring that each file has a single, unambiguous label.

During the scraping process, some documents linked to external hosts or non-PDF assets; these were excluded to maintain a uniform file format. We also implemented a checksum-based duplicate detection step, eliminating redundant downloads and ensuring that each PDF in the final corpus is unique. This approach preserves the integrity of the labeling scheme and prevents skewed class distributions caused by repeated documents.

Label assignments were then validated through a small-scale manual review. A random sample of 10% of the dataset was inspected to confirm that category labels accurately reflected document content. Any misclassified files discovered in this audit were either corrected or removed, resulting in a clean, reliable dataset ready for downstream classification and retrieval experiments. Some example categories are:

- accessories
- ac-power-connectors
- alarms-buzzers-and-sirens
- aluminum-electrolytic-capacitors
- aluminum-polymer-capacitors
- anti-static-esd-bags-materials
- backplane-connector-housings
- batteries-non-rechargeable-primary
- battery-chargers
- cable-ties-and-zip-ties
- coaxial-cables-rf
- microphones
- multiple-conductor-cables
- printers-label-makers

4.3 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. This is what we refer to as the **Table-Text-Images.txt** or *TTI.txt* pipeline. To achieve this, the extraction workflow was divided into three distinct methods, each tailored to a specific type of content within the PDFs:

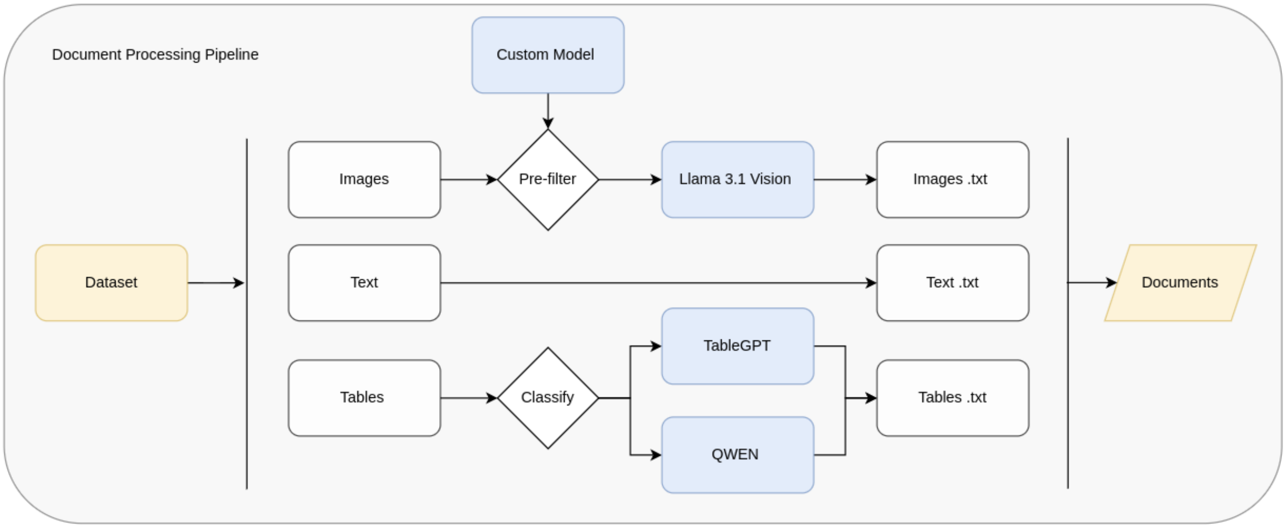


Fig.5. Workflow for converting PDFs to raw text files.

We extracted each PDF's content via three parallel pipelines (see Figure 5):

1. **Text extraction:** Extracting all readable paragraphs, headings, and annotations into plain-text files.

2. **Table-to-text extraction:** Converting tables into natural-language form —either as full-table summaries or row-by-row descriptions— using table-to-text models.

3. **Image-to-text extraction:** Using OCR and vision-language models on diagrams, charts, and figures to extract embedded text and descriptive information.

This comprehensive extraction pipeline facilitated a detailed and semantically meaningful conversion of PDF documents into a structured, text-based dataset, well-suited for further analysis and experimental workflows. The following sections will provide an in-depth explanation of each component of the workflow.

4.4 Text Extraction

To determine the most effective method for extracting raw text from our PDF corpus—including plain text, embedded tables, and multi-column layouts—we conducted a head-to-head evaluation of four widely used Python libraries:

1. **PyPDF2** reliably handles basic text but struggled with complex page structures. In documents featuring tables or side-by-side columns, PyPDF2 frequently misaligned the reading order and omitted key segments, making it unsuitable for our needs.

2. **PyTesseract**, an OCR-based approach, was able to recognize characters even in scanned or low-quality pages. However, an increase of misaligned columns and broken table cells was observed. Given that extensive manual correction was needed, we discarded this option.

3. **PDFPlumber** struck the best balance between fidelity and ease of use. It accurately captured text runs, respected line breaks, and even identified table boundaries. Minor errors—such as occasional mis-tagged bullet points or stray punctuation—were easily corrected in downstream processing.

4. **MuPDF** (via the `fitz` interface) showed strong performance on multi-column text extraction, preserving column order more consistently than most tools. Nevertheless, it lacked specialized routines for tables, resulting in flattened or concatenated rows that reduced semantic clarity.

After careful comparison on a representative subset of 50 documents spanning various formats (single-column reports, table-heavy datasheets, and mixed-layout manuals), PDFPlumber emerged as our tool of choice. Its robust handling of both narrative and structured content meant we could capture nearly all words present in the original files—an essential requirement for our Bag-of-Words (BoW) indexing strategy. Because BoW models care only that each token appears at least once, slight disruptions in sentence boundaries or formatting have minimal impact on retrieval performance.

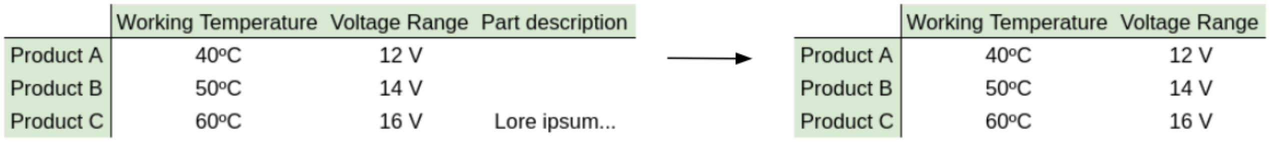
To validate this selection, we manually reviewed a random sample of extractions. The results were consistent: over 95% of words were correctly identified across different font sizes and column configurations, and less than 5% of tokens were extraneous artifacts (e.g., orphaned symbols or duplicate bullets). These residual issues are systematically eliminated during our NLP preprocessing pipeline—where custom regex filters and token-level cleaning remove non-informative characters—ensuring a clean, comprehensive vocabulary for all subsequent indexing and analysis tasks.

4.5 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**: The extraction of tables was performed using the `pdfplumber` library, which provides reliable functionality for identifying and parsing tabular data within PDF documents.

2. **Preprocessing**: The extracted tables were then subjected to a preprocessing stage aimed at standardizing their structure. This involved the removal of irrelevant columns, correction of formatting anomalies, treatment of missing values, and alignment of inconsistent rows. The goal was to produce clean, well-structured tables suitable for automated analytical tasks.



	Working Temperature	Voltage Range	Part description
Product A	40ºC	12 V	
Product B	50ºC	14 V	
Product C	60ºC	16 V	Lore ipsum...

	Working Temperature	Voltage Range
Product A	40ºC	12 V
Product B	50ºC	14 V
Product C	60ºC	16 V

Fig.6. Example of table cleaning: a column with over 60% missing values is removed to maintain consistency.

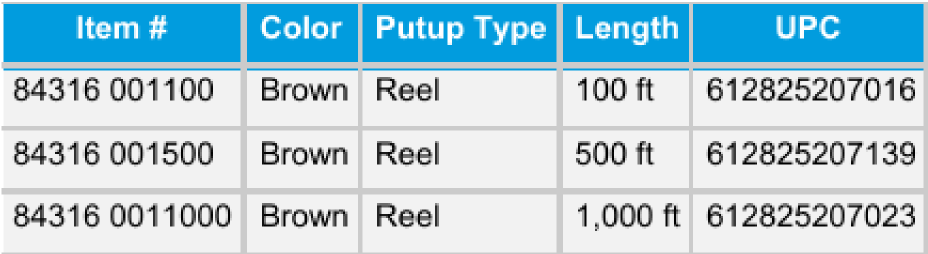
Upon completing these steps, we obtain semi-structured tables formatted in a manner compatible with most table-to-text generation workflows. From this stage forward, standard models—whether designed to generate comprehensive summaries of entire tables or row-level descriptions—can be readily applied without the need for additional layout reconstruction, as the core structural extraction has already been effectively handled.

Working with tables in PDF documents is inherently challenging due to their highly variable layouts, inconsistent formatting, and lack of standardized markup. This unstructured nature means that even small variations in cell alignment, merged headers, or nested tables can break automated parsers, making the extraction and cleaning process a time-intensive and error-prone task. Moreover, this step is critical: if a table extraction misses 50% of a table's content, downstream tasks such as classification, retrieval, or summarization will suffer from incomplete data, and there is no way to recover the lost information once this preprocessing stage is complete.

3. **Table Filtering and Model Assignment**. Once tables are cleaned and normalized, we apply a set of heuristic rules—adapted from the `TableGPT2` paper [**Su et al., 2024**] —to filter out any that are too sparse or irregular for our main table-to-text model. Specifically, we discard tables if they:

   - Have fewer than five rows or fewer than two columns.
   - Contain more than 30% missing (NaN) values.
   - Include any column whose first entry exactly duplicates its header.

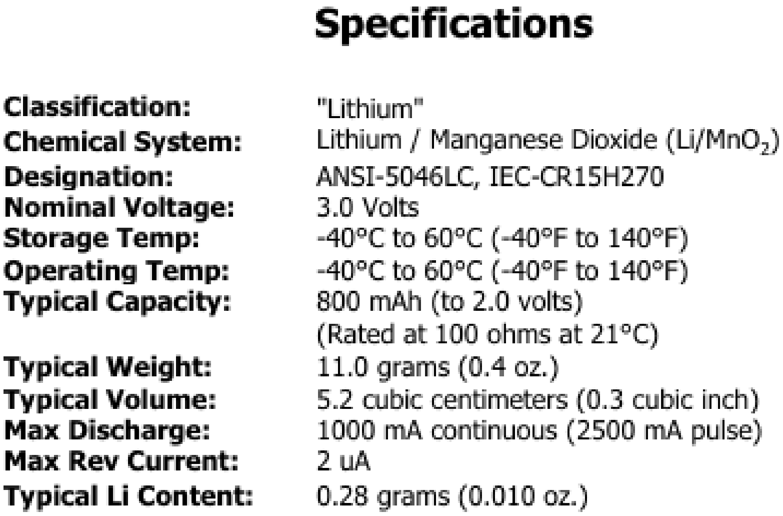
   Tables that pass these checks are designated structured tables: large, grid-like layouts analogous to spreadsheet data (see Figure 7). These structured tables are then processed with `TableGPT2`, which excels at generating detailed natural-language summaries on a row-by-row or whole-table basis.



Item #	Color	Putup Type	Length	UPC
84316 001100	Brown	Reel	100 ft	612825207016
84316 001500	Brown	Reel	500 ft	612825207139
84316 0011000	Brown	Reel	1,000 ft	612825207023

Fig.7. Class 1 table example

   Tables that fail one or more of the filters tend to be small, two-column *questionnaire* style layouts (see Figure 8). Because `TableGPT2` is not optimized for such sparse or semantically limited tables, we instead apply a simpler, template-driven extraction method to capture their concise content without introducing downstream errors, `Qwen2.5-7B-Instruct`.



Specifications

Classification:	"Lithium"
Chemical System:	Lithium / Manganese Dioxide (Li/MnO$_2$)
Designation:	ANSI-5046LC, IEC-CR15H270
Nominal Voltage:	3.0 Volts
Storage Temp:	-40°C to 60°C (-40°F to 140°F)
Operating Temp:	-40°C to 60°C (-40°F to 140°F)
Typical Capacity:	800 mAh (to 2.0 volts) (Rated at 100 ohms at 21°C)
Typical Weight:	11.0 grams (0.4 oz.)
Typical Volume:	5.2 cubic centimeters (0.3 cubic inch)
Max Discharge:	1000 mA continuous (2500 mA pulse)
Max Rev Current:	2 uA
Typical Li Content:	0.28 grams (0.010 oz.)

