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

Methodology Overview

Pablo de Vicente Abad, 6/05/2025

This report outlines the methodology adopted for developing an intelligent semantic search system for industrial parts. The goal is to enable users to retrieve relevant technical documents or product specifications through natural language queries, enhanced with semantic understanding. The process can be schematized as follows:

Query → Query Transformation → Search in VSM → Document Retrieval

This transformed query is used to search within a pre-existing Vector Space Model (VSM), where all documents have been embedded. The final step involves retrieving the most semantically relevant documents based on proximity in the vector space.

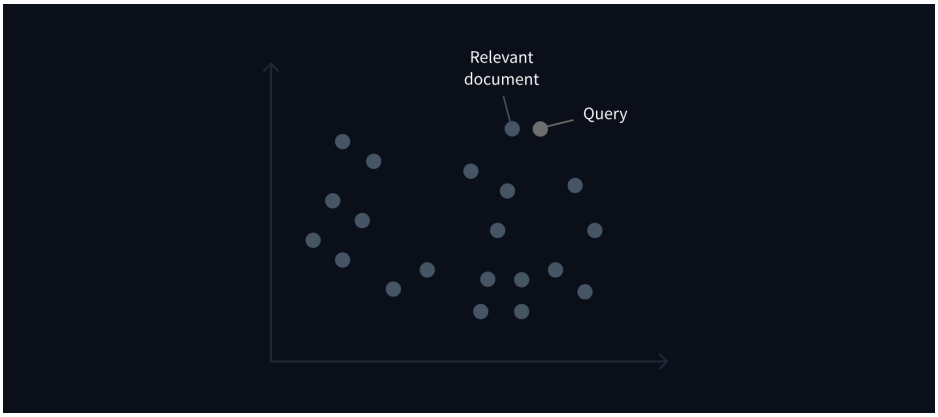


Fig 0: Retrieval on a VSM

Data & Vector Space Setup

The first step in building the semantic search system involves preparing the corpus and constructing the underlying vector space model (VSM) used for document retrieval. To achieve this, all part descriptions, specification sheets, and related documentation are extracted and converted into clean, plain-text format.

This step has already been accomplished in the Research Internship 2 ***Categorization of Textual Industrial (Technical) Reports***. Read documentation for more information about the process.

Intelligent Query Transformation (Intelligent Semantic Search)

Semantic search offers a significant advancement over traditional keyword-based retrieval by emphasizing user intent and contextual meaning. Unlike exact-match methods, semantic search interprets the relationships between words in a manner that more closely resembles human reasoning. In this project, although user personalization or history is not leveraged, semantic techniques still play a crucial role in enriching and disambiguating part-related queries. The following link provides a good overview on semantic search and the related concepts.

The primary goal remains straightforward: enable the user to look up a part using natural language. However, queries often vary in structure and terminology, so it is essential to translate them into a representation that aligns with the vocabulary and semantics of the vector space. This is achieved through the following multi-stage transformation pipeline:

1. Preprocessing

Apply tokenization, part-of-speech tagging, and lemmatization, and regular expressions to normalize units.

```
"3mm" = "3 mm"
```

2. Query Expansion

To improve recall and robustness, the query is automatically expanded with semantically related terms. One of the methods employed is **Synonym Substitution**, either through lexical databases like WordNet or contextual embeddings, to capture domain-relevant variations. This step helps bridge the gap between user phrasing and document terminology, particularly in highly specialized or variable industrial vocabularies.

```
"bolt" → {"fastener", "screw"},
"width" → {"diameter", "thickness"}
```

3. Semantic Parsing

To further enhance the query, **Semantic Parsing** techniques are applied. **Named Entity Recognition (NER)** is used to extract structured elements such as material types, dimensions, and component categories.

