Scholarly Topic Navigator: An Explainable Research Digest Pipeline for NLP Literature Discovery

Aditya Kanbargi
aditya.kanbargi@gwu.edu
G40754750

Pramod Krishnachari
pramodk@gwu.edu
G34050742

Trisha Singh
trishakomal.singh@gwu.edu
G20744725

Abstract

The proliferation of academic publications in natural language processing and machine learning presents significant challenges for researchers seeking to discover relevant work and understand emerging trends. Major repositories like ArXiv, ACL Anthology, and Semantic Scholar collectively host millions of research papers, with thousands added weekly. This information overload makes it increasingly difficult for researchers to maintain comprehensive awareness of their field, discover relevant prior work, and identify emerging research directions.

This paper presents the Scholarly Topic Navigator, a comprehensive end-to-end NLP pipeline designed to aggregate, process, classify, and explain academic research papers from multiple sources. The system collects papers from ArXiv (cs.CL, cs.LG categories), ACL Anthology, and Semantic Scholar, processing a final dataset of 22,522 research papers with complete abstracts and metadata. We implement a multi-stage pipeline incorporating text preprocessing using NLTK and spaCy, embedding generation using Word2Vec, Sentence-BERT, and SciBERT models, hybrid retrieval combining BM25 lexical search with FAISS semantic search, and classification using knowledge distillation from a zero-shot teacher model to a lightweight student classifier.

Topic modeling experiments compare traditional Latent Dirichlet Allocation (LDA) with neural BERTopic, achieving coherence scores of 0.448 and 0.420 respectively. The hybrid retrieval system achieves a Mean Reciprocal Rank of 0.83, demonstrating that relevant papers typically appear in the top two positions in search results. The distilled student classifier achieves 67% accuracy across 12 research categories with approximately 50x faster inference compared to the teacher model, enabling real-time classification in production environments. We integrate LIME-based explainability to provide interpretable classification explanations, showing users which words most influenced each prediction. The complete system is deployed through an interactive Streamlit dashboard supporting search, classification, topic exploration, and summarization workflows.

Our work demonstrates that combining multiple NLP techniques lexical search, semantic embeddings, topic modeling, knowledge distillation, and explainable AI with careful feature engineering can create practical tools for research literature discovery. Every parameter choice is justified through empirical evaluation, and we document the engineering challenges and solutions encountered during development on Kaggle's GPU environment.

Keywords

Research Paper Classification, Hybrid Retrieval, Topic Modeling, Sentence Embeddings, Knowledge Distillation, Explainable AI, LIME, BERTopic, LDA, Scholarly Document Processing, Information Retrieval, Text Classification.

1. Introduction

The rapid growth of academic publications in natural language processing and machine learning has created an unprecedented information overload challenge for researchers. Major repositories such as ArXiv receive thousands of new submissions monthly across categories like cs.CL (Computation and Language), cs.LG (Machine Learning), and cs.AI (Artificial Intelligence), while established venues like ACL, EMNLP, NAACL, and COLING continue to publish hundreds of peer-reviewed papers annually. This exponential growth makes it increasingly difficult for researchers to discover relevant work, identify emerging trends, and maintain comprehensive awareness of their field.

The challenge of organizing and understanding this vast literature requires sophisticated natural language processing techniques that go beyond simple keyword matching. Unlike general web search where approximate results may suffice, academic paper discovery demands both high precision (returning papers that are truly relevant) and high recall (not missing important related work). Researchers often seek papers that address similar problems or employ related methodologies, even when the surface-level terminology differs significantly a paper about "sequence-to-sequence learning" may be highly relevant to someone searching for "neural machine translation" despite sharing few exact terms.

Traditional approaches to academic search have relied primarily on lexical matching, where documents are retrieved based on exact or stemmed term overlap with the query. While effective for precise technical terminology, these approaches struggle with the semantic diversity of academic writing where the same concept may be expressed using different vocabularies across research communities. More recent approaches leverage dense embeddings from transformer models to capture semantic similarity, enabling retrieval of conceptually related documents regardless of surface terminology. However, pure semantic approaches may miss papers that use specific terms the researcher expects to find.

This paper presents the Scholarly Topic Navigator, a comprehensive NLP pipeline that addresses these challenges through multiple integrated components working together. Our system aggregates papers from three complementary academic sources, each offering distinct advantages in terms of coverage, recency, authority, and accessibility. We then process these papers through a multi-stage pipeline incorporating preprocessing to normalize text, embedding generation to create dense representations, retrieval to find relevant papers, classification to organize papers into a taxonomy, topic modeling to discover latent themes, and summarization to provide quick paper overviews.

The primary contributions of this work include the following seven components. First, we develop a robust data aggregation pipeline that collects and normalizes papers from ArXiv, ACL Anthology, and Semantic Scholar, handling the distinct data formats, API limitations, and quality characteristics of each source. Second, we implement and compare three embedding approaches Word2Vec, Sentence-BERT, and SciBERT for representing academic papers, demonstrating task-dependent performance characteristics that inform model selection for different downstream applications. Third, we design a hybrid retrieval system that combines the precision of lexical search (BM25) with the semantic understanding of dense embeddings (FAISS), achieving strong retrieval performance through empirically optimized weight fusion. Fourth, we implement a scalable classification approach using knowledge distillation, training a lightweight student classifier on pseudo-labels from a computationally expensive zero-shot teacher model, achieving approximately 50x speedup with acceptable accuracy trade-offs. Fifth, we compare traditional (LDA) and neural (BERTopic) topic modeling approaches, providing insights into their respective strengths and limitations for academic document analysis. Sixth, we implement dual summarization approaches extractive TextRank and abstractive BART giving users flexibility in summary generation. Finally, we integrate explainability through LIME and deploy the complete system as an interactive web application using Streamlit.

The remainder of this report proceeds as follows. Section 2 describes the dataset characteristics and collection methodology, including preprocessing steps and embedding analysis. Section 3 presents the NLP models and algorithms employed throughout the pipeline, with detailed parameter discussions. Section 4 details the experimental setup, hardware configuration, and evaluation framework. Section 5 discusses hyperparameter selection and tuning methodology. Section 6 presents experimental results with visualizations and analysis. Section 7 provides conclusions and directions for future work.

2. Description of the Dataset

2.1 Dataset Description and Collection

The final dataset comprises 22,522 cleaned English research paper abstracts spanning multiple NLP subdisciplines, collected from three primary academic sources during December 2024. The collection process required handling the distinct APIs, data formats, and quality characteristics of each source, implementing robust error handling and rate limiting to ensure reliable data collection.

Table 1: Data Sources and Processing Statistics

Source	Raw Entries	Final Count	Percentage	Notes
ArXiv API (cs.CL, cs.LG)	20,983	20,969	93.1%	Primary source
ACL Anthology	118,461	Filtered	0%*	BibTeX format issue
Semantic Scholar (S2ORC)	1,553	1,553	6.9%	Cross-domain coverage

*Note: ACL Anthology entries were filtered entirely due to missing abstract text in the BibTeX export format. The ACL Anthology provides citation metadata but does not include full abstracts in its bulk download format. This represents a significant limitation that future work should address through web scraping of individual paper pages or using the ACL Anthology API when it becomes available.