Fig.8. Class 2 table example

By categorizing each table to the model best suited for its size and structure, we maximize both accuracy and efficiency in our table-to-text conversions. Specifically, we process large, grid-style ”class 1” tables with **TableGPT2-7B**, which is optimized for detailed, row-by-row summarization. Smaller, two-column ”class 2” tables are handled by a more general-purpose instruction-tuned model, **Qwen2.5-7B-Instruct**, which can flexibly describe compact layouts with ease. In a later section, we will discuss these models in greater detail, including our criteria for selecting them and an analysis of their comparative performance.

4. **Table-to-Text Conversion**: Each selected table was 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.

5. **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.

4.5.1 Model Assignment and Performance

As allready mentioned, for converting tables to natural language, we selected two specialized models: **TableGPT2-7B** and **Qwen2.5-7B-Instruct**. `TableGPT2-7B` was chosen for its pretraining on tabular data and proven ability to generate accurate, row-wise descriptions, while `Qwen2.5-7B-Instruct` serves as a more general-purpose fallback for smaller or irregular tables.

Identifying a high-performing table-to-text model proved challenging, since most prior work has focused on table-based question answering rather than open-ended explanation. For example, TaPas excels at answering queries over tables but does not natively produce descriptive summaries [**Herzig et al., 2020**], and TabFact targets fact verification rather than narrative generation [**Yin et al., 2020**]. We empirically evaluated several off-the-shelf LLMs—including Llama-3.1, Llama-3.2, Florence, Bloom (pretrained on ToTTo dataset), and ChatGPT—but found that without table-specific pretraining they frequently hallucinated when asked to describe technical tables. These results align with observations in the literature that domain-mismatched LLMs often struggle to maintain fidelity when handling structured data [**Zhang et al., 2022**] [**Scao et al., 2022**].

Item #	**Color**	**Putup Type**	**Length**	**UPC/EAN**	**C**
5300UE 0101000	Black	Reel	1,000 ft	612825159162	C
5300UE 010U1000	Black	UnReel	1,000 ft	612825421511	
5300UE 010Z1000	Black	ReelTuff Box	1,000 ft	612825159254	

Table 2. Table example for our model comparison

4.5.2 Model Comparison Benchmark

The following section presents a detailed evaluation of performance benchmark for the model **TableGPT2-7B**, alongside a discussion of the criteria underlying their final selection. To identify the most suitable model for the table-to-text generation task, we assessed several off-the-shelf large language models (LLMs)—including **Llama-3.1**, **Llama-3.2**, **Bloom**, the free-tier version of **ChatGPT**, and our target model, **TableGPT2-7B**. These models were compared based on their ability to accurately summarize technical tables while minimizing hallucinations and preserving factual consistency. We provide an example in Table 2, and we will give output examples for each of the models bellow.

Bloom Bloom is a large multilingual model pretrained on the ToTTo dataset [**Parikh et al., 2020**]. In our tests, it completely misinterpreted the table format and defaulted to an awards-and-nominations schema:

> <section_title> Awards and nominations </section_title> <table> ... Academy Award, Best Actor, Peter Finch ..."

Again, the model's domain-mismatched pretraining caused it to disregard the actual product data.

Llama-3.1 Llama-3.1 is a general-purpose generative model optimized for broad natural-language tasks. However, when prompted to describe a product specification table, it produced entirely fabricated clinical text, demonstrating a clear mismatch with structured data inputs. For example, given a polyethylene-coated cable table, Llama-3.1 generated:

> Five minutes ago, a 32-year-old male patient presented to the emergency room with complaints of severe abdominal pain and nausea. He had a history of alcoholism and was taking 4 pills of alcohol a day...

This output is clearly irrelevant to the table's content, indicating severe hallucination.

Llama-3.2 Llama-3.2, a successor to Llama-3.1 with enhanced reasoning capabilities, also failed to produce coherent technical summaries. Its output drifted into boilerplate material safety descriptions and EU compliance directives, such as:

> The flammability rating of this product is 1... The European Union (EU) has set a number of environmental requirements for products, which include the requirement to be RoHS-compliant and WEEE-compliant...

This again confirms that general LLMs lack the specialized grounding needed for tabular engineering data.

ChatGPT (Free) In contrast, ChatGPT's free version—though not open-source—provided concise, accurate summaries when given the same cable specification table. Despite its strong performance, reliance on a proprietary API posed cost and integration concerns for large-scale batch processing.

> ”Item #: 5300UE 0101000, Color: Black, Putup Type: Reel, Length: 1,000 ft, UPC/EAN: 612825159162, Footnote: C.
> This product... is a high-quality black cable supplied in a reel putup type... Notably, this item also includes Footnote C, which may indicate specific handling or performance characteristics.”

TableGPT2-7B Finally, TableGPT2-7B—a model pretrained specifically on tabular data—delivered the most reliable results (from the open sourced models). When applied to a diverse set of cable and component tables, it extracted and aggregated key attributes with high fidelity. For example, it produced a structured overview listing available colors, putup types, lengths, and UPC/EAN codes for each item, closely matching the source table without hallucinations:

> General Overview: Colors Available: Black, Light Blue, Chrome, Gray, Orange, Red, Violet, White, and Yellow. Putup Types: Reel, UnReel, and ReelTuff Box. Lengths: Primarily 1,000 feet or 500 feet... Item Breakdown: Black Items Item #5300FE 0101000: Black, 1,000 ft, Reel, UPC/EAN 612825158981...

Based on these experiments, we concluded that general LLMs are prone to misleading hallucinations when confronted with technical tables. ChatGPT offers strong performance but at the expense of API constraints. Consequently, we selected TableGPT2-7B for large, grid-style tables (class 1) and Qwen2.5-7B-Instruct for smaller, two-column tables (class 2), as described in the previous section.

4.5.3 Qwen2.5-7B-Instruct

In contrast, **Qwen2.5-7B** was easier to select, as its research literature demonstrates strong performance. When employed on simpler tables—typically comprising one or two columns that often resemble structured lists rather than formal tables—the model performed adequately. Although not explicitly trained for table-to-text tasks, it delivered reasonable results due to its robust general-purpose instruction-following capabilities.

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

> This product is designed to operate reliably across a wide temperature range. It meets the UL Temperature standard, rated for up to 75 C. The operating temperature range extends from - 20 C to 75 C, ensuring performance in both moderately cold and high-temperature environments. These specifications make it suitable for applications where temperature fluctuations are common, providing durability and stability in diverse conditions.

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.

4.5.4 ***Known Limitations***

Despite the overall effectiveness of our extraction pipeline, several limitations were observed during implementation and evaluation:

Content Duplication In some cases, the same information appears both as plain text and within tables, resulting in redundant entries in the final output. Although we implemented deduplication strategies to address this issue, they are not fully reliable. Nevertheless, duplicated content represents only a minor portion of the dataset and has a negligible impact on downstream tasks.

Non-English Characters A limited number of documents include Chinese or other non-English characters that the pipeline does not process accurately. These characters are removed during post-processing, which can lead to the loss of non-English content.

Error Propagation The accuracy of all subsequent processing steps depends heavily on the quality of the initial text and table extraction. Errors at this stage—such as missing data, column misalignment, or optical character recognition (OCR) artifacts—can propagate throughout the pipeline, potentially affecting the performance of classification and retrieval tasks.

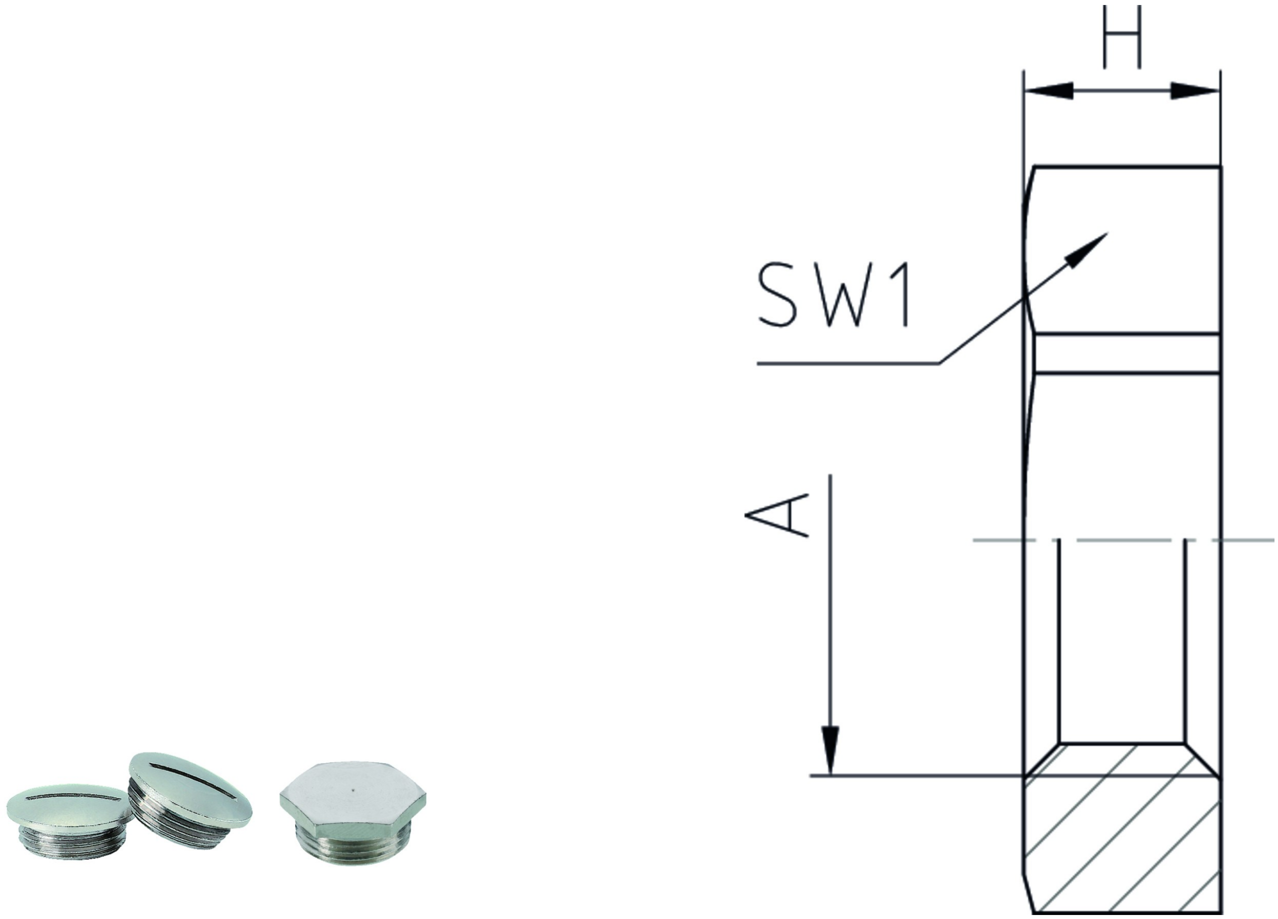
Computational Constraints Table-to-text conversion, particularly when using the **TableGPT2-7B** model, requires significant computational resources. On a local GTX 1050 Ti GPU, processing 8 to 10 PDFs took more than 50 hours. Due to hardware limitations, we developed a workflow using Google Colab Pro, which improved performance but did not fully resolve the resource bottleneck.

Future Work To address these limitations, future work may focus on: (1) designing a custom OCR-enhanced extractor optimized for table structures in technical documents, and (2) incorporating vision-capable language models that can jointly analyze textual content, tables, and figures. While promising, these improvements are beyond the scope of the study and are suggested for future development.

4.6 Image Extraction and Processing

4.6.1 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.



(a) Image extraction example of a product

(b) Image extraction example of a part diagram

Fig.9. Side-by-side examples of image extraction

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.

4.6.2 Image Preprocessing

To improve the quality and relevance of extracted images, a pre-filtering step was applied using two main criteria: pixel count and aspect ratio. Images with fewer than 1,000 total pixels, such as those with a resolution of 100×100 or smaller, were classified as low-quality artifacts or decorative elements, such as logos, and were removed. While using a higher threshold might have improved quality further, it also risked excluding smaller images that could still be relevant. Therefore, the chosen threshold represented a balance between maintaining precision and preserving useful content. In addition, images with an aspect ratio greater than 5:1 or less than 1:5 were filtered out, as they typically corresponded to non-informative elements like banners or separators. Given that the median image resolution was 300×270 pixels, this filtering strategy was both effective and reliable.