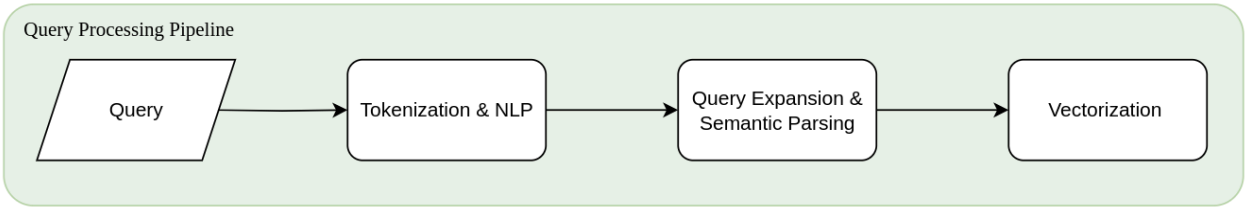
```
{"material" : "aluminum", "part_type" : "bolt"}
```

4. Vectorization

We chose FastText to turn our enriched queries into the same vector space as our documents. Its subword approach handles technical terms, abbreviations, and variations in industrial part names better than many other models. Embedding both queries and documents with FastText means we can directly compare them using vector similarity

> *For a detailed justification of FastText and its performance in domain-specific vector representations, refer to the Research Internship Report.*

By using this shared embedding model, the system is able to match queries to relevant documents not only through lexical overlap, but also through deeper semantic relationships encoded in the vector space.



Methodological Innovation Areas

The project presents two main opportunities for innovation across the intelligent search pipeline, particularly in **how queries are interpreted -Semantic Search Systems-** and **how documents are retrieved -Document Retrieval-** from the corpus.

1. Semantic Search Systems

While traditional semantic search systems often leverage user-specific context — such as location, browsing history, or long-term behavior — these signals are not available in our setting. Nonetheless, there is substantial room to explore alternatives that are **corpus-centric** and tailored to our set of data.

1.1. Intelligent Query Understanding

A key area of innovation lies in the **query expansion and semantic parsing process**. Since the system does not rely on individual user context, the emphasis shifts toward **corpus-aware** interpretation strategies. Instead of applying generic expansions (e.g., WordNet synonyms), the system can learn meaningful substitutions directly from the domain-specific corpus.

- The query “3 mm bolt” may be expanded to include “0.118 inch bolt” if imperial measurements are commonly used in the documents.
- Terms like “bolt” can be associated with related part types such as “fastener” or “screw” based on co-occurrence patterns in the data.

This approach, referred to as **Intelligent Substitution**, leverages corpus-derived patterns, frequent terminology, and domain conventions to steer the expansion process.

1.2. Corpus-Guided Semantic Models -- not implemented

One promising approach is the use of **Corpus-Steered Query Expansion** in combination with **Large Language Models (LLMs)**. Instead of relying solely on pretrained models, the idea is to fine-tune or prompt LLMs using the vocabulary and structure of the actual corpus, enabling more accurate and context-aware expansions.

We will not be employing this approach, but it would be interesting to explore.

1.3 Key References

We identify potential areas for methodological contribution, both in query expansion and semantic parsing. Below, there is relevant literature listed

Query Expansion Techniques: A Survey
https://arxiv.org/pdf/1708.00247
→To address the vocabulary problem, various techniques have been proposed, such as, relevance feedback, interactive query filtration, corpus dependent knowledge models, corpus independent knowledge models, search result clustering, and word sense disambiguation. Almost all popular techniques expand the initial query by adding new related terms. This can also involve selective retention of terms from the original query

Using word2vec to find additional query terms was especially useful for this particular dataset, which consists of call transcripts created through automatic speech recognition (ASR)

At the beginning of the seventies, the addition of similar terms into the initial query started playing a crucial role in QE (e.g.,[129,194,275]). Researchers assumed that a set of similar words that frequently appear in documents, belong to the same subject, thus, similar documents formed a cluster [215]. Two types of clustering have been discussed in document retrieval systems: clustering of terms and clustering of documents [275]. A well-known example of term based clustering is Qiu and Frei [220]'s corpus-based expansion technique that uses a similarity thesaurus for expanding the original query. A similarity thesaurus is a collection of documents based on specific domain knowledge, where each term is expressed as a weighted document vector. Another approach proposed by Crouch and Yang [75] built a statistical corpus thesaurus by clustering the entire document collection using the link clustering algorithm

2.2 Weighting and Ranking of Query Expansion Terms

Corpus-Steered Query Expansion with Large Language Models (2023)
https://arxiv.org/pdf/2402.18031
→Explores LLM prompting for domain-sensitive information retrieval.