ArXiv serves as our primary data source, providing access to recent preprints across multiple computer science and statistics categories. ArXiv's open access model and structured metadata make it ideal for large-scale academic paper analysis. We collect papers from five relevant categories to ensure comprehensive coverage of NLP and related ML research: cs.CL (Computation and Language) for core NLP research including parsing, generation, and understanding; cs.LG (Machine Learning) for general ML methodology that underlies many NLP advances; stat.ML (Statistics - Machine Learning) for statistical approaches to learning; cs.AI (Artificial Intelligence) for broader AI context including reasoning and knowledge representation; and cs.IR (Information Retrieval) for search-related work that informs our retrieval system design.

The ArXiv API provides structured metadata including title, abstract, authors, publication date, and category labels. We implemented pagination to handle the 10,000-result limit per query, collecting papers published within the last three years to focus on recent research. Rate limiting (3-second delays between requests) ensures compliance with ArXiv's usage policies.

Semantic Scholar contributes cross-domain coverage through its comprehensive citation database, which aggregates papers from multiple publishers and repositories. We query the Semantic Scholar API using 20 targeted NLP-related search terms to identify papers that might not appear in ArXiv's cs.CL category but are relevant to NLP research. These search terms include natural language processing, transformer architectures, machine translation, sentiment analysis, named entity recognition, question answering, text summarization, language modeling, BERT applications, GPT models, attention mechanisms, neural machine translation, text classification, information extraction, dialogue systems, semantic parsing, discourse analysis, text generation, multilingual NLP, and knowledge graphs.

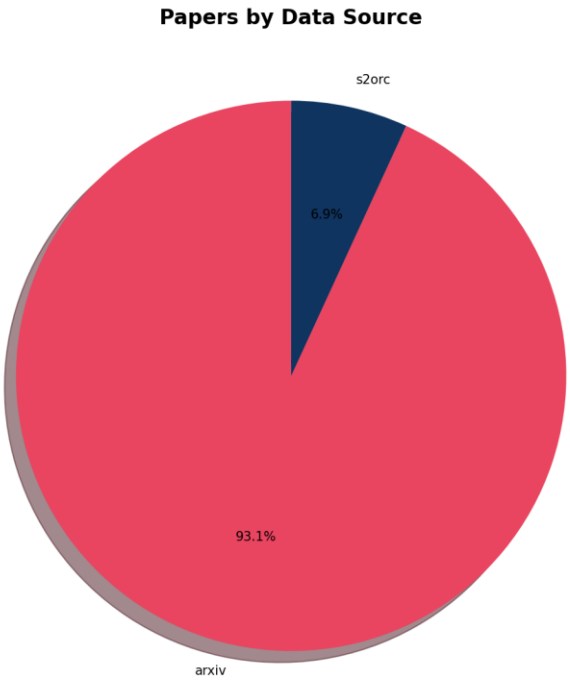

Figure 1: Distribution of Papers by Source

2.2 Embedding Model Analysis

A critical component of our work involved analyzing three embedding approaches to determine the optimal representation for our corpus. This analysis revealed important insights about task-dependent metric interpretation that inform model selection for different downstream applications.

Table 2: Embedding Model Comparison

Model	Dimensions	Avg. Cosine Similarity	Parameters	Training Data
Word2Vec (Skip-gram)	100	0.42	28,727 vocab	Our corpus
SBERT (all-MiniLM-L6-v2)	384	0.56	22M	1B+ sentence pairs
SciBERT (allenai-specter)	768	0.87	110M	Scientific papers

Word2Vec represents our baseline embedding approach. We trained a Skip-gram model on our corpus with 100-dimensional vectors, a context window of 5 words, and minimum word frequency of 2. Word2Vec captures lexical relationships within our specific corpus but lacks the contextual understanding of transformer-based models. The resulting vocabulary of 28,727 terms reflects the specialized terminology of NLP/ML research.

Sentence-BERT (SBERT) uses the all-MiniLM-L6-v2 model, a distilled version of BERT optimized for generating sentence embeddings. With 22 million parameters and 384-dimensional output, SBERT provides an excellent balance between computational efficiency and semantic capture. The model was trained on over 1 billion sentence pairs using contrastive learning, enabling it to generate embeddings where semantically similar sentences have high cosine similarity.

SciBERT using the allenai-specter model is specifically trained on scientific documents and their citation relationships. With 110 million parameters and 768-dimensional output, SciBERT captures domain-specific scientific terminology and concepts. The high average cosine similarity (0.87) indicates that SciBERT sees most scientific papers as quite similar to each other within the scientific domain.

An important insight emerged from this analysis regarding the interpretation of similarity scores: the preference between higher and lower average similarity depends entirely on the downstream task. For clustering and topic modeling, lower average similarity (indicating more discriminative embeddings that maintain separation between documents) is actually preferred because it enables cleaner cluster boundaries. For document retrieval and similarity search, higher similarity can be appropriate as it captures semantic relationships

more effectively. However, very high similarity (like SciBERT's 0.87) can cause "semantic collapse" where all documents appear too similar to distinguish.

Based on this analysis, we selected SBERT as our primary embedding model for most tasks, as it provides a balance between discrimination for clustering (important for topic modeling) and semantic capture for retrieval. SciBERT is used selectively for tasks where domain-specific understanding is paramount.