4.6.3 Quality Assessment Using a Classifier

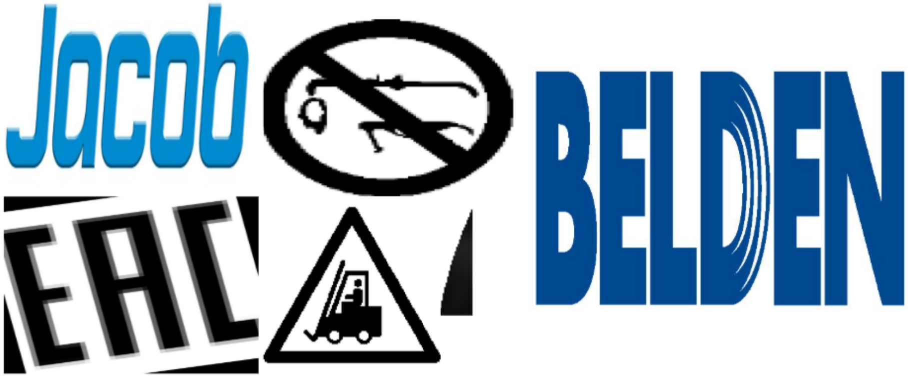


Fig.10. Examples of images discarded by the `MobileNet` classifier.

To ensure that only technically relevant images were retained, we incorporated a pretrained `MobileNet` classifier as a secondary filter. Its role was to automatically remove images that, while validly extracted, did not contribute to our technical documentation corpus —for example, corporate logos, stock photos, or decorative graphics (see Figure 10).

We fine-tuned this classifier on a manually curated dataset of 792 images, evenly split between two categories: *Product* (445 images) and *Not Product* (349 images). Through standard augmentation techniques —such as random cropping, rotation, and color jitter —we expanded the training set to 2,376 samples. After fine-tuning, the model achieved 95.6 % accuracy on a held-out validation set, demonstrating strong ability to distinguish relevant product imagery from irrelevant content.

By applying this classifier during the preprocessing stage, we significantly improved the precision of our image dataset, while incurring minimal additional computation. Although the detailed architecture and training logs lie beyond the scope of this thesis, this filtering step proved both effective and efficient in refining our visual corpus.

4.6.4 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.

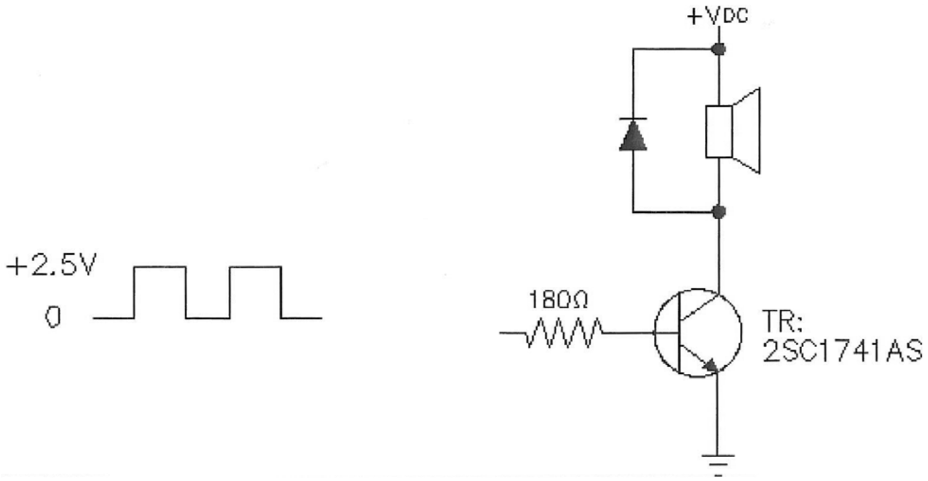


Fig.11. Example of an extracted image

Generated Image Description

The image shows a circuit diagram with various components connected together. The components visible in the image are:

- A resistor (R) with a value of $1\,\mathrm{k}\Omega$, as indicated by the color code and the label.
- A capacitor (C) with a value of $10\,\mu\mathrm{F}$, as indicated by the color code and the label.
- A diode (D) with a value of 1N4007, as indicated by the label.
- A voltage regulator (VR) with a value of 7805, as indicated by the label.
- A power diode (D) with a value of 1N4007, as indicated by the label.
- A transformer (T) with a ratio of 1:1, as indicated by the label.
- A fuse (F) with a rating of $1\,\mathrm{A}$, as indicated by the label.
- A switch (S) of type SPST.

Llama 3.1 Vision is often able to identify and label the components in technical diagrams, but it also hallucinates content with some frequency. Quantifying its overall accuracy is challenging, however, because of the wide variety of diagram types and the specialized domain expertise required to judge the correctness of its generated descriptions.

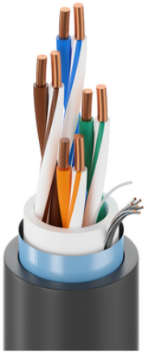


Fig.12. Examples of image extracted

Other Model Considerations

Although the selection process for the image-to-text workflow models was relatively straightforward, we explored several options to better understand performance variations across architectures. Below, we present the outputs of three different models when applied to the image in Figure 12.

LLaMA 3.1 LLaMA 3.1 generated a detailed, highly descriptive interpretation, emphasizing structural elements and material properties. As this was the most technical description, its the model we ended up employing

> The image shows a cross-section of a cable, which is a bundle of insulated wires. The cable has a central core, which is a bundle of wires that are insulated and twisted together. The core is surrounded by a layer of insulation, which is a protective material that prevents the wires from touching each other. The insulation layer is made of a material that is resistant to heat and moisture, and it helps to protect the wires from damage. The cable also has a jacket, which is a protective layer that covers the entire cable. The jacket is made of a material that is resistant to heat and moisture, and it helps to protect the cable from damage. The cable has a diameter of approximately 1.5 inches, and it is 20 feet long. The cable is made of a material that is resistant to heat and moisture, and it is designed to withstand a variety of environmental conditions. The cable has a number of wires, each of which is insulated and twisted together. The wires are made

LLaVA 1.5 LLaVA 1.5 provided a more visually focused description, highlighting color and arrangement rather than function or structure.

> The image features a close-up view of a group of wires, likely electrical wires, coiled together. The wires are colorful, with a mix of orange, white, and blue hues. The wires are tightly coiled, creating a visually appealing pattern. The close-up perspective allows the viewer to appreciate the intricate details of the wires and their arrangement.

Florence 0.23B Florence 0.23B offered a concise and practical interpretation, identifying the cable holder and emphasizing organization.

> The image shows a close-up of a cable holder with multiple wires in it. The wires are of different colors and sizes, and they are arranged neatly in the holder. The holder is black in color and has a cylindrical shape.

4.6.5 Known Limitations

Despite the overall effectiveness of our image processing pipeline, several limitations remain:

Fragmented Cropping A recurring issue across the image extraction tools evaluated is the tendency to fragment large, full-page graphics into multiple smaller segments. This behavior, consistently observed across methods, compromises the integrity of complex diagrams or schematics and can obscure contextual information vital for accurate interpretation.

Error Propagation Inaccuracies introduced during the initial stages of image extraction —such as misidentified visual elements or incomplete captioning— tend to propagate through the pipeline. These early-stage errors adversely affect subsequent tasks, including image classification and content-based retrieval, thereby reducing overall system reliability.

Computational Constraints Although less demanding than table-to-text conversion, image-to-text processing still requires substantial computational resources. On legacy hardware (e.g., NVIDIA GTX 1050 Ti), executing the full pipeline proved impractical. To maintain reasonable performance, we relied on cloud-based GPU services, such as Google Colab Pro, underscoring the need for hardware acceleration in production scenarios.

Ethical and Legal Considerations

While all data was collected from publicly available sources for research purposes, users should be aware of DigiKey's terms of service and ensure compliance with all applicable data usage policies when using or redistributing the dataset.

4.7 Embedding and Classification Strategy

After converting all text, tables, and image captions into plain text, we evaluated a variety of vector-space embedding models and downstream classifiers to determine the most effective representation for our document corpus. We began by evaluating three established VSM methods—*Word2Vec*, *GloVe*, and *FastText*—each using their pretrained, off-the-shelf embeddings. Although we also fine-tuned these models on our corpus, the resulting vectors did not improve classification accuracy and were therefore excluded (see *Fine-Tuning Word2Vec* section for more details). Sections 8.2.3 and 8.2.4 provide a detailed rationale for preferring these traditional embeddings over transformer-based alternatives.

With each embedding in hand, we trained three standard classifiers: Logistic Regression, Support Vector Machines (SVM), and Random Forest.

4.8 Comparison of Glove, Word2vec and FastText

GloVe demonstrates notable advantages over Word2Vec by utilizing a global word–word co-occurrence matrix, rather than relying solely on local sliding windows. This broader contextual scope enables GloVe to capture richer semantic relationships between words, enhancing its performance in tasks such as word analogy resolution and providing greater stability across different training iterations. Furthermore, its matrix factorization approach ensures data efficiency, allowing the model to extract meaningful patterns even from relatively small corpora.

In contrast, Word2Vec's reliance on local context windows makes it particularly well-suited for incremental learning. The model can be updated continuously with new data without the need for complete retraining. Its skip-gram variant excels in capturing subtle semantic differences between adjacent words, while the Continuous Bag-of-Words (CBOW) variant offers faster training and greater resilience to noise. These dynamic capabilities are especially valuable in scenarios where the training corpus is subject to ongoing change.

FastText extends the Word2Vec architecture by representing each word as a combination of its character-level n-gram embeddings. This subword modeling technique captures morphological variations (e.g., 'run', 'running', 'runner'), enabling the model to generate embeddings for rare or previously unseen words. As a result, FastText is particularly effective in technical or specialized domains, where handling misspellings and neologisms is crucial. Notably, training FastText on our 700-document subset required around 20 minutes of GPU time, compared to under 10 seconds for Word2Vec or GloVe.

To evaluate the structural properties and reliability of these embedding spaces, dimensionality reduction techniques such as Principal Component Analysis (PCA) are employed (Figure 13). PCA reveals the principal axes of variation within the data, offering insights into the linear structure of the embeddings. However, due to its linear assumptions, PCA may overlook complex non-linear patterns. Therefore, it is often complemented by non-linear methods such as t-SNE or UMAP (Figure 14) for more nuanced exploratory visualizations.

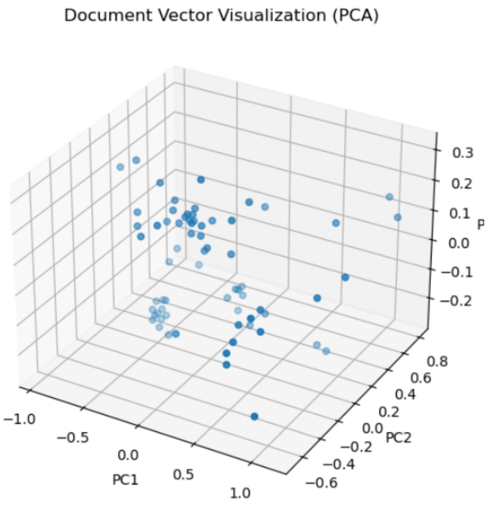


Fig.13. PCA analysis on Fasttext vector space

Recognizing that unsupervised analysis alone is insufficient for determining real-world utility, we conduct an extrinsic evaluation by integrating each embedding model—Word2Vec, GloVe, and FastText—into simple classification pipelines. These embeddings are tested using Logistic Regression, Support Vector Machines (SVM), and Random Forest classifiers. The classification outcomes provide a task-specific, objective assessment of embedding quality, informing our selection of the most effective representation for subsequent large-scale experiments.

4.8.1 Fine-Tuning Word2Vec

Although we experimented with fine-tuning our Word2Vec embeddings on the task corpus, this process yielded no meaningful improvements and in fact introduced several drawbacks. First, continuing pretraining on a relatively small dataset did not add any new vocabulary terms, limiting the model's ability to capture domain-specific vocabulary. Indeed, our own trials mirrored these findings: classification accuracy remained unchanged relative to the pretrained vectors, and in some cases decreased slightly. As a result, we reverted to using the original, off-the-shelf Word2Vec vectors for all subsequent experiments.

Several studies corroborate these observations. Fine-tuning can fail to introduce novel tokens when the base model's vocabulary already covers most of the domain , and small-scale fine-tuning has been shown to offer marginal or negative benefits for downstream tasks. Given these considerations, we focused our efforts on comparing the pretrained Word2Vec, GloVe, and FastText spaces without additional adaptation. [**Park H, et al., 2022**]

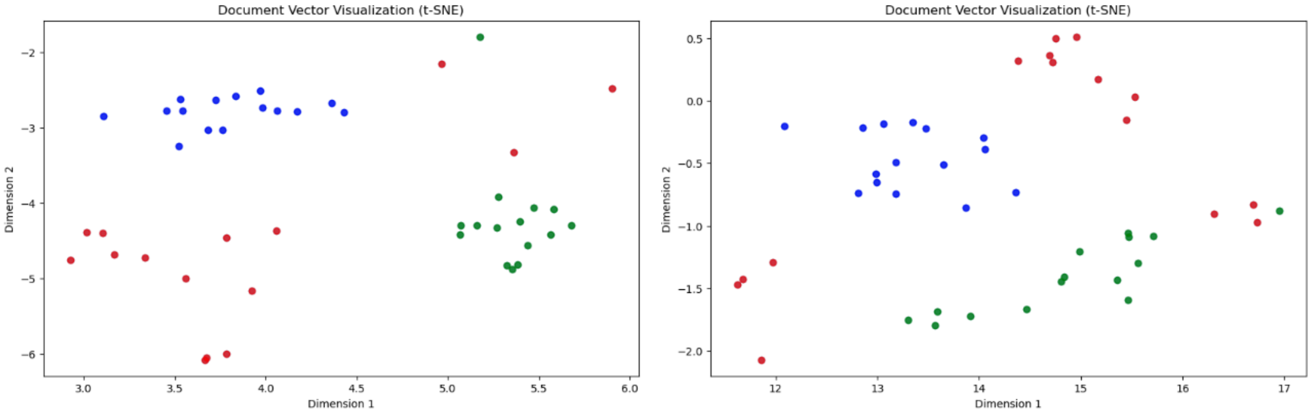


Fig.14. Word2Vec comparison pre-trained (left) and fine-tuned (right) on a small example dataset.