A review of ontology based query expansion
ScienceDirect
→This paper examines the meaning of context in relation to ontology based query expansion and contains a review of query expansion approaches

A survey of Semantic Parsing Techniques
https://www.mdpi.com/2073-8994/16/9/1201
→This study adopts a systematic review method and strictly follows the PRISMA framework, deeply analyzes the key ideas, methods, problems, and solutions of traditional and neural network methods, and explores the model performance, API application, dataset, and evaluation mechanism

Really good survey. Infeasable to carry out semantic parsing on my own

Rule-based approaches involve creating a set of rules for the grammar of a language. The rules are then used to identify entities in the text based on their structural and grammatical features. These methods can be time-consuming and may not generalize well to unseen data.

Machine learning approaches involve training an AI-driven machine learning model on a labeled dataset using algorithms like conditional random fields and maximum entropy (two types of complex statistical language models). Techniques can range from traditional machine learning methods (e.g., decision trees and support vector machines) to more complex deep learning approaches, like recurrent neural networks (RNNs) and transformers. These methods generalize better to unseen data, but they require a large amount of labeled training data and can be computationally expensive.

Furthermore, while general NER models can identify common entities like names and locations, they may struggle with entities that are specific to a certain domain
Domain-specific NER models can be trained on specialized, domain-specific data, but procuring that information can itself prove challenging.

bert-base-NER is a fine-tuned BERT model that is ready to use for Named Entity Recognition and achieves state-of-the-art performance for the NER task. It has been trained to recognize four types of entities: location (LOC), organizations (ORG), person (PER) and Miscellaneous (MISC). https://huggingface.co/dslim/bert-base-NER

https://medium.com/@b.terryjack/nlp-pretrained-named-entity-recognition-7caa5cd28d7b
ner implementation

2. Comparison of Ranking Strategies: VSM vs. GNN vs. Hybrid -- Skip to section 3.

An essential component of this system involves the evaluation of ranking strategies used to retrieve and order relevant documents. Three main approaches are considered: traditional **Vector Space Model (VSM)** ranking, **Graph Neural Network (GNN)**-based ranking, and a **hybrid method** that integrates both.

VSM-Based Ranking

The baseline approach relies on computing the **cosine similarity** between the vectorized query and each document in the corpus. The top-*K* documents with the highest similarity scores are retrieved. This method is efficient and well-established in semantic retrieval systems but lacks the ability to model higher-order relationships between documents.

GNN-Based Re-ranking

To address the limitations of pure VSM-based similarity, a second stage introduces **GNN-based re-ranking**. In this approach:

- **Stage 1**: The top-*K* candidate documents are initially retrieved using VSM.
- **Stage 2**: These candidates are **embedded into a graph structure**, where nodes represent documents and edges reflect relationships such as co-occurrence, shared terminology, or hierarchical part associations.
- A GNN is then applied over this subgraph to refine document representations or compute **context-aware similarity scores**.

2.1. Key References

I find **Paper 5** particularly interesting, as it aligns closely with the core objectives of this project. That said, there remains significant room for innovation in this area. I would greatly value external input to help refine and expand upon these ideas.

1. GNNRank: Learning Global Rankings from Pairwise Comparisons via Directed Graph Neural Networks
https://arxiv.org/pdf/2202.00211
→ Presents a solid framework for learning ranking functions with GNNs; useful as a conceptual basis for treating ranking as a graph-based problem. There is a lot of math here that i do not comprehend though

2. Semantic Evidence-aware Graph Neural Network (SE-GNN)
https://arxiv.org/pdf/2109.11800

→ Explores how to integrate semantic signals into GNNs for enhanced document understanding, which could strengthen re-ranking mechanisms in our system.