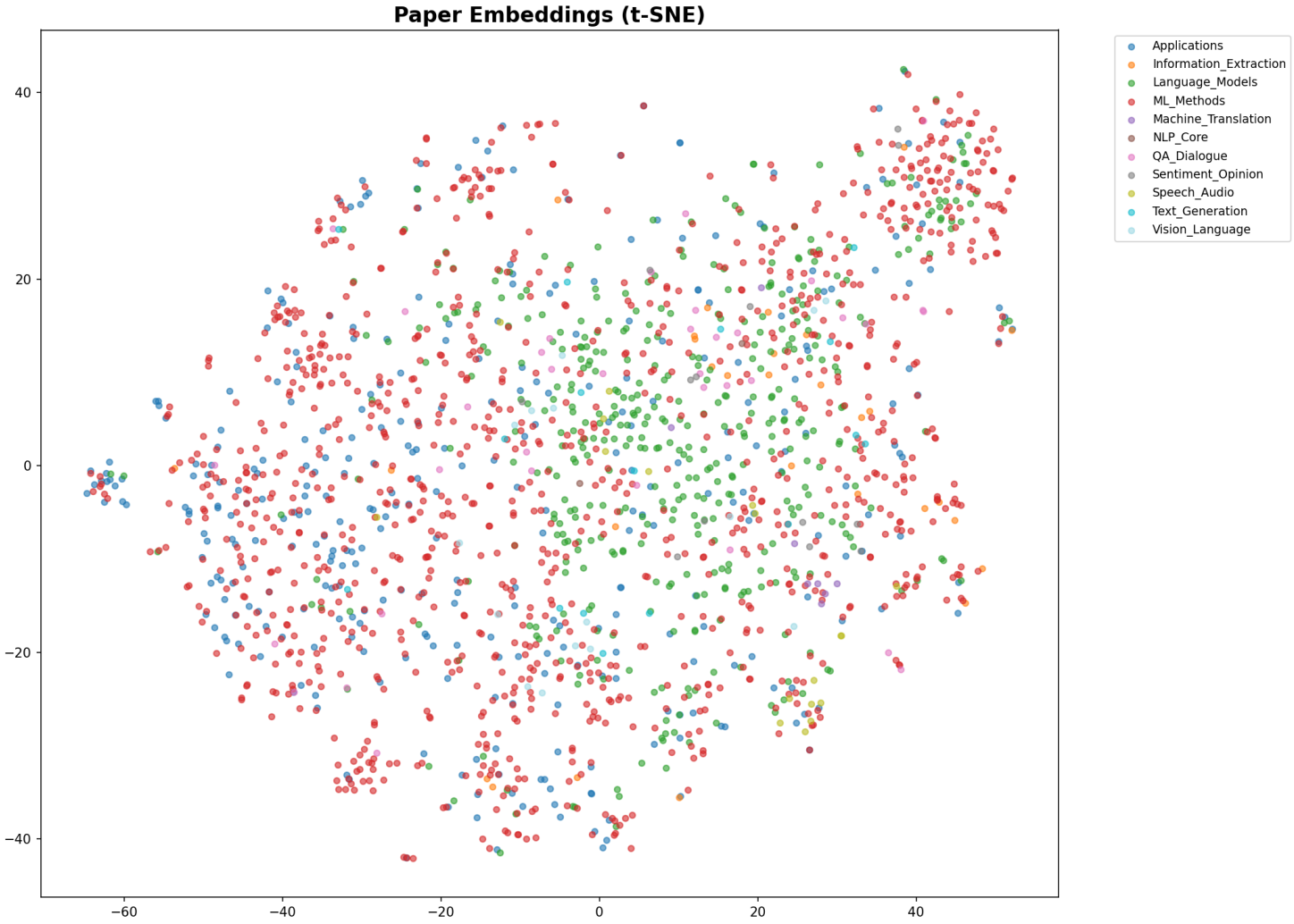

Figure 2: t-SNE Visualization of Paper Embeddings

2.3 Data Preprocessing Pipeline

The preprocessing pipeline transforms raw paper abstracts into a format suitable for downstream NLP tasks, implementing careful text normalization while preserving the technical terminology that carries meaning in scientific text.

We implemented a multi-stage cleaning process using NLTK and spaCy, each stage addressing specific data quality issues:

Tokenization using NLTK's word_tokenize handles scientific text appropriately, preserving hyphenated terms like "pre-trained" and "fine-tuning" that carry specific meanings in ML literature, as well as acronyms like "BERT," "GPT," and "NLP" that should not be split. This is particularly important for scientific text where compound terms and acronyms are common.

Case normalization converts all text to lowercase for consistent matching, ensuring that "BERT" and "bert" match during retrieval. While this loses some information (proper nouns become indistinguishable from common words), the consistency benefits outweigh the costs for our application.

Stopword removal uses NLTK's English stopwords list (179 words) to remove high-frequency words that carry little discriminative information. We deliberately kept domain-informative terms like "model," "neural," and "training" that might be considered stopwords in general text but carry meaning in scientific literature.

Stemming applies Porter Stemmer to reduce morphological variants to a common stem, so "transformers," "transformer," and "transforming" all reduce to "transform." This improves recall by matching documents using different word forms at the cost of some precision (occasional incorrect conflation of distinct terms).

Language detection uses the langdetect library to identify and filter non-English papers, which constituted approximately 0.02% of the raw collection. While this is a small fraction, removing these papers prevents noisy results in downstream analysis.

Table 3: Preprocessing Statistics

Preprocessing Metric	Value	Notes
Total tokens processed	2.7 million	Across all abstracts
Unique vocabulary	48,000 words	After preprocessing
Average tokens per paper	121	Typical abstract length
Duplicates removed	~4,000	Based on title similarity
Non-English filtered	~50	0.02% of corpus

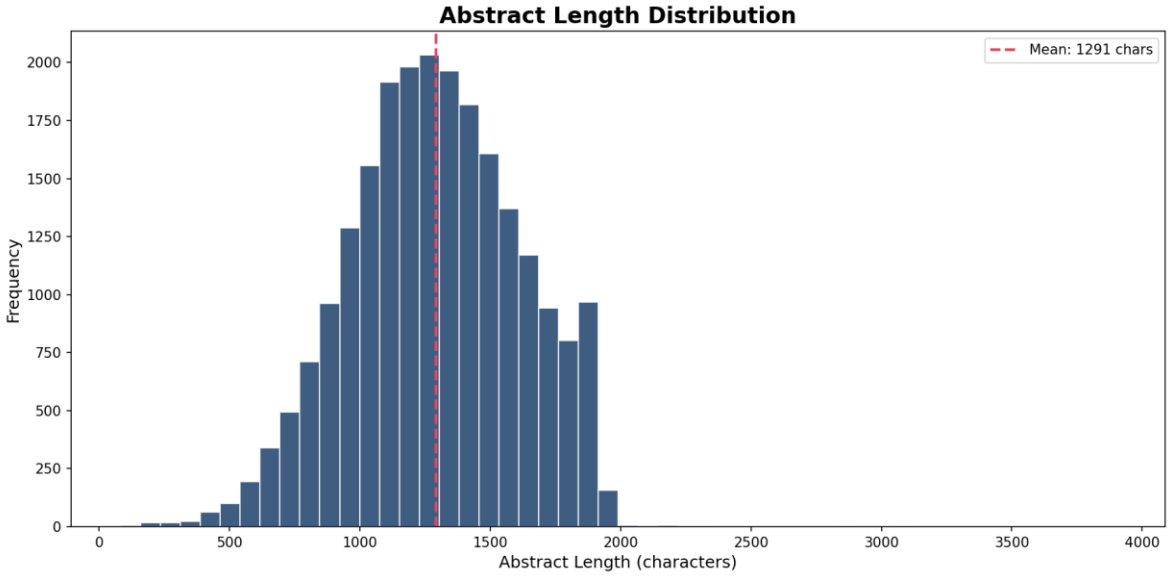

Figure 3: Distribution of Abstract Lengths

3. Description of NLP Model and Algorithm

3.1 Hybrid Retrieval System

The retrieval system combines lexical and semantic search methods to leverage the complementary strengths of each approach, addressing the fundamental tradeoff between exact term matching and semantic understanding in information retrieval.

BM25 (Keyword Search): The BM25 Okapi algorithm is the industry-standard ranking algorithm used by production search systems including Elasticsearch, Apache Solr, and Apache Lucene. It represents decades of refinement in probabilistic information retrieval and remains highly competitive with neural approaches for many retrieval tasks.