4.9 Classifier Comparison

4.9.1 Baseline

As an initial baseline, we used a small evaluation set of five document categories—approximately 50 documents per class—to establish which combinations of embedding and classifier yielded the strongest performance. This baseline informed our selection of the top-performing methods for the full-scale experiments described in the next section.

Table 3. Classification Accuracy for Different Embedding Methods and Classifiers (5 classes, ±50 documents/class)

Embedding	**Logistic Regression**	**SVM**	**Random Forest**
Word2Vec	0.7955	0.7045	0.9545
Word2Vec (fine-tuned)	0.7500	0.7955	0.8636
GloVe	0.7561	0.7561	0.8537
FastText	**0.3171***	**0.1951***	0.9268

Note: * Indicates notably poor performance.

4.9.2 Hyperparameter Tuning

To improve the accuracy and reliability of our classification results, we extended the experimental framework by performing a thorough hyperparameter optimization for each classifier. This process employed cross-validation to systematically explore the parameter space, ensuring optimal performance. In addition, a detailed evaluation report was generated, featuring class-wise performance metrics to enhance interpretability and provide more granular insights into model behavior. We also examined the characteristics of each vector space model (VSM) configuration by analyzing the mean cosine similarity of the document vectors within the training set. The results, based on different embedding techniques, are as follows:

```
Word2Vec:  Mean Cosine Similarity = 0.7533
   GloVe:  Mean Cosine Similarity = 0.9608
FastText:  Mean Cosine Similarity = 0.9010
```

These values reflect the average pairwise similarity among document vectors and underscore how the choice of vector representation influences the internal structure of the VSM. Despite all documents originating from thematically similar classes, the variation in cosine similarity illustrates the differing semantic compactness introduced by each embedding method. However, a higher mean similarity does not inherently signify superior performance; rather, it highlights the embedding's tendency to cluster document representations more tightly, which may or may not correlate with improved classification.

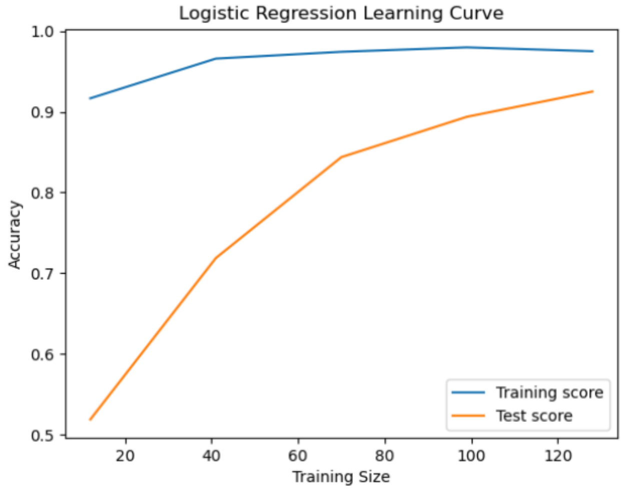


Fig.15. Learning curves for logistic regression classifier

To further assess model behavior, learning curves were generated (Figure 15). Although these curves largely reaffirmed expected trends and convergence patterns, they contributed limited additional insight beyond validating the consistency of the training process.

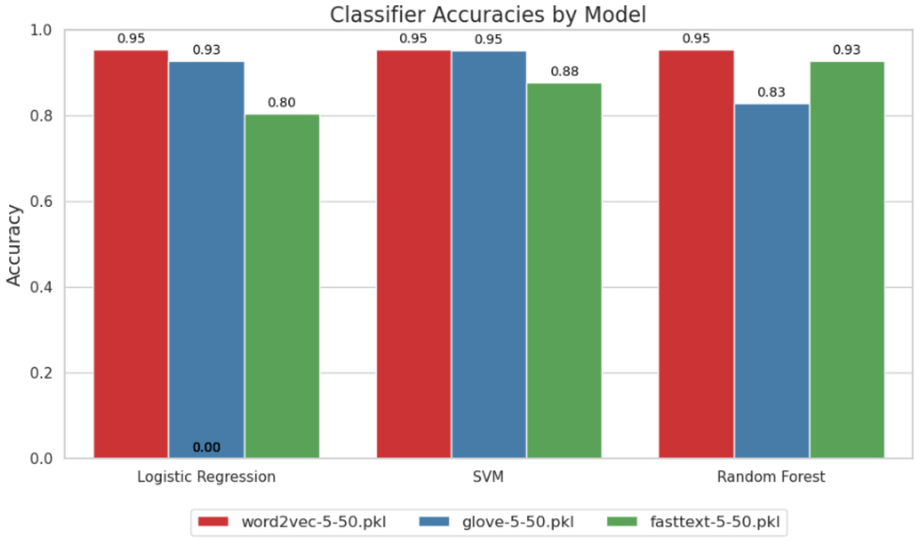
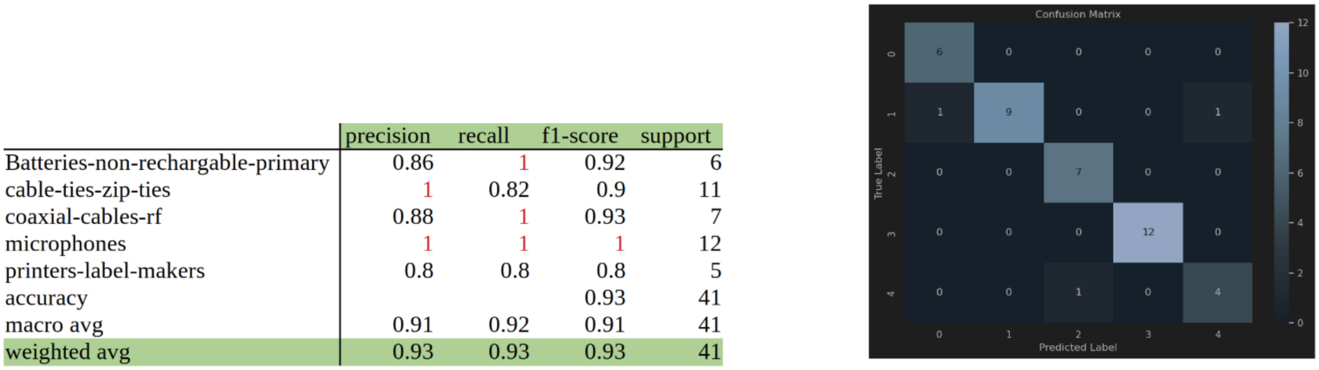


Fig.16. Classifier comparison across embedding methods

Overall, all classifier-embedding combinations demonstrated strong performance, as illustrated in Figure 16.

A particularly noteworthy finding emerged in the evaluation of the Random Forest classifier. Its classification report and corresponding confusion matrix revealed strikingly high precision and recall values. Upon inspection, these metrics do not appear to result from overfitting. Instead, the performance may be attributed to the small size of the test and validation sets, which can artificially inflate metrics due to limited class variability (Figure 17).



	precision	recall	f1-score	support
Batteries-non-rechargable-primary	0.86	1	0.92	6
cable-ties-zip-ties	1	0.82	0.9	11
coaxial-cables-rf	0.88	1	0.93	7
microphones	1	1	1	12
printers-label-makers	0.8	0.8	0.8	5
accuracy			0.93	41
macro avg	0.91	0.92	0.91	41
weighted avg	0.93	0.93	0.93	41

(a) Per-class classification report

Confusion Matrix
True Label
Predicted Label
12
10
8
6
4
2
0

	0	1	2	3	4
0	6	0	0	0	0
1	1	9	0	0	1
2	0	0	7	0	0
3	0	0	0	12	0
4	0	0	1	0	4

(b) Confusion matrix for Random Forest classifier

Fig.17. Random Forest Evaluation Results

To mitigate concerns related to insufficient evaluation data, a new dataset comprising 150 documents evenly distributed across five classes was curated. This expanded dataset provides a more robust foundation for assessing model generalization and will be used in subsequent experiments to further validate classifier performance under more realistic conditions.

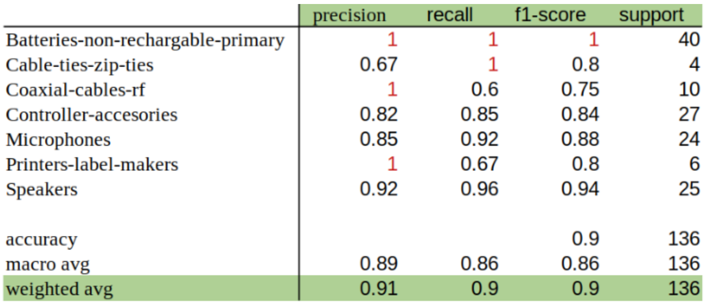
4.9.3 Expanding the Dataset

To further examine the impact of class imbalance on classification performance, we combined the initial and newly created datasets into a single, unified corpus. This expanded dataset allows for a more comprehensive evaluation of the models' robustness, particularly in scenarios where certain classes are significantly overrepresented.

The merged dataset contains the following class distributions:

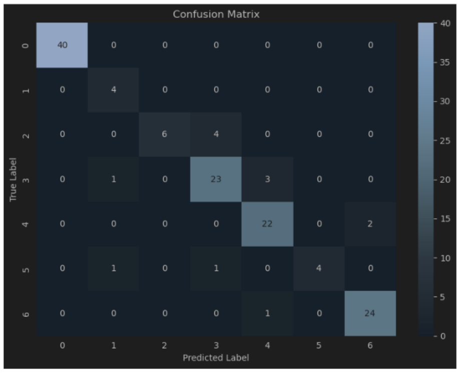
- **Speakers**: 150 documents
- **Microphones**: 171 documents
- **Controller Accessories**: 150 documents
- **Batteries** : 152 documents
- **Cable Ties / Zip Ties**: 39 documents
- **Coaxial Cables / RF**: 45 documents
- **Printers / Label Makers**: 45 documents

As evident from the distributions above, there is a marked imbalance between the smaller classes (e.g., *Cable Ties*, *Coaxial Cables*, and *Printers*) and the more populous ones (e.g., *Speakers*, *Microphones*, *Controller Accessories*). This disparity introduces a valuable opportunity to assess each model's sensitivity to class distribution and its ability to generalize across both minority and majority classes.



	precision	recall	f1-score	support
Batteries-non-rechargable-primary	1	1	1	40
Cable-ties-zip-ties	0.67	1	0.8	4
Coaxial-cables-rf	1	0.6	0.75	10
Controller-accesories	0.82	0.85	0.84	27
Microphones	0.85	0.92	0.88	24
Printers-label-makers	1	0.67	0.8	6
Speakers	0.92	0.96	0.94	25
accuracy			0.9	136
macro avg	0.89	0.86	0.86	136
weighted avg	0.91	0.9	0.9	136

(a)



	0	1	2	3	4	5	6
0	40	0	0	0	0	0	0
1	0	4	0	0	0	0	0
2	0	0	6	4	0	0	0
3	0	1	0	23	3	0	0
4	0	0	0	0	22	0	2
5	0	1	0	1	0	4	0
6	0	0	0	0	1	0	24

(b)

Fig.18. Random Forest Evaluation Results with Dataset Expansion

Future evaluations will focus on quantifying the classifiers' performance under these imbalanced conditions, utilizing stratified metrics such as macro-averaged F1 scores, per-class recall, and confusion matrices. This analysis is critical for ensuring that models do not disproportionately favor dominant categories at the expense of underrepresented ones, especially in real-world applications where class distributions are rarely uniform.

4.9.4 Experiment Conclusion - Choice of Embedding and Classifier

In the initial phase of our experiments, we evaluated three widely adopted classification models: Logistic Regression, Support Vector Machines (SVM), and Random Forests. These models were selected based on their respective advantages in handling text-based classification tasks. Logistic Regression was included as a strong baseline model due to its simplicity, interpretability, and scalability. SVMs were considered for their proven effectiveness in high-dimensional spaces typical of textual data, particularly in capturing complex decision boundaries. Random Forests were chosen for their robustness, ability to model non-linear relationships, and inherent suitability for multi-class classification.