3. Open-Domain Question Answering Using Early Fusion of Knowledge Bases and Text
https://arxiv.org/pdf/1809.00782
→ While focused on QA, the way they structure and fuse graphs with textual data offers a compelling retrieval-oriented approach we can adapt.

4. AdapterGNN: Parameter-Efficient Fine-Tuning Improves Generalization in GNNs
https://arxiv.org/pdf/2304.09595
→ Very math-heavy, but highlights efficiency techniques in GNN fine-tuning; possibly useful in long-term iterations, though not a current focus.

5. How Can Graph Neural Networks Help Document Retrieval: A Case Study on CORD19 with Concept Map Generation
https://arxiv.org/pdf/2201.04672
→ Highly relevant to our task—directly explores GNNs for document retrieval; worth considering as a reference implementation and foundation for new contributions.

6. Hybrid Recommendation System using Graph Neural Network and BERT Embeddings
https://arxiv.org/abs/2310.04878
→ Focused on recommendation, but the integration of GNN and BERT provides valuable ideas for hybrid semantic retrieval.

2.2 Additional proposals

ISO 10303/STEP Ontologies: While STEP provides rich part ontologies, its rigid classification may not align with my use case, since bolts can serve in diverse assemblies.

Per-Document Graphs: I could parse each document into a micro-graph (nodes for materials, dimensions, tolerances), then merge these into a global graph for GNN training.

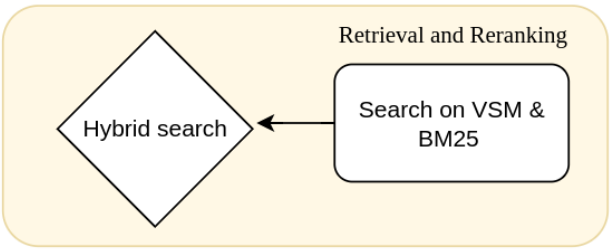
Advanced Document Parsing: For complex PDF layouts I may investigate LayoutLM or Donut, which often outperform traditional tools like pdfplumber or PyMuPDF in capturing tables and structured metadata.

3. Retrieval on VSM

After a more in-depth review, we concluded that the previously proposed methodological innovations—which relied on recomputing document embeddings—are infeasible given our current computational and time constraints. We will then work within the pre-constructed VSM space.

Instead, we propose to explore a hybrid retrieval approach that combines:

- **Dense embeddings** (generated by Word2Vec) for capturing semantic similarity, and
- **Sparse ranking** (BM25) for leveraging exact term-matching and document-length normalization.

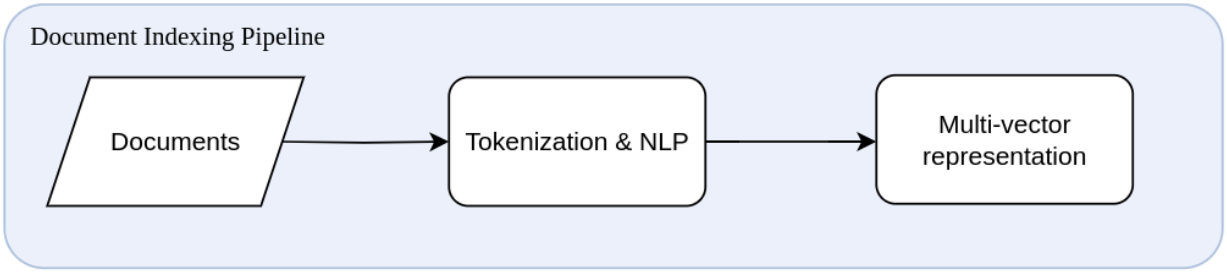


3.1 Multivector Retrieval Considerations

To balance context depth and embedding granularity, we can:

- **Index larger document chunks** for BM25 and coarse semantic matching, then
- **Embed smaller sub-chunks** (e.g., sentences) for fine-grained semantic ranking

This “multivector” strategy maintains the broader context in the sparse index while preserving high-resolution semantic signals in the dense index—helping to improve precision without sacrificing recall.



3.2 Key References

References are ordered in importance, gaining relevance as they are added.