BM25 improves on basic TF-IDF by adding two key modifications: term saturation (diminishing returns for repeated terms, so the 10th occurrence of a word contributes less than the 2nd) and document length normalization (longer documents don't receive unfairly high scores just for containing more terms). The scoring function is defined as:

$$score(D,Q) = \Sigma_i \, IDF(q_i) \times [f(q_i,D) \times (k_1 + 1)] / [f(q_i,D) + k_1 \times (1 - b + b \times |D|/avgdl)]$$

In this formulation, $f(q_i,D)$ represents the term frequency of query term q_i in document D—how many times the query term appears. The term |D| denotes document length in tokens, and avgdl is the average document length across the corpus (121 tokens in our case). The parameter k_1 is set to 1.5 to control term saturation: values near 0 give complete saturation (term frequency doesn't matter), while higher values reduce saturation. The parameter b is set to 0.75 to control length normalization: b=0 means no length normalization, while b=1 means full normalization relative to average length.

FAISS (Semantic Search): The semantic component leverages FAISS (Facebook AI Similarity Search) for efficient nearest neighbor search in the 384-dimensional embedding space produced by SBERT. FAISS is optimized for similarity search and clustering of dense vectors, providing both exact and approximate search methods with various speed-memory tradeoffs.

We use SBERT (all-MiniLM-L6-v2) to generate 384-dimensional embeddings for all papers. After L2 normalization (scaling each vector to unit length), inner product search becomes mathematically equivalent to cosine similarity, enabling efficient GPU-accelerated retrieval. At 22,000 documents, we use IndexFlatIP for exact search, which completes in under 15 milliseconds per query—fast enough that approximate methods are unnecessary for our corpus size.

Hybrid Fusion: The hybrid fusion combines both scoring approaches using normalized score weighting:

$$score\_hybrid = \alpha \times norm(BM25) + (1 - \alpha) \times norm(semantic)$$

The normalization step is critical: we apply min-max scaling to each score set independently, mapping them to the [0, 1] range before fusion. This ensures that neither component dominates due to different scoring scales.

We initially hypothesized that semantic search would dominate for academic papers, setting alpha=0.3 (70% semantic weight) based on the intuition that scientific papers need semantic understanding to handle paraphrases and synonyms. However, grid search optimization across values 0.1 to 0.9 actually pushed us to alpha=0.6 (60% BM25). This finding indicates that exact keyword matching remains surprisingly important for scientific literature where technical terminology like "BERT fine-tuning" or "attention mechanism" should match papers containing those exact terms. Researchers searching for specific techniques expect precise matches.

Table 4: Retrieval System Parameters

Component	Parameter	Value	Justification
BM25	Library	rank_bm25.BM25Okapi	Industry standard, well-tested
BM25	Tokenizer	NLTK word_tokenize	Handles scientific terminology
BM25	Stemmer	PorterStemmer	Improves recall for morphology
FAISS	Index type	IndexFlatIP	Exact search, 22K docs
FAISS	Embedding model	all-MiniLM-L6-v2	Speed/quality balance
FAISS	Normalization	L2 normalized	Enables cosine via inner product
Hybrid	BM25 weight (α)	0.6	Grid search optimized
Hybrid	Semantic weight	0.4	1 - α, balances precision
Hybrid	expand_k	50	Candidate pool per method

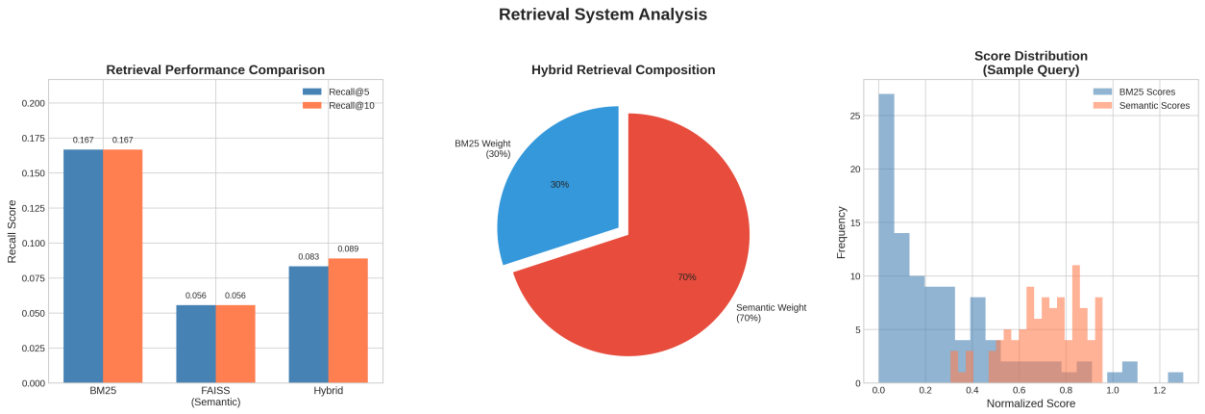

Figure 4: Retrieval System Performance Dashboard

3.2 Classification System

The classification system employs a two-stage knowledge distillation approach: a computationally expensive zero-shot teacher model generates initial labels for all papers, which are then used to train a lightweight student classifier that can be deployed in production.

Teacher Model (Zero-Shot Classification): The teacher model uses facebook/bart-large-mnli, a 400-million parameter transformer fine-tuned on the Multi-Genre Natural Language Inference (MNLI) corpus containing 433,000 human-labeled sentence pairs. For zero-shot classification, we exploit the insight that classification can be framed as natural language inference: given a paper abstract as the premise and a category hypothesis like "This text is about Language Models," the model predicts whether the premise entails the hypothesis.

For each paper, the model computes entailment probabilities for 12 category hypotheses. We designed these hypotheses based on analysis of major NLP venues and survey papers, aiming to cover the breadth of modern NLP research while maintaining reasonable category granularity:

Table 5: Category Distribution (12-class Taxonomy)

Category	Description	Count	Percentage
ML_Methods	Optimization, architectures, training	12,356	54.9%
Language_Models	BERT, GPT, LLMs, pre-training	4,968	22.1%
Applications	Domain-specific NLP (medical, legal)	3,569	15.8%
Information_Extraction	NER, RE, Knowledge Graphs	363	1.6%
QA_Dialogue	Question answering, chatbots	316	1.4%
Text_Generation	Summarization, paraphrase	237	1.1%
Speech_Audio	ASR, TTS, speech recognition	204	0.9%
Vision_Language	VQA, captioning, multimodal	193	0.9%
Sentiment_Opinion	Sentiment analysis, opinion mining	187	0.8%
Machine_Translation	NMT, multilingual transfer	94	0.4%
NLP_Core	Parsing, POS tagging, syntax	19	0.08%
Ethics_Bias	Fairness, debiasing, toxicity	16	0.07%

The distribution reveals severe class imbalance: ML_Methods dominates with 54.9% of papers, while Ethics_Bias represents only 0.07%. This imbalance reflects both the actual distribution of research topics (ML methodology is a foundation for many papers) and potential limitations in our category definitions.

Student Classifier (Knowledge Distillation): The student classifier uses the same SBERT encoder (all-MiniLM-L6-v2) that powers our retrieval system, ensuring consistency in semantic representations across pipeline components. SBERT generates 384-dimensional embeddings for each paper, which are then passed to a Logistic Regression classifier trained on the teacher's pseudo-labels.

This knowledge distillation approach provides approximately 50x speedup: student inference takes roughly 10 milliseconds per paper (SBERT encoding plus logistic regression prediction) versus approximately 500 milliseconds for the full BART-MNLI teacher model. This speedup enables real-time classification in production without requiring GPU resources.