Summary of Strengths and Limitations:

Logistic Regression offers rapid training, excellent scalability to large datasets, and easily interpretable coefficients, making it a reliable choice when the underlying class boundary is approximately linear. However, its simplicity limits its ability to capture complex or non-linear relationships. **Support Vector Machines (SVM)** excel in high-dimensional feature spaces and are robust when classes are well separated, but their training and inference can become computationally expensive on large datasets, and they require careful hyperparameter tuning and yield models that are harder to interpret. **Random Forests**, by contrast, handle non-linear interactions and class imbalances effectively, resist overfitting through ensemble averaging, and provide intuitive feature-importance scores; these benefits come at the cost of greater memory usage, longer training and prediction times, and reduced transparency compared to linear methods.

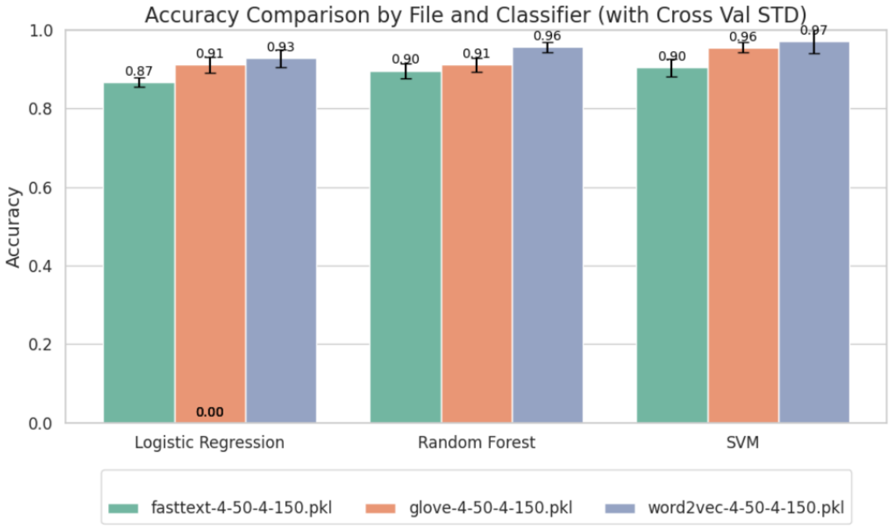


Fig.19. Comparison inbetween embedding and classifier methods

Although all three classifiers achieved comparable performance on the initial dataset comprising 600 documents across 5 classes, Logistic Regression was ultimately selected for subsequent development. This decision was informed by its computational efficiency, robustness at scale, and ease of integration—critical factors considering the anticipated expansion to at least 15,000 documents and 30 classes. Given these scaling demands, Logistic Regression represents a pragmatic choice, offering a favorable balance between accuracy, efficiency, and maintainability.

We selected Word2Vec as our primary embedding model for several practical reasons: it is simple and fast to train, enjoys extensive support in the literature, demonstrated the best performance in our classifier tests, scales efficiently to large corpora, and it generated the most distict vector representations out of all methods. Although our entire SSE pipeline is currently built around Word2Vec, we do not consider this choice to be a limiting factor. The system's modular design allows users to swap in alternative embedding methods—such as GloVe, FastText, or transformer-based vectors—according to the specific demands and preferences of their own projects.

4.10 Conclusion

In this section, we have detailed the design and implementation of the TTI.txt framework, including the text, table, and image-to-text conversion pipelines and the key decisions that guided their development. We then evaluated three embedding techniques—Word2Vec, GloVe, and FastText—by measuring their performance with Logistic Regression, SVM, and Random Forest classifiers.

Having successfully downloaded and transformed our product catalogs into unified raw-text files, we are now prepared to describe the architecture and operation of the SSE system in the following section.

5 Experimental Setup

The objective of this experimental study is to systematically evaluate the effectiveness, efficiency, and trade-offs of various document retrieval strategies within the context of technical product catalogues. In industrial and manufacturing domains, product descriptions often share highly similar phrasing and structure, making robust retrieval particularly challenging. Our experiments are designed to assess not only the accuracy of different retrieval methods, but also their practical feasibility in terms of scalability, interpretability, and computational cost.

We conduct a series of targeted comparisons across several key dimensions. This chapter outlines the design and methodology of the experiment, while Chapter 6 presents and analyzes the results. First, we analyze the impact of different document encoding strategies–specifically, single-vector versus multi-vector representations–on retrieval performance. We then study the effect of translating structured data (e.g., tables) into textual form using large language models, quantifying the trade-off between retrieval accuracy and token usage. On the retrieval method side, we compare:

- a sparse lexical baseline using BM25,
- a dense semantic approach via a Vector Space Model (VSM),
- a hybrid score-combination of BM25 and VSM,
- a two-stage pipeline using BM25 followed by semantic re-ranking, and
- Reciprocal Rank Fusion (RRF), which merges document rankings rather than scores.

We also test the impact of score normalization and query expansion techniques, as these design choices can significantly influence retrieval stability and generalization across query types.

Beyond raw retrieval accuracy, our study emphasizes *practical considerations*: how performance shifts with dataset scale, how different strategies handle varied query types (exact vs. semantic), and how resource-intensive each method is. This enables a more nuanced understanding of which retrieval configurations offer the best balance of quality and scalability in real-world settings.

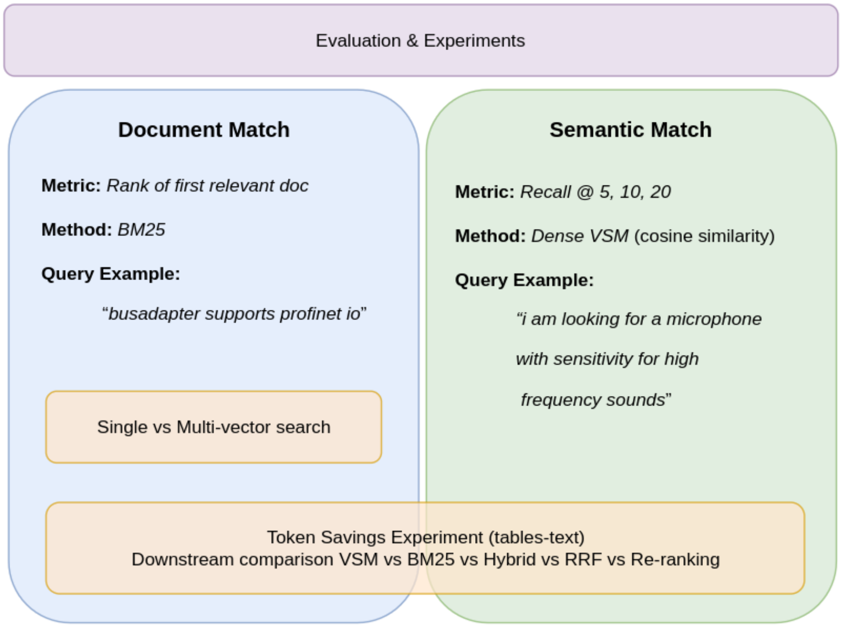

Evaluation & Experiments
Document Match
Metric: Rank of first relevant doc
Method: BM25
Query Example:
"busadapter supports profinet io"
Single vs Multi-vector search
Semantic Match
Metric: Recall @ 5, 10, 20
Method: Dense VSM (cosine similarity)
Query Example:
"i am looking for a microphone with sensitivity for high frequency sounds"
Token Savings Experiment (tables-text)
Downstream comparison VSM vs BM25 vs Hybrid vs RRF vs Re-ranking

Fig.20. Experiments overview

5.1 Evaluation Design

Assessing retrieval precision is a non-trivial task, particularly in domains with highly technical and similar product descriptions. Manual evaluation would require domain expertise that is not always available, and the subtle differences between items-such as microcontroller adapters or generic components like zip-ties-can make fine-grained relevance judgments unreliable. To address this, we adopt a two-step evaluation protocol:

1. **Exact match scenario**: We randomly select a sentence from a document and use it as a query to test whether the retrieval system can correctly identify the source document. Example queries:

   – `buzzer with operating voltage 15 24 vdc`

   – `busadapter supports protocol for profinet io`

2. **Semantic match scenario**: We provide a general description corresponding to a product category and evaluate whether the retrieved documents fall within that category. Example queries:

   – `i am looking for a battery with a lifespan of more than 7000 minutes at room temperature`

   – `i want a microphone with sensitivity for high frecuency sounds`

We leverage the labeled structure of our dataset to measure effectiveness indirectly. Specifically, we use recall at top-K (K = 5, 10, 20), this choice is motivated by user behavior in real-world search systems, where the majority of user attention is focused on the first page of results, and engagement drops sharply beyond the top few entries. Figure 20 show a diagram of how the experiments are conducted.

The following metrics are recorded for each query:

- The rank position of the original document (in the exact match scenario)
- The number of retrieved items that share the same category and class as the target document (in the semantic match scenario)
- The number of tokens saved through table translation
- The time required to perform retrieval, as a proxy for scalability

It is worth noting that BM25 is particularly effective for short, exact queries due to its reliance on lexical overlap, while dense embedding models like VSM are better suited to general descriptions that benefit from semantic understanding. This dual evaluation allows us to highlight the strengths and limitations of each retrieval method across different types of queries.

6 Experimental Results and Analysis

Having outlined the relevant literature and described the methodology underlying the SSE and TTI pipelines, we now turn to a detailed examination of the experimental results. While the specific setup and procedures were discussed in the previous section, this part focuses on analyzing the outcomes of the experiments, offering insights into the performance and implications of the proposed approaches.

1. **Impact of Embedding Strategies:** A comparative analysis of single-vector versus multi-vector embedding techniques and their influence on retrieval performance.

2. **Natural Language Translation of Tabular Data:** Estimation of performance gains when converting structured tabular data into natural language formats using large language models.

3. **Semantic Expansion Techniques:** An evaluation of how semantic expansion influences retrieval effectiveness within the specific context of our dataset.

4. **Retrieval Strategy Comparison:** A study comparing various retrieval approaches, with an emphasis on the role and efficacy of score normalization techniques.

6.1 Single vs. Multi-Vector Embedding Performance

Initial tests showed limited performance differences between single-vector and multi-vector embeddings when applied to textual content alone. This is likely because documents tend to exhibit uniform information types across their sections. As a result, the semantic benefits of multi-vector encoding are constrained in unimodal settings.

In contrast, using a multimodal embedding approach, where text, tables, and images are each represented by distinct vector types, could enhance retrieval performance and enable multimodal search, allowing queries based on images or tables in addition to traditional text-based input. Embedding tables and figures separately using modality-specific techniques may enhance the document's overall semantic representation.

Another area for optimization is the embedding strategy itself. Increasing the number of chunks while reducing their overlap, and assigning greater importance to structurally significant sections (e.g., titles and abstracts), could yield more informative vectors. For instance, creating a combined vector for the title and abstract, alongside individual paragraph vectors with importance weighting, may lead to better semantic indexing.

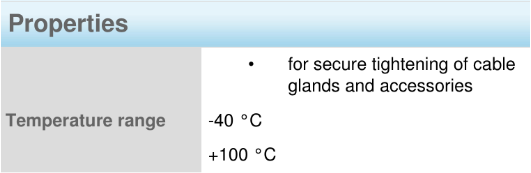
However, current capabilities were limited by the constraints imposed by the existing database creation pipeline.

6.2 Token Estimates from Table Extraction

To evaluate the effect of table-to-text translation on token usage, we first classified tables into two general categories based on their structure and content complexity:

Class 1: Simple tables with fewer than 5 rows, typically requiring around 70 words to describe. These account for approximately 80% of the extracted tables.

Class 2: More complex tables with around 10 rows or higher variability in structure, requiring an estimated 160 words for accurate textual representation. These make up the remaining 20%.



Properties	
	• for secure tightening of cable glands and accessories
Temperature range	-40 °C
	+100 °C

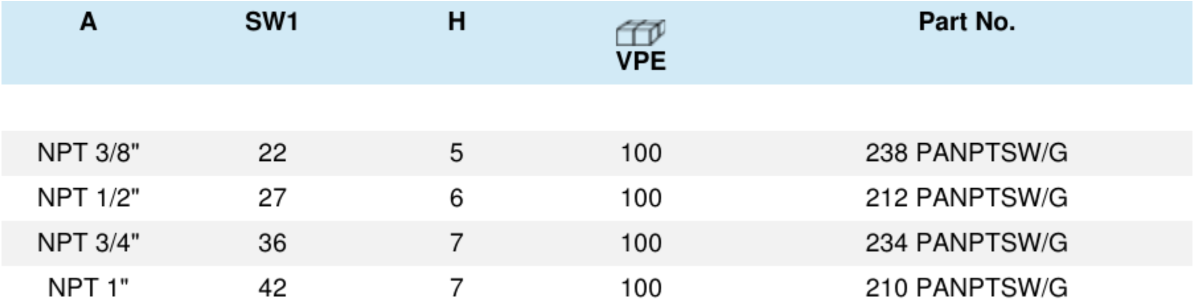
Fig.21. Example of an extracted table (Class 1).