1. **Query Expansion Based on NLP and Word Embeddings.** The authors augment traditional term-based retrieval with embedding-driven expansion—working more on the Query Expansion part.
   https://trec.nist.gov/pubs/trec27/papers/JARIR-CC.pdf

2. **Fusion Retrieval: Combining Vector Search and BM25 for Document Retrieval** Focused on answer generation. Uses "Hybrid Retrieval: A combination of FAISS search in similarity and BM25 scoring to improve accuracy". at one point in its methodology https://www.aionlinecourse.com/ai-projects/fusion-retrieval-combining-vector-search-and-bm25-for-enhanced-document-retrieval

3. **Word2Vec-Powered Algorithm for Efficient Retrieval of Bill of Quantities.** Details a domain-specific retrieval pipeline that uses averaged Word2Vec embeddings alongside BM25 re-ranking. It improves the BM25 algorithm by introducing positional weight —Interesting, even though it implements it for a different use case, we could use it for our task https://ieeexplore.ieee.org/abstract/document/10405620/

4. **Hybrid Search: Combining BM25 and Semantic Search for Better Results with LangChain.** A practical tutorial demonstrating how to integrate BM25 indexing with transformer-based semantic embeddings in a production-ready search application. Outlines the methodology and uses it for RAG. Our workflow would be similar to it https://medium.com/etoai/hybrid-search-combining-bm25-and-semantic-search-for-better-results-with-lan-1358038fe7e6

5. **Sparse Meets Dense: A Hybrid Approach to Enhance Scientific Document Retrieval.** Although not exactly what we will be doing, it does go into the hybrid approach for document retrieval https://arxiv.org/pdf/2401.04055

6. **Leveraging dense retrieval and summarization-based re-ranking for case law retrieval**. It does not use "sparse" retrieval but it includes ***paragraph*** search for complementing the initial ranking. Similar to our *chunk* approach in a sense. DoSSIER@COLIEE 2021: Leveraging dense retrieval and summarization-based re-ranking for case law retrieval

7. **Domain-specific Question Answering with Hybrid Search.** The system uses a linear combination of signals – cosine similarity from the dense retriever, BM25 scores, and domain - specific boosts (e.g. host matching) Again, not exactly our case but it does leverage more than one search system for question answering. Domain-specific Question Answering with Hybrid Search

Methodology Workflow

Although no single paper applies my exact combination of techniques, each component has precedent in the literature. What sets my approach apart is the increased depth and complexity of the preprocessing and data-construction pipeline. The workflow can be divided into:

1. **Query Processing Pipeline :** Transforms a query into the required format. Applying stemming, tokenization, tf-idf weighting, query expansion and semantic parsing.
2. **Document Indexing Pipeline :** We generate multiple embeddings for each document—one from a subword model, one from a word-piece model, and one using a domain-specific vocabulary—so we capture technical terms more effectively.
3. **Retrieval and Reranking :** Combines a search in the Vector Space Model with a rerank of the results, making use of BM25 search engine. The inner workings of the search system will be specified later on.