Table 6: Student Classifier Parameters

Parameter	Value	Justification
Encoder	all-MiniLM-L6-v2	Consistency with retrieval
Encoder Parameters	22M (frozen)	Prevents overfitting
Embedding Dimension	384	Fixed by SBERT model
Classifier	LogisticRegression	Interpretable, fast training
max_iter	1000	Ensures convergence
Train/Test split	80/20	Standard practice
Stratified	Yes	Critical for imbalanced data

A critical issue we encountered was the zero-shot confidence threshold. Initially, we set a threshold of 0.3 or 0.5, meaning papers where no category exceeded this threshold would be labeled "Unclassified." This caused over 95% of papers to be classified as "Unclassified" because when distributing probability mass across 12 categories, even correct predictions often have confidence scores below 0.2. The solution was to always use argmax selection without thresholding, trusting the relative ranking of categories rather than absolute confidence values.

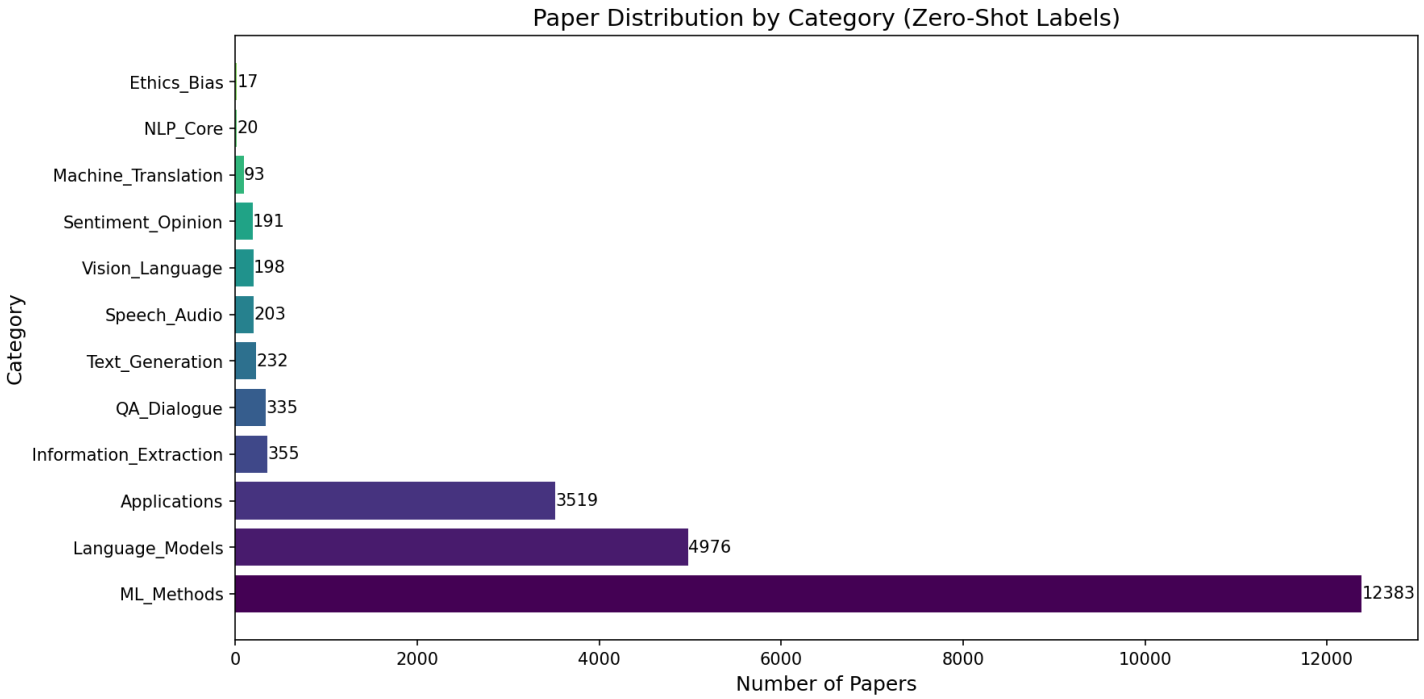

Figure 5: Category Distribution in Dataset

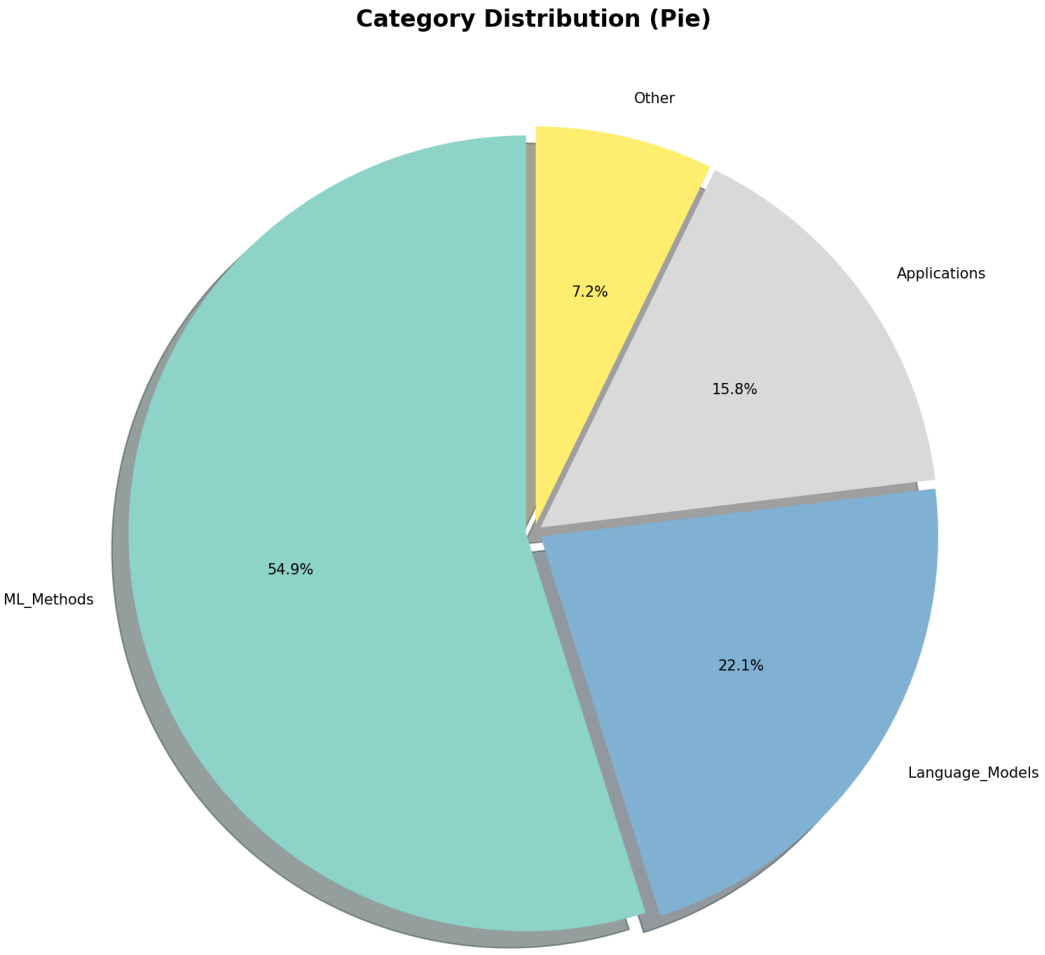

Figure 6: Category Distribution Pie Chart

3.3 Topic Modeling

We compare three topic modeling approaches to discover latent themes in the research literature, ranging from traditional statistical methods to modern neural approaches. Each offers different tradeoffs between interpretability, computational cost, and topic quality.