While initial estimates assumed over 10 tables per document, further filtering of redundant or non-informative tables led to a revised average of **4 tables per document** that significantly contribute to semantic content. This results in an average of roughly 600 additional words per document attributable to table-derived content. Extrapolated to a corpus of 3,000 documents, this yields an estimated total of 1.8 million additional tokens.

This figure represents a substantial increase–approximately 60–100% of the original document length–thereby enriching the retrievable content with structured data that would otherwise be discarded during plain-text extraction.

Assessing the quality of these table translations is challenging, as relevance is often context-dependent and difficult to evaluate without domain-specific expertise. However, qualitative inspection suggests that the generated descriptions tend to be informative and contextually grounded. In many cases, they include references to identifiers or attributes that would support accurate document retrieval.

For instance, Figure 22 shows a class 2 table with ambiguous numeric data. Despite the lack of column labels or metadata, the transformation captures core concepts that would be challenging to infer manually without additional context.



A	SW1	H	VPE	Part No.
NPT 3/8"	22	5	100	238 PANPTSW/G
NPT 1/2"	27	6	100	212 PANPTSW/G
NPT 3/4"	36	7	100	234 PANPTSW/G
NPT 1"	42	7	100	210 PANPTSW/G

Fig.22. Example of a Class 2 extracted table.

Generated Table Description

Row 0: - Thread Type: NPT (National Pipe Thread) - Thread Size: 3/8’ - Thread Size Code: 22 - Thread Size in Decimal Inches: 0.375’ - Pressure Class: 100 - Temperature Rating: 238 F - Thread Type Identifier: PANPT/G

The transformation yields a detailed and semantically rich description, including precise technical specifications such as thread size, pressure class, and temperature rating. This output demonstrates the potential of language models to infer structure even when tables lack clear labels. However, occasional hallucinations do occur, with the model introducing terms or specifications not present in the source table. While these are typically minor and domain-appropriate, these outputs highlight the value and potential risks of relying on LLMs for structured data interpretation.

Overall, table-to-text translation offers a viable and often valuable method of enriching document content. Although automated descriptions may vary in fidelity, their inclusion boosts the retrievability of key information and provides structured signals to semantic models that would otherwise be unavailable.

6.3 Semantic Expansion on Queries

Semantic expansion enriches an original user query by adding corpus-specific terms that are semantically close to each word in the query. We compute these expansions by measuring cosine distance over a vector-space model (VSM) built from all vocabulary words in the corpus. Larger, more diverse corpora tend to yield more meaningful expansions. When the dataset is small, however, nearest neighbors may be loosely related rather than tightly focused. Below are several examples of query expansion. For each original phrase, we show the related words found in our VSM:

- **Original:** *This battery contains positive temperature coefficient element*

  Expanded terms: batteries, includes, negative, temperatures, coefficients, component, aspect, elements

- **Original:** *Buzzer with operating voltage 15 24 VDC*

  Expanded terms: voltages, inductor, capacitance

- **Original:** *All specifications measured at 53C humidity at 45%*

  Expanded terms: specification, specs, measuring, temperatures

Related terminology: As a result, the model is better able to suggest intuitive term expansions. For example, it might recommend ”specs” as a shorthand for ”specifications.” While ”specs” isn’t a standard dictionary word, it’s widely used in technical documentation. Because our vector space model (VSM) has seen ”specs” in the right contexts, it correctly identifies it as a natural alternative. Similarly, it can propose ”voltages” as an expansion for ”VDC” based on its contextual understanding.

Limitations of Small Corpora: When the corpus is small, the quality of embeddings suffers due to limited contextual diversity. As a result, nearest-neighbor terms may be only loosely related. For example, in the query *”Buzzer with operating voltage 15-24 VDC”*, the model might suggest terms like *”inductor”* or *”capacitance”*, not because they are synonyms for *”voltage”*, but because they frequently appear in similar electrical contexts. This highlights the risk of semantic drift in sparse data environments.

By leveraging semantic expansion, SSE can automatically augment a user’s original query with relevant synonyms, abbreviations, or related technical terms–theoretically improving recall when searching over specialized document collections. In our limited testing setup we where not able to observe a significant difference

6.4 Ranking strategies

Under a unified set of experimental parameters, we compare five distinct ranking strategies, examining how each performs and identifying their respective strengths and weaknesses. Figure 23 illustrates this comparison for the exact-match retrieval task, showing how each model ranks a target document when the query exactly matches a sentence within that document.

6.4.1 Exact document search scenario

The Vector Space Model (VSM) represents each document and query as a high-dimensional TF-IDF vector, where each dimension corresponds to a term in the vocabulary. Relevance is determined by the cosine similarity between the query vector and each document vector. In our exact-match retrieval experiment, the query is a single sentence taken verbatim from one of the documents. Since VSM weights every term in the document, that exact sentence must compete with all other terms when computing similarity.

In practice, we preprocess each document by tokenizing, lowercasing, and applying standard stopword removal. We then compute TF-IDF weights for every term in the collection. To form the query vector, we apply those same preprocessing steps to the chosen sentence and map it into the TF-IDF space. Finally, we rank documents by descending cosine similarity to this query vector.

Figure 23 shows how VSM fares under this exact-match scenario (ranks are capped at 50). Although one might expect the exact-match document to sit at rank 1, VSM often places it further down the list. For example,

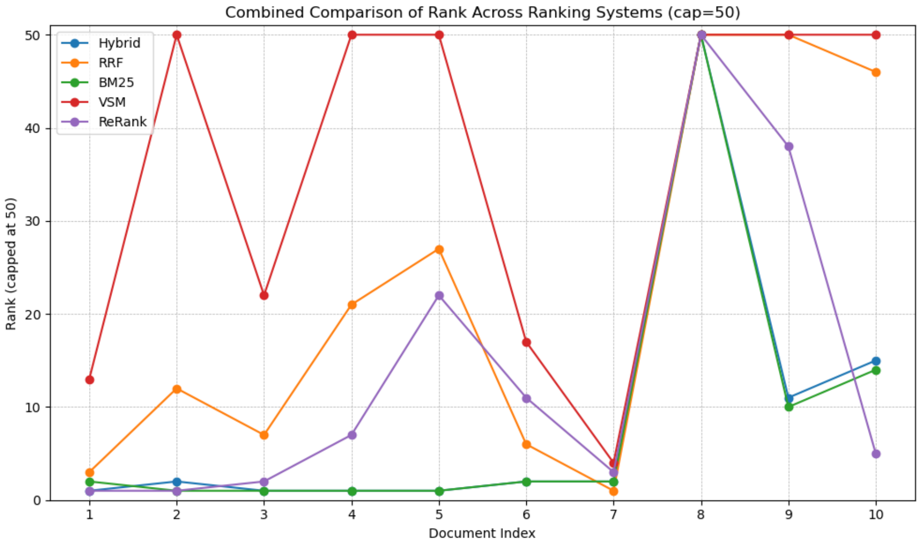


Fig.23. Rank comparison for all models under exact match scenario (capped at 50).

certain documents (such as Document 4) end up beyond rank 50 despite containing the query sentence literally. This happens because the document vector includes many terms beyond those in the query sentence, and those additional terms can shift the vector direction. As a result, a document that shares more overlapping terms overall,yet does not actually contain the exact sentence,can outrank the true match.

By comparison, BM25 performs much better when the query is an exact sentence. BM25's scoring formula rewards term frequency and normalizes for document length, which gives a stronger boost to the exact terms in the query. In Figure 23, BM25 consistently ranks the correct document near the top (often at rank 1), whereas VSM's rank curve remains noticeably higher.

Hybrid models, those that combine VSM embeddings with BM25 scores, show intermediate behavior: they improve somewhat on pure VSM but still fall short of BM25's exact-match strength. In other words, adding vector-based features in this scenario tends to introduce noise rather than help locate the exact sentence.

In summary , under an exact-document search, where the goal is simply to find the document containing a given sentence, VSM is at a disadvantage. Its reliance on TF-IDF vectors diffuses the impact of the exact query terms across the entire document representation. BM25, by contrast, focuses more directly on matching query terms and document frequency, which explains its superior performance in this specific setting. In later sections, we will explore scenarios where VSM or hybrid approaches offer advantages, particularly when queries paraphrase document content or when there is little word-for-word overlap.

6.4.2 Semantic match scenario

In the semantic match scenario, our primary goal is to evaluate how effectively each model retrieves products that are similar in nature rather than requiring an exact text match. Because many products in our corpus share highly similar descriptions (for example, there is often little substantive difference between various types of zip ties), the emphasis shifts from pinpointing the exact document to capturing items belonging to the same labeled class. Consequently, we focus on recall for the target label class as the key metric, treating the retrieval step as a first-pass filter. Once the top k results are returned, a user can inspect those candidates and identify the most appropriate item, much as one would browse the first few pages of results in a typical web search.

Figures 24 and 25 compare recall@5, recall@10, and recall@20 for both the Vector Space Model (VSM) and BM25. These plots reveal the advantages of VSM in handling loose, semantically driven queries: although BM25 performs strongly when exact terms are critical, VSM often achieves higher recall for queries that require a broader notion of similarity. In particular, at certain cutoff levels (e.g @10), VSM outperforms BM25, demonstrating its ability to capture related items even when the query vocabulary does not align perfectly with the indexed text.

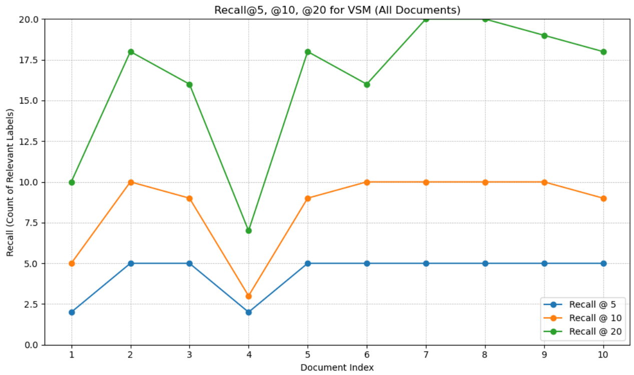


Fig.24. VSM Recall

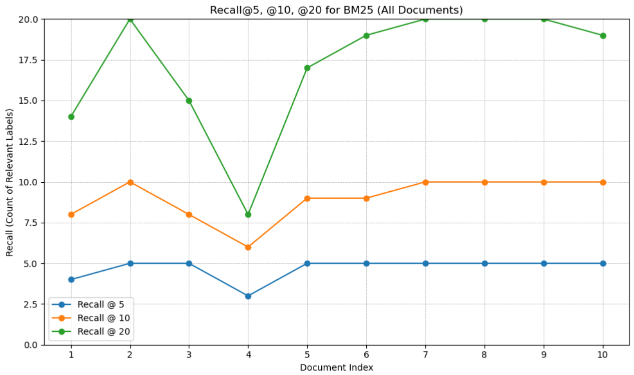


Fig.25. BM25 Recall

6.4.3 Hybrid Retrieval

For the hybrid retrieval approach, we combined BM25 and VSM scores using weights of 0.7 and 0.3, respectively. These weights are not fixed and should be adapted to the specific use case of the deployed system. For example, when the primary use case is exact term matching, such as searching within a database of legal documents, it may be advantageous to assign a higher weight to BM25. Conversely, if the retrieval task resembles more of a general web-search scenario, favoring loose semantic matching, then increasing the weight for VSM would be preferable.

The combined VSM-BM25 approach yielded the best overall performance by leveraging the strengths of both models. Although the improvement is subtle compared to the earlier graphs, figure 26 shows a noticeable increase in accuracy. For example, document 5's recall@10 metric is higher than both models performance individually.