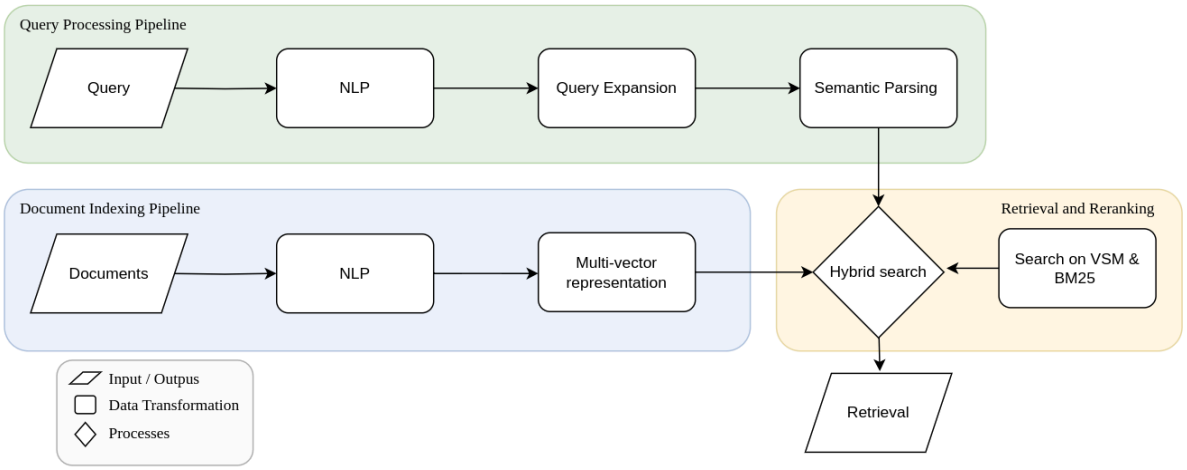


Fig 1: Methodology proposal overview.

These enhancements extend and deepen existing workflows: they borrow established methods but **chain and refine them in a unified pipeline**, giving more reliable inputs for model training and evaluation. This **end-to-end complexity**—especially in **database construction and chunk and query processing**—constitutes the core novelty of my proposed methodology.

Considerations and Additional Literature

Recent work on end-to-end semantic search offers valuable guidance for our hybrid retrieval pipeline, even if we do not replicate every component.

For example, the paper *Semantic Search Pipeline: From Query Expansion to Concept Forging*, published just one month ago, outlines a comprehensive workflow. The underlying strategy closely aligns with our approach: combining dense vector techniques with traditional term-based models.

Likewise, *A Semantic Search Pipeline for Causality-driven Adhoc Information Retrieval* demonstrates how multiple representations can be fused to improve recall and precision. Their architecture integrates learned embeddings with BM25 post-filtering, whereas our design will use Word2Vec-based expansion and BM25 ranking in tandem.

Unlike most recent pipelines—which focus on end-to-end RAG workflows and rely almost exclusively on heavy transformer encodings—our system integrates lightweight yet powerful **query expansion, explicit semantic parsing, and multivector embeddings within a transparent data-processing pipeline**. By first expanding user queries via IDF-filtered Word2Vec synonyms, then parsing key concepts to guide vector weighting, and finally combining multiple retrieval signals (dense vectors + BM25), we achieve both high recall and interpretability. In contrast to purely transformer-based or BM25-only methods, our hybrid approach lets us tune each stage—from tokenization through TF-IDF weighting—while tailoring retrieval strictly to document search, not generation.

Finally, there are two principal strategies for hybrid retrieval:

1. **Parallel fusion**: run both search A (e.g., Word2Vec) and search B (e.g., BM25) independently, then merge their top-k results.

The spacy library 1 was used for prepossessing (i.e. lemmatizing and tokenizing). We used the python rank25 library 2 to implement a lexical index optimized for Okapi BM25 scoring. The default values were used for the $k1$ (1.5) and b (0.75) parameters.

2. **Cascaded reranking**: first retrieve a preliminary set with search A, then rerank that subset using search B to produce the final top-k.

$Q1$, $Q2$, and $Q3$ each produce a set of candidate documents ($Q1'$, $Q2'$, and $Q3'$ respectively). These results are sent to an aggregator module that deduplicates and re-ranks all the candidate documents. If a document appears in multiple results sets, its scores are summed. The top 500 documents are returned as the final result set.

We are currently evaluating which strategy—fusion or cascaded reranking—yields the best balance of recall, latency, and ranking quality for our use case.

The results indicate that an approximate chunk size of 1000 characters with a split ensuring it ends on a sentence delimiter, with an overlap of 100 characters provides a favorable balance between retrieval granularity and computational efficiency. [7]

Results and Analysis

Effect of Chunk Size on Retrieval Performance

To determine the optimal chunk size for document segmentation, we experimented with various chunk sizes and overlap values. Table 1 summarizes the NDCG scores obtained for different configurations.

Chunk size	Chunk Overlap	Context nDCG
1000	100	**0.828**
2000	500	0.802
5000	1000	0.795

Table 1: Effect of Approximate Chunk Size on Retrieval Performance