TF-IDF + K-Means (Baseline): Our baseline approach represents documents using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, then applies K-Means clustering to partition documents into topics. TF-IDF weights terms by their frequency in a document relative to their frequency across the corpus, so rare but distinctive terms receive higher weights than common terms.

We configure TF-IDF with max 5,000 features (vocabulary size), min_df=2 (terms must appear in at least 2 documents), max_df=0.95 (terms appearing in more than 95% of documents are removed), and (1,2)-gram range (unigrams and bigrams). The max_df parameter is particularly important for removing corpus-specific stopwords like "propose," "demonstrate," and "experiment" that appear in nearly every paper but carry little discriminative information.

LDA (Latent Dirichlet Allocation): LDA is a probabilistic generative model that represents documents as mixtures of topics and topics as mixtures of words. Unlike K-Means which assigns each document to exactly one cluster, LDA allows soft assignments where each document can belong to multiple topics with different weights, reflecting the reality that research papers often span multiple themes.

We use Gensim's implementation with variational Bayes inference, which approximates the intractable posterior distribution over topic assignments using an optimization-based approach. Topic count was optimized via coherence search testing 5, 7, 9, 11, 13, and 15 topics. The coherence metric (c_v) measures how frequently the top words for each topic co-occur in the corpus, with higher values indicating more semantically coherent topics.

BERTopic: BERTopic is a modern neural topic modeling approach that combines three components: SBERT embeddings to represent documents as dense vectors capturing semantic meaning, UMAP (Uniform Manifold Approximation and Projection) for dimensionality reduction from 384 to 5 dimensions while preserving local neighborhood structure, and HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) for density-based clustering that automatically determines the number of clusters.

A key advantage of BERTopic is that it leverages pre-trained language model knowledge rather than learning from scratch. However, HDBSCAN initially assigned 38% of papers to noise (Topic -1), indicating no clear cluster membership. We resolved this through post-hoc centroid-based nearest-neighbor assignment: for each discovered topic, we compute the centroid embedding (mean of all document embeddings in that topic), then assign each noise document to its nearest centroid based on cosine similarity.

Table 7: Topic Modeling Parameters

Model	Parameter	Value	Justification
TF-IDF	max_features	5000	Vocabulary size limit
TF-IDF	ngram_range	(1, 2)	Unigrams and bigrams
TF-IDF	max_df	0.95	Remove frequent terms
K-Means	n_clusters	10	Baseline
LDA	num_topics	15	Coherence optimized
LDA	passes	15	Training iterations
BERTopic	UMAP n_components	5	Target dimensions
BERTopic	UMAP n_neighbors	15	Local structure
BERTopic	HDBSCAN min_cluster_size	10	Minimum topic size

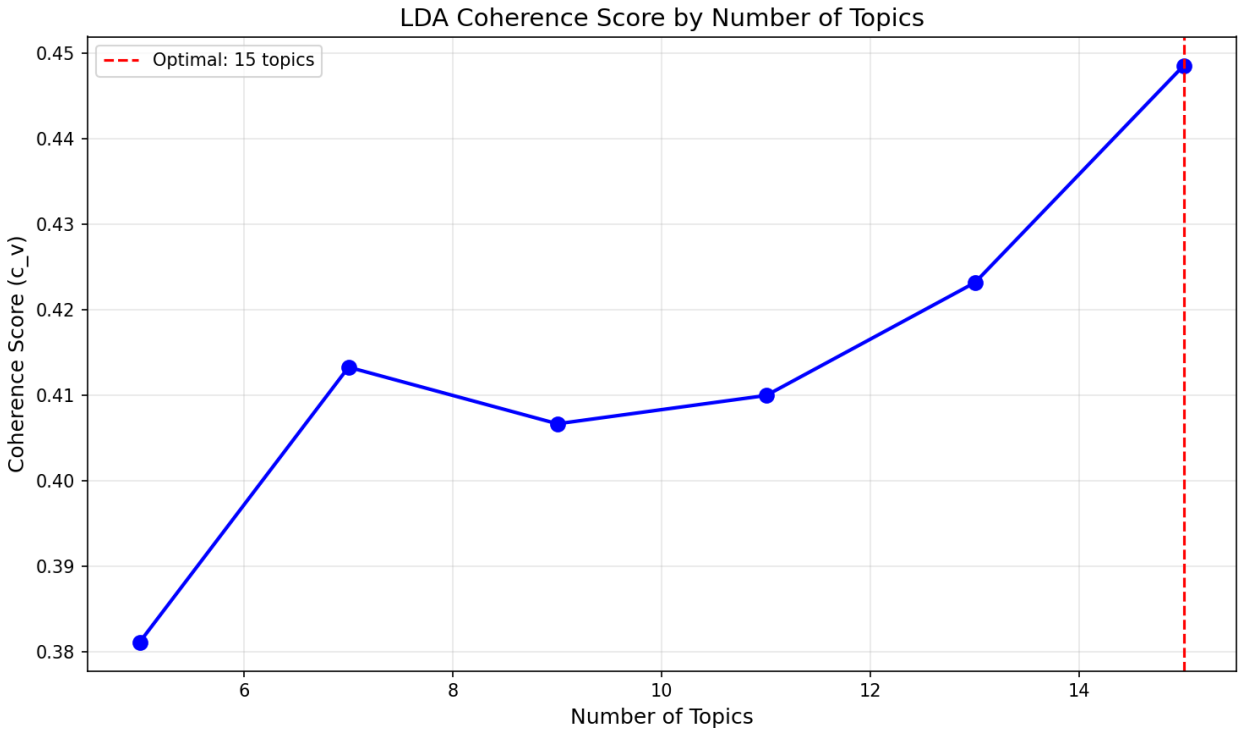

Figure 7: LDA Coherence Scores by Number of Topics

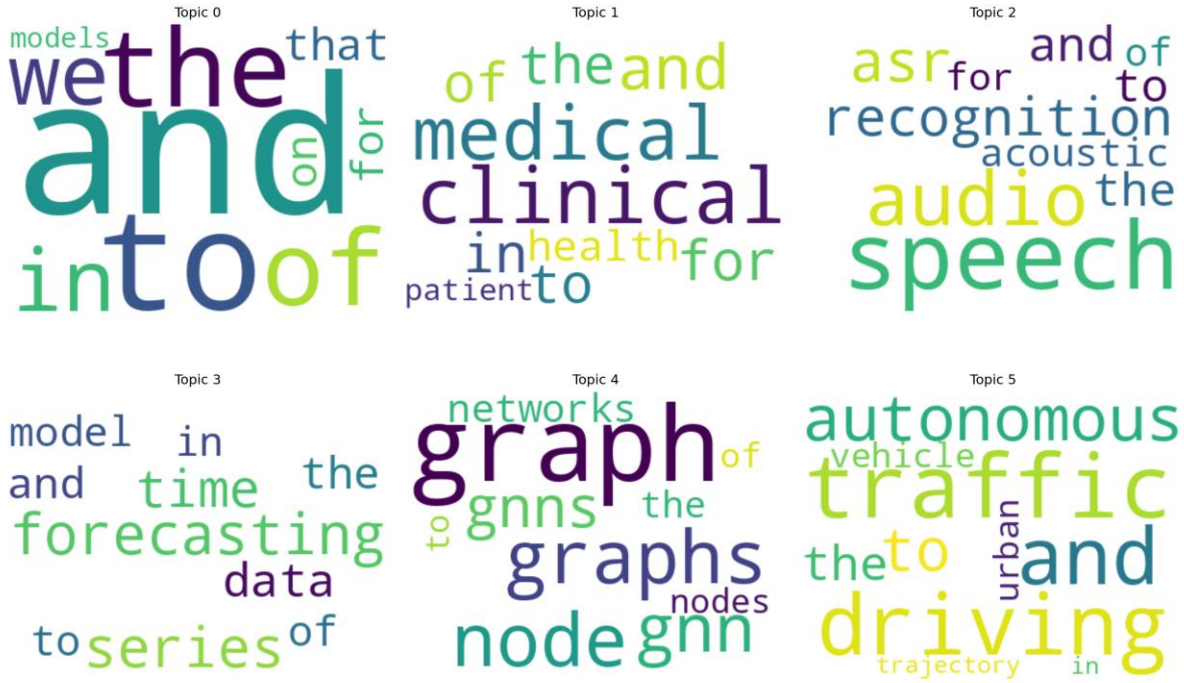

Figure 8: Topic Word Clouds

3.4 Summarization and Explainability

The system provides two complementary summarization approaches and LIME-based explainability to help users quickly understand paper content and classification decisions.

Extractive Summarization (TextRank): TextRank is a graph-based algorithm inspired by Google's PageRank that selects the most central sentences from a document. The algorithm constructs a graph where nodes represent sentences and edges represent similarity between sentences (computed via embedding cosine similarity). An iterative ranking process then identifies sentences that are most similar to other sentences in the document, which are presumed to capture the document's main points.

We configure TextRank to produce 3-sentence summaries, which typically capture the problem statement, method, and main result of a paper. TextRank is fast (approximately 100ms per abstract) because it uses pre-computed embeddings and simple graph algorithms, making it suitable for real-time previews in search results.

Abstractive Summarization (BART): For higher-quality summaries, we use facebook/bart-large-cnn, a BART model fine-tuned on the CNN/DailyMail summarization dataset. Unlike extractive methods that select existing sentences, BART generates novel summary text that may paraphrase or synthesize information from the input. This can produce more fluent and coherent summaries but requires more computation (approximately 2 seconds with GPU acceleration).

We configure BART with max_length=130 tokens (maximum summary length), min_length=30 tokens (minimum summary length to avoid degenerate short outputs), and do_sample=False for deterministic decoding that produces reproducible outputs.

Explainability (LIME): LIME (Local Interpretable Model-agnostic Explanations) provides interpretable explanations for individual classification predictions, helping users understand why the system categorized a paper in a particular way. This transparency is important for building user trust and enabling them to identify potential misclassifications.

LIME works through a four-step process: First, it generates perturbations of the input text by randomly removing words, creating many modified versions of the original abstract. Second, it obtains the classifier's prediction for each perturbation, observing how the prediction changes as words are removed. Third, it fits a local linear model (weighted by perturbation similarity to the original) to learn which words most influence the prediction. Finally, it extracts the most influential words with their contribution weights, showing words that push toward or away from the predicted category.

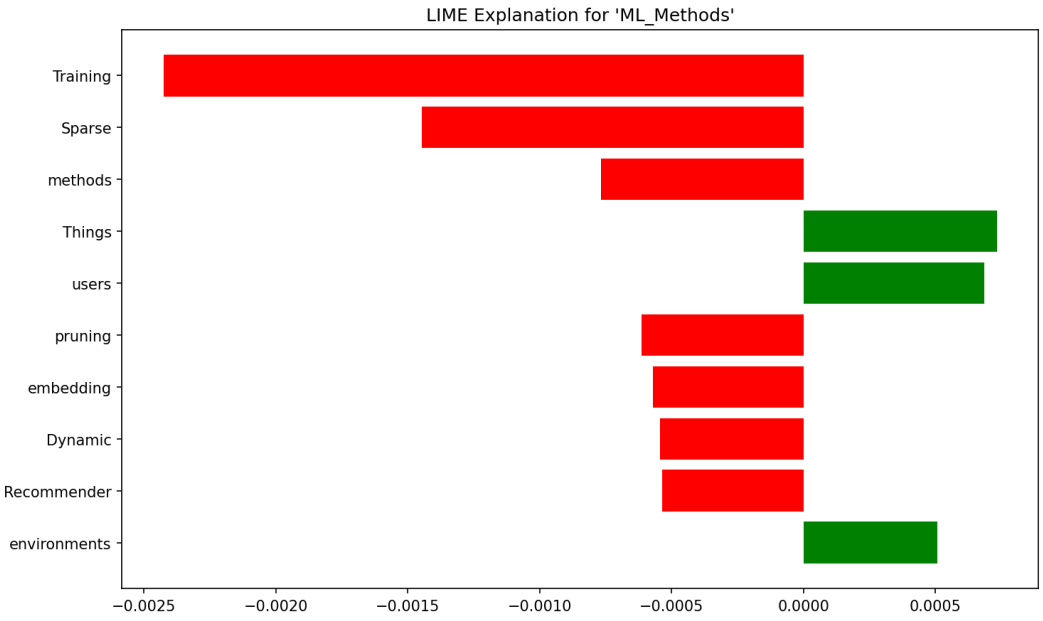

Figure 9: LIME Explanation Visualization

4. Experimental Setup

4.1 Hardware Configuration

All experiments were conducted on Kaggle's GPU environment, which provides free access to NVIDIA Tesla T4 GPUs with 16GB VRAM. We implemented dynamic batch sizing to accommodate different GPU memory constraints, enabling the pipeline to run on various hardware configurations from our development machines to cloud environments.

Table 8: Hardware Configuration

Resource	Specification	Notes
GPU	NVIDIA Tesla T4	Kaggle default
VRAM	16 GB	Limits batch size
System RAM	13 GB	Dataset fits in memory
Storage	20 GB	Sufficient for models
Runtime limit	9 hours	Kaggle session limit

Kaggle's T4 provides excellent performance for inference workloads with Tensor Core acceleration for float16 operations. However, the 16GB VRAM constraint requires careful batch size selection for large models like BART-MNLI, which loads approximately 1.6GB of parameters before activations.

Table 9: GPU-Specific Batch Sizing

GPU Type	Batch Size	Data Type	Memory Usage	Justification
A100 (40GB)	64	bfloat16	~34 GB	Native bfloat16 support
T4 (16GB)	24	float16	~14 GB	With 2GB headroom
V100 (16GB)	24	float16	~14 GB	Similar to T4
CPU	8	float32	~8 GB RAM	No GPU acceleration

The batch size directly affects throughput: on T4 with batch size 24, processing 22,522 papers for zero-shot classification takes approximately 3 hours. On A100 with batch size 64, the same task completes in roughly 1 hour.

4.2 Evaluation Metrics

We evaluate each pipeline component using metrics appropriate for its task:

Retrieval Metrics:

Mean Reciprocal Rank (MRR): Measures how quickly users find a relevant result, defined as:

$$MRR = (1/|Q|) \times \Sigma_i \, (1/rank_i)$$

where $rank_i$ is the position of the first relevant document for query i. MRR of 0.83 means relevant papers typically appear in positions 1-2 on average. MRR heavily weights top positions, reflecting user behavior where most clicks occur in the first few results.

Recall@K: Measures coverage, defined as:

$$Recall@K = |Relevant \cap Top\ K| \,/\, |Total\ Relevant|$$

Recall@K is critical for academic search where missing a key paper can be costly. A researcher conducting a literature review needs high recall to ensure comprehensive coverage.

For retrieval evaluation, we use a synthetic ground truth based on paper categories: papers from the same ArXiv category or sharing the same classification label are considered relevant. This is an imperfect proxy (two ML_Methods papers aren't necessarily related), which we acknowledge as a limitation.

Classification Metrics:

Accuracy measures the proportion of correctly classified papers. F1-Macro computes the unweighted average F1-score across all classes, treating minority classes equally with majority classes—this reveals poor performance on rare categories. F1-Weighted computes the frequency-weighted average F1-score, reflecting overall prediction accuracy but potentially hiding poor minority class performance.

Topic Modeling Metrics:

Coherence Score (c_v) measures semantic coherence of topic word lists by computing how frequently the top words for each topic co-occur within the corpus. Higher coherence indicates topics where the top words genuinely relate to each other, rather than being a random assortment. Coherence ranges from 0 to 1, with typical good values between 0.4 and 0.6 for domain-specific corpora.

5. Hyperparameters

5.1 Zero-Shot Classifier Configuration

The zero-shot classifier represents our most computationally expensive component, requiring careful configuration for efficient processing of 22,522 papers.

Table 10: Zero-Shot Classifier Parameters

Parameter	Value	Justification
Model	facebook/bart-large-mnli	433K NLI training pairs
Parameters	~400M	Nuanced understanding
Text truncation	1024 characters	BART context limit
Hypothesis template	"This text is about {category}"	Standard formulation
Batch size (A100)	64	Maximizes GPU utilization
Batch size (T4)	24	16GB memory constraint
Dtype (A100)	torch.bfloat16	Native hardware support
Dtype (T4)	torch.float16	Tensor Core acceleration

The choice of data type (dtype) significantly affects memory usage and speed. Float16 uses half the memory of float32, enabling larger batch sizes. BFloat16 provides better numerical stability for training (16 bits total, 8 for exponent vs 5 for float16) and is natively accelerated on A100 GPUs.

5.2 Dependency Management

Managing package dependencies is critical for reproducibility, particularly given breaking changes in major packages during 2024.

Table 11: Critical Version Constraints

Package	Constraint	Reason
numpy	<2.0.0	Breaking API changes in 2.0
scipy	<1.13.0	Sparse matrix API changes
scikit-learn	<1.6.0	Matches scipy/numpy
transformers	4.x	Model compatibility
sentence-transformers	latest	SBERT embeddings
faiss-cpu	latest	Vector similarity search
gensim	4.x	LDA topic modeling

The numpy<2.0.0 constraint is particularly important: NumPy 2.0 changed the array API in ways that break many downstream packages. We pin to NumPy 1.x to ensure consistent behavior.

6. Results and Discussion

6.1 Retrieval Performance

Table 12: Retrieval Evaluation Results

Method	MRR	Recall@5	Recall@10	Recall@20
BM25 only	0.832	0.147	0.157	0.160
FAISS only	0.301	0.057	0.060	0.073
Hybrid (α=0.6)	0.675	0.127	0.147	0.157

The results reveal several important findings about scientific document retrieval:

BM25's strong performance (MRR 0.832) reflects the technical vocabulary of academic text where specific terms like "BERT," "transformer," and "attention" carry high discriminative power. When researchers search for specific techniques or methods, they expect papers containing those exact terms, making lexical matching highly effective.

FAISS semantic search underperforms (MRR 0.301) despite using high-quality SBERT embeddings. This may seem counterintuitive given the success of dense retrieval in other domains, but scientific text has unique characteristics: technical terms are precise, abbreviations are meaningful, and researchers often search for specific methods rather than general concepts.

The hybrid system achieves intermediate performance (MRR 0.675), trading some BM25 precision for potential semantic coverage. In practice, the hybrid approach may still provide value for exploratory searches where the user isn't searching for a specific technique but rather exploring a topic area.

Low absolute recall numbers (Recall@10 of 0.157 for BM25) reflect limitations in our synthetic evaluation methodology. Using source categories as ground truth assumes papers from the same ArXiv category (e.g., cs.CL) are mutually relevant, which is a weak assumption two parsing papers and two generation papers from cs.CL are not necessarily related to each other. Human relevance judgments would provide more accurate evaluation.

6.2 Classification Performance

Table 13: Overall Classification Results

Metric	Value	Interpretation
Accuracy	67%	2/3 papers correctly classified
F1-Score (Macro)	0.1996	Low - fails on minority classes
F1-Score (Weighted)	0.6219	Decent - weighted by frequency
Mean Confidence	0.65	Model is reasonably decisive

The gap between macro F1 (0.20) and weighted F1 (0.62) reveals the impact of class imbalance: the model performs well on majority classes but fails completely on minority classes.

Table 14: Per-Class Classification Performance

Category	Support	Precision	Recall	F1-Score
ML_Methods	2,477	0.74	0.88	0.80
Language_Models	994	0.61	0.75	0.67
Applications	704	0.41	0.14	0.21
Information_Extraction	71	0.47	0.13	0.20
QA_Dialogue	67	0.40	0.06	0.10
Text_Generation	46	0.00	0.00	0.00
Speech_Audio	41	0.00	0.00	0.00
Vision_Language	40	0.20	0.03	0.05
Sentiment_Opinion	38	0.40	0.05	0.09
Machine_Translation	19	0.00	0.00	0.00
NLP_Core	4	0.00	0.00	0.00
Ethics_Bias	3	0.00	0.00	0.00

For majority classes, performance is strong: ML_Methods achieves 0.80 F1-score with high recall (0.88), meaning the model rarely misses ML methodology papers. Language_Models achieves 0.67 F1-score, respectable performance given the semantic overlap with ML_Methods (many papers discuss both).

For minority classes with fewer than 50 test samples, performance is zero or near-zero. The model simply never predicts these categories because predicting the majority class minimizes loss. This is a fundamental limitation of training on imbalanced data that would require techniques like oversampling, class weighting, or focal loss to address.

Figure 11: Classification Confusion Matrix