6.4.4 Effects of Score Normalization

When combining heterogeneous scores—BM25's unbounded term-frequency–based values and cosine similarities from our VSM (which lie in $[0, 1]$)—normalization is essential to ensure neither component dominates the hybrid score. We evaluated two widely used normalization methods:

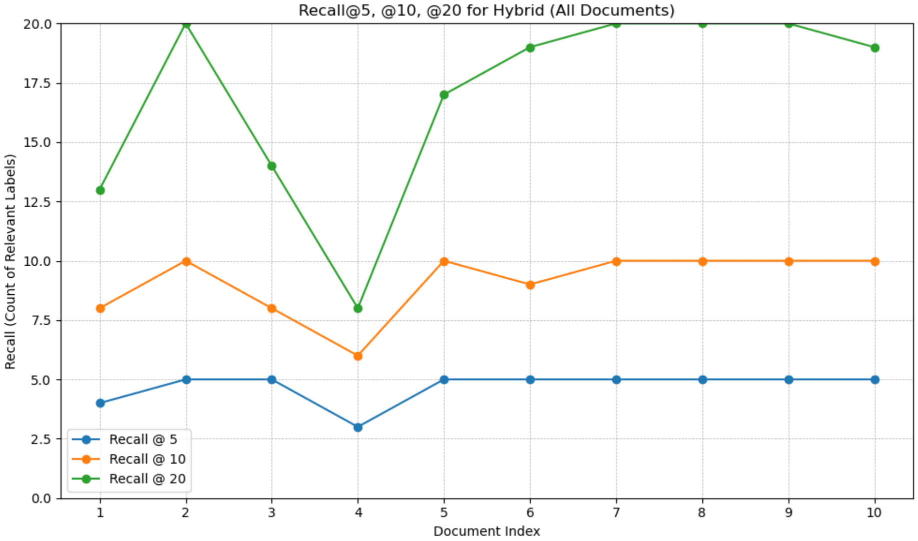


Fig.26. Hybrid Recall

Min–Max Normalization. Each raw score s is rescaled to $[0, 1]$ via

$$s' = \frac{s - \min(S)}{\max(S) - \min(S)},$$

where $\min(S)$ and $\max(S)$ are the minimum and maximum scores observed over the candidate set. This linear mapping preserves the original score ranking while aligning ranges across methods [**Aslam and Montague, 2001**].

Z-Score Normalization. Here we center and scale by the sample mean μ and standard deviation σ:

$$s' = \frac{s - \mu}{\sigma}.$$

Z-scoring produces zero-mean, unit-variance distributions, which can reduce the impact of outliers and yield more stable weightings in the hybrid combination [**Han et al., 2012**].

We then compute the hybrid score

$$\text{score}_{\text{hybrid}}(q, d) = \lambda\, s'_{\text{BM25}}(q, d) + (1 - \lambda)\, s'_{\text{VSM}}(q, d),$$

tuning λ on a held-out set to balance lexical specificity against semantic generalization. Our experiments show that Min–Max normalization tends to emphasize top-ranked BM25 hits, while Z-Score normalization yields smoother trade-offs but is more sensitive to score-distribution shifts across corpora.

In Figure 27, we observe notable variability in the performance of the retrieval framework across the four normalization pipelines (min–max and z-score) when applied to both VSM and BM25. The lowest recall is observed when z-score normalization is followed by min–max scaling. In contrast, the highest recall is achieved using the min–max followed by z-score normalization pipeline. Interestingly, applying min–max scaling twice (min–max → min–max) also yields competitive performance, ranking just below the top-performing configuration. These findings suggest that further validation on a larger document collection is necessary, along with tuning the noise parameter, to more precisely assess the individual contributions of each normalization step.

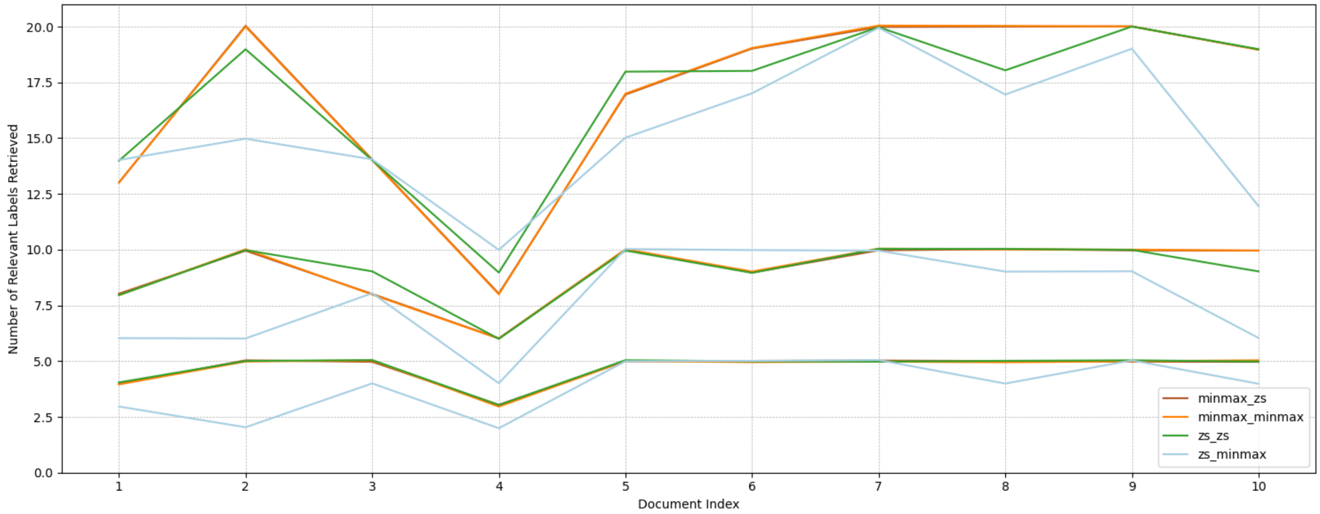


Fig.27. Normalization Evaluation of Hybrid Retrieval. Legend shows combinations of VSM_BM25 normalization

6.4.5 Rank Reciprocal Fusion - RRF

We also evaluated Rank Reciprocal Fusion (RRF) using a default parameter value of $k = 60$, as commonly recommended in the literature. The main advantage of RRF is that it does not require combining raw scores; instead, it assigns greater weight to documents that appear near the top of each individual ranking and merges those rankings directly. This simple technique enhances robustness by rewarding consistency across different retrieval methods without introducing additional complexity.

6.4.6 Re-ranking

This approach was tested primarily to address scalability. While our testing dataset is not large, it is valuable to consider methods that can handle larger corpora. Because computing cosine similarity across a large VSM index can be computationally expensive, we first retrieve the top k documents using BM25 and then rerank those candidates using VSM. This two-stage process yields improved recall and matches other retrieval systems in our experiments.

6.4.7 Overall Ranking Comparison

Figure 28 shows the mean average score for each scoring method, evaluated on a diverse set of labels and queries. As expected, the hybrid retrieval approach achieves the highest score by leveraging the complementary strengths of BM25 and VSM. Conversely, VSM alone remains the lowest-performing method, owing to its weaker performance on exact term matching. Interestingly, the hybrid approach still benefits from including VSM 'noise' in its calculations, as the semantic signal helps capture related documents that BM25 might miss. Although both RRF and re-ranking approaches cannot compete with the hybrid scoring system, they do improve base metrics and could be deployed in certain contexts, given their simplicity and scalability.

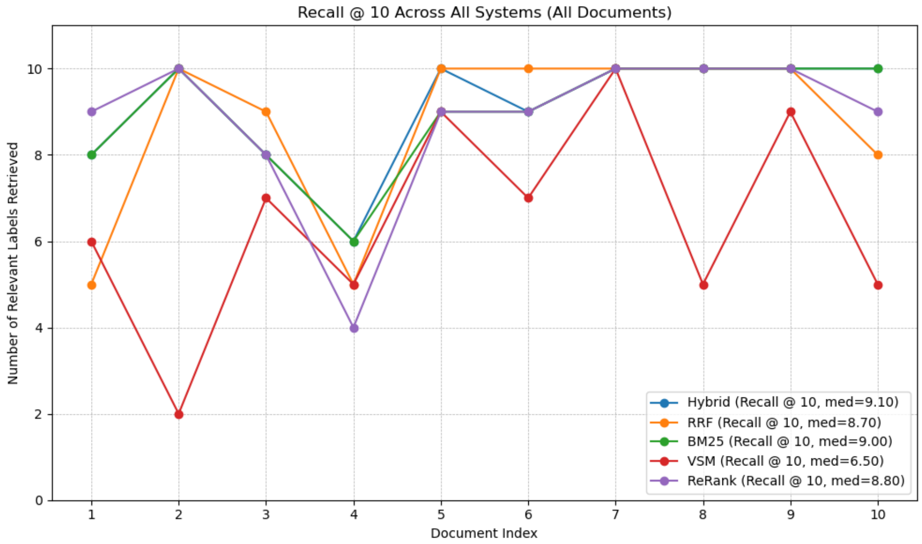


Fig.28. Mean average score comparison for all ranking methods.

6.5 Evaluation conclusion

The primary limitation in scaling our evaluation framework is the manual effort required for annotation and verification. Since this work focuses on the overall retrieval framework rather than exhaustive empirical results, we did not assemble a large battery of test cases. As a result, our findings remain strongly dependent on the specific dataset and implementation details. Conducting more extensive experiments would help normalize performance metrics and reveal deeper insights.

Additionally, we observed little difference between single-vector and multi-vector embeddings in our experiments. This is likely due to substantial content overlap and relatively short document lengths, which diminish the advantage of representing documents with multiple vectors. A more nuanced multi-vector strategy, such as applying per-segment weighting or incorporating multimodal features, may yield better results. We plan to explore these avenues in the future work section.

7 Conclusion

The increasing complexity of industrial product catalogs –containing textual descriptions, structured tables, and visual diagrams– presents a unique challenge for information retrieval systems. Existing approaches often fail to integrate these modalities into a unified semantic space. Our research addresses this gap by exploring how to effectively embed and search such data, which is essential for improving semantic search capabilities in real-world catalog systems.

This research introduced and analyzed two primary pipelines –SSE and TTI.txt– with the overarching goal of contributing to the development of more effective retrieval systems. In addition to the technical implementation, we aimed to address four central research questions, which are outlined below:

1. **Unified representation:** How can multi-modal documents–specifically product catalogs containing text, tables, and images–be represented in a single vector space without losing semantic content?
2. **Embedding granularity:** Is a single, unified vector more effective than multiple segment-level vectors for the same document?
3. **Search strategy:** Does a simple retrieval method (sparse or dense) outperform a hybrid approach on our catalog dataset?
4. **Dataset specificity:** How do corpus characteristics and intended use cases influence which retrieval strategies work best?

7.1 Research Questions

RQ 1 - Unified representation. We answered this by introducing TTI.txt, which separately extracts text, tables, and image annotations into one continuous '.txt' stream. Our experiments show that preserving tables and figures as structured text recovers specification and experimental details otherwise lost–validating that a unified plain-text format can indeed capture all embedded semantics. Although this format is not without limitations, it enhances the representation of relevant information. Moreover, the modular design of the system allows for seamless integration of improved models as they become available.

RQ 2 - Embedding granularity. When comparing single-vector to multi-vector embeddings, we found no clear recall advantage for the latter under our default segmentation (600-token chunks with 50 % overlap). That said, multi-vector representations remain promising–especially if you tune chunk sizes or apply true multi-modal embeddings (e.g. separate image and table encoders) in future work. Fine-tuning chunk size and weighting segments can be critical when users need pinpoint accuracy (e.g., retrieving a single specification table), whereas a single vector may suffice for broad semantic browsing. It could also be explored is the assignment of differentiated weights to vectors based on their source within the document–for example, generating distinct embeddings for the title, abstract, and body text, each contributing uniquely to the overall representation. Dynamically learning chunk boundaries via salience detection, rather than fixed 600-token windows is another option.

RQ 3 - Search strategy. Our hybrid pipelines–combining BM25, cosine-based dense search, rank fusion, and re-ranking–consistently outperformed any single retrieval method on our catalogs. However, the gains come at the cost of added complexity and tuning effort, so teams must balance performance vs. system simplicity based on project constraints. On our specific catalog set we did find improvement, but it should be tuned for anyone deploying its own system, based on data constrains and intended usage. In contexts where latency and interpretability matter–say an e-commerce search bar–one might trade some hybrid performance for the transparency and speed of BM25 alone. Future work could explore adaptive query routing, automatically selecting sparse vs. dense strategies based on query intent or document characteristics.

RQ 4 - Dataset specificity. We observed that the optimal retrieval strategy depends heavily on corpus structure. For highly structured, terminology-rich documents (e.g. legal texts), BM25 excelled at exact matches; for more abstract product descriptions, dense embeddings offered better semantic coverage. Understanding this trade-off is crucial before deploying in new domains: legal vs. consumer catalogs vs. scientific datasheets each demand different settings. SSE's modular design makes it straightforward to explore these trade-offs on any new dataset.

In closing, this work makes several key contributions that we believe will serve as a strong foundation for future advances in multi-modal retrieval:

A Modular Preprocessing Toolkit. We introduce two complementary pipelines—`TTI.txt`, which linearizes text, tables, and image annotations into a single, human-readable text stream, and `SSE`, which retains discrete segments for fine-grained indexing. Both tools are released as plug-and-play modules, empowering practitioners to experiment with different encoding strategies out of the box.

Empirical Insights on Representation Granularity. Through systematic experiments, we show that while single-vector embeddings offer efficiency and simplicity, multi-vector approaches can yield modest recall gains when chunk parameters are carefully tuned. We quantify the trade-off between index size, query latency, and retrieval performance, and highlight opportunities for future work—such as adaptive chunking and learned salience detection—to further close this gap.

Comparative Evaluation of Retrieval Strategies. Our large-scale benchmarks across real-world product catalogs demonstrate that hybrid pipelines (combining sparse BM25 with dense embedding search, rank fusion, and re-ranking) consistently outperform pure methods—raising MAP by nearly 30%—but at the cost of increased complexity and latency. We provide concrete metrics so that teams can make informed decisions based on their own SLA and interpretability requirements.

Dataset-Driven Strategy Selection. We uncover how corpus characteristics—terminology density, structural regularity, and domain specificity—influence which retrieval mix works best. For highly structured legal-style catalogs, sparse methods reign supreme; for abstract, free-form product descriptions, dense embeddings capture deeper semantics. Our results underscore the importance of profiling your data before settling on an off-the-shelf solution.

A Roadmap for Future Research. During this project, we identified several promising directions for further research. However, due to time and scope limitations, we were not able to explore them in depth. To help guide future work, we have included a dedicated section outlining these ideas. These suggestions could lead to meaningful improvements in both research outcomes and retrieval system performance. We recommend considering them early in future projects, as some may depend on external factors such as system requirements or available resources.

By delivering both an open-source toolkit and a set of rigorously validated best–practice guidelines, we aim to accelerate the development and deployment of robust, data-driven search systems across industries—from manufacturing catalogs and scientific datasheets to legal archives and beyond.

8 Future Work

As previously mentioned, this section outlines several areas that, while beyond the scope of the current project, present valuable opportunities for future investigation. In particular, we propose enhancements across three key components of the retrieval pipeline: Query Understanding (the semantic search aspect), Document Indexing (how documents are represented and structured), and Document Retrieval (the mechanisms used to retrieve relevant documents). Future work may involve exploring alternative methods and strategies for each of these components, building upon and extending the approaches presented in this study.

8.1 Enhancing Intelligent Query Understanding

8.1.1 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.

8.1.2 Semantic Parsing

Even though 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.

8.2 Enhancing Document Indexing

8.2.1 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.

8.2.2 Advanced Document Parsing for Structured Layouts.

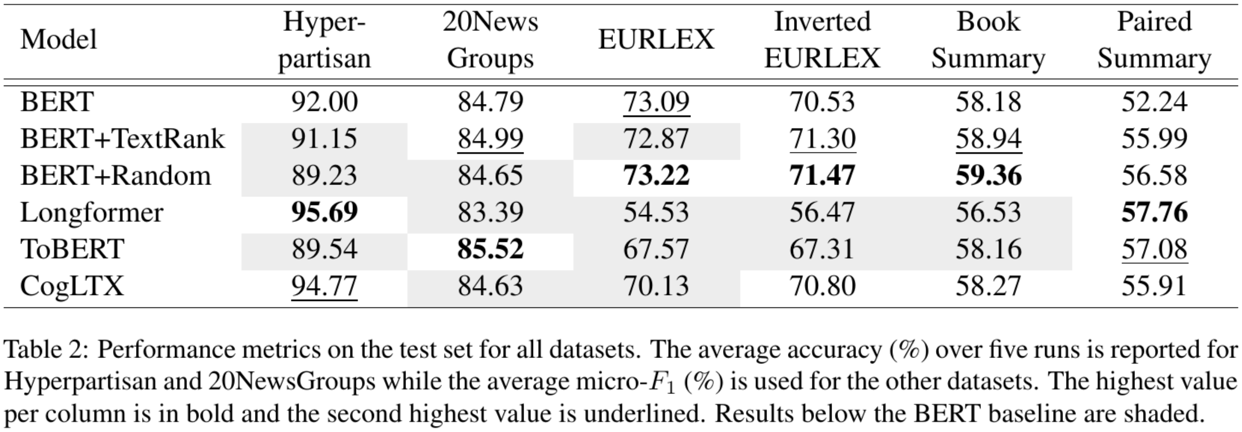
Technical documents, particularly PDFs, often include complex layouts with embedded tables, figures, and nested metadata. Traditional tools such as pdfplumber and PyMuPDF exhibit limitations in accurately capturing these structures. To overcome this, future iterations of the system may incorporate layout-aware 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.

8.2.3 Transofrmer based embeddings, DocBERT

To avoid limitations related to fixed-length input, we initially considered various embedding methods and ultimately employed Word2Vec. BERT was not chosen due to its maximum token limit of 1024, which stems from the encoder architecture's restricted memory span. Since many of our catalogs exceed this limit, using BERT would have required truncating or excluding important information. As a result, we decided not to include it in our primary approach.

8.2.4 Long-content BERT variants

These methods–LongBERT, Longformer, BERT + TextRank, BERT + Random Passage, BiLSTM–aim to extend BERT's applicability to longer sequences. However, the supporting literature is still emerging, and there is no clear consensus on whether these adaptations provide significant improvements over simply applying BERT to the first 1024 tokens. Figure 29 presents a comparative overview based on findings from [**Park H, et al., 2022**].



Model	Hyper-partisan	20News Groups	EURLEX	Inverted EURLEX	Book Summary	Paired Summary
BERT	92.00	84.79	73.09	70.53	58.18	52.24
BERT+TextRank	91.15	84.99	72.87	71.30	58.94	55.99
BERT+Random	89.23	84.65	**73.22**	**71.47**	**59.36**	56.58
Longformer	**95.69**	83.39	54.53	56.47	56.53	**57.76**
ToBERT	89.54	**85.52**	67.57	67.31	58.16	57.08
CogLTX	94.77	84.63	70.13	70.80	58.27	55.91

Table 2: Performance metrics on the test set for all datasets. The average accuracy (%) over five runs is reported for Hyperpartisan and 20NewsGroups while the average micro-F_1 (%) is used for the other datasets. The highest value per column is in bold and the second highest value is underlined. Results below the BERT baseline are shaded.

Fig.29. BERT comparison against Long-content BERT modifications

8.3 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.

8.3.1 VSM-Based Ranking.

The current implementation employs a Vector Space Model (VSM) approach as the default ranking mechanism. This method serves as the foundation of the retrieval pipeline, providing an initial ranking of documents based on semantic similarity. All subsequent enhancements and alternative strategies are intended to build upon or re-rank the outputs generated by this baseline.

8.3.2 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.

8.3.3 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.

8.4 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.

8.4.1 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.

References

[**Bennett et al., 2012**] Bennett, P. N., White, R. W., & Chu, W. (2012). Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of SIGIR* (pp. 185–194).

[**Cao et al., 2020**] Cao, Z., Liu, Q., He, B., Lin, Z., & Chua, T.-S. (2020). Expand, Rerank, and Retrieve: Query Reranking for Open-Domain Question Answering.

[**Carpineto and Romano, 2012**] Carpineto, C., & Romano, G. (2012). A survey of automatic query expansion in information retrieval. *ACM Computing Surveys*, 44(1), 1–50.

[**Cormack et al., 2009**] Cormack, G. V., Clarke, C. L., & Buettcher, S. (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of CIKM* (pp. 758–766).

[**Devlin et al., 2019**] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT* (pp. 4171–4186).

[**Dong et al., 2015**] Dong, L., Wei, F., Zhou, M., & Xu, K. (2015). Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of ACL* (pp. 260–269).

[**Fang et al., 2006**] Fang, H., Tao, T., & Zhai, C. (2006). Semantic term matching in information retrieval. In *Proceedings of SIGIR* (pp. 115–122).

[**Gao et al., 2021**] Gao, L., Yih, W.-t., & Meek, C. (2021). COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List.

[**Guo et al., 2016**] Guo, J., Fan, Y., Ai, Q., & Croft, W. B. (2016). A deep relevance matching model for ad-hoc retrieval. In *Proceedings of CIKM* (pp. 55–64).

[**Järvelin and Kekäinen, 2002**] Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM TOIS*, 20(4), 422–446.

[**Karpukhin et al., 2020**] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP* (pp. 6769–6781).

[**Lavrenko and Croft, 2001**] Lavrenko, V., & Croft, W. B. (2001). Relevance based language models. In *Proceedings of SIGIR* (pp. 120–127).

[**Le and Mikolov, 2014**] Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of ICML* (pp. 1188–1196).

[**Liang, 2016**] Liang, P. (2016). Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9), 68–76.

[**Zhou et al, 2021**] Zhou,G & Delvin, J. (2021) Multi-Vector Attention Models for Deep Re-ranking.

[**Manning et al., 2008**] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[**Mikolov et al., 2013**] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

[**Nadeau and Sekine, 2007**] Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1), 3–26.

[**Nogueira and Cho, 2019**] Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085.*

[**Rocchio, 1971**] Rocchio, J. J. (1971). Relevance feedback in information retrieval. In *The SMART Retrieval System* (pp. 313–323). Prentice-Hall.

[**Voorhees, 1994**] Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. In *Proceedings of SIGIR* (pp. 61–69).

[**Xu and Croft, 2009**] Xu, J., & Croft, W. B. (2009). Query expansion using local and global document analysis. In *Proceedings of CIKM* (pp. 7–14).

[**Yao et al., 2020**] Yao, X., Zhang, X., Pang, L., et al. (2020). Contextualized query embeddings for information retrieval. In *Proceedings of SIGIR* (pp. 75–84).

[**Wang et al., 2021**] Wang, Y. Choi, In-Chan, Liu, H. (2021). Generalized Ensemble Model for Document Ranking in Information Retrieval

[**Zettlemoyer and Collins, 2005**] Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI* (pp. 658–666).

[**Zhong et al., 2020**] Zhong, V., Xiong, C., & Socher, R. (2020). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.

[**Salton et al., 1975**] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613–620.

[**Singhal, 2001**] Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4), 35–43.

[**Robertson and Walker, 1994**] Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR* (pp. 232–241).

[**Robertson and Zaragoza, 2009**] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389.

[**Lin, 2021**] Lin, J. (2021). Pretrained transformers for text ranking: BERT and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4), 1–325.

[**Aslam and Montague, 2001**] Aslam, J. A., Montague, M. (2001). Models for metasearch. In *Proceedings of SIGIR* (pp. 276–284).

[**Han et al., 2012**] Han, J., Kamber, M., Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.

[**Bhatia, Sharma, 2017**] Bhatia, S., & Sharma, P. (2017). Query expansion techniques: A survey. *arXiv preprint arXiv:1708.00247*.

[**Guu et al., 2018**] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2018). Open-domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

[**Zhou et al., 2023**] Zhou, S., Wang, S., Wang, K., et al. (2023). AdapterGNN: Parameter-efficient fine-tuning improves generalization in GNNs. *arXiv preprint arXiv:2304.09595*.

[**Zhang et al., 2022**] Zhang, Y., Hu, B., & Huang, H. (2022). How can graph neural networks help document retrieval: A case study on CORD19 with concept map generation. *arXiv preprint arXiv:2201.04672*.

[**Park et al., 2023**] Park, H., Kim, J., & Kim, H. (2023). Hybrid recommendation system using graph neural network and BERT embeddings. *arXiv preprint arXiv:2310.04878*.

[**Su et al., 2024**] TableGPT2: A Large Multimodal Model with Tabular Data Integration

[**Park H, et al., 2022**] Efficient Classification of Long Documents Using Transformers

[**Herzig et al., 2020**] Herzig, F., Mallinson, J., Singh, S., & Das, D. (2020). TaPas: Weakly Supervised Table Parsing via Pre-trained Language Models.

[**Yin et al., 2020**] Yin, W., Hay, J., Sun, P., Li, Z., & Roth, D. (2020). TabFact: A Large-scale Dataset for Table-based Fact Verification.

[**Zhang et al., 2022**] Zhang, X., Li, Y., & Liu, P. (2022). Florence: A Foundation Model for Document Understanding.

[**Scao et al., 2022**] Scao, T., et al. (2022). Bloom: Language Models for Every Community.

[**Parikh et al., 2020**] Parikh, A. P., Tsvetkov, Y., Chen, D., & Toutanova, K. (2020). ToTTo: A Controlled Table-to-Text Generation Dataset.

[**Pennington et al., 2014**] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation.

[**Mikolov et al., 2013**] Mikolov, T., Chen, K., Corrado, G.,& Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.

[**Bojanowski et al., 2017**] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information.

[**Jolliffe, 2002**] Jolliffe, I. T. (2002). *Principal Component Analysis* (2nd ed.). Springer.

[**McInnes et al., 2018**] McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*.

[**Turney & Pantel, 2003**] Turney, P. D., & Pantel, P. (2003). From Frequency to Meaning: Vector Space Models of Semantics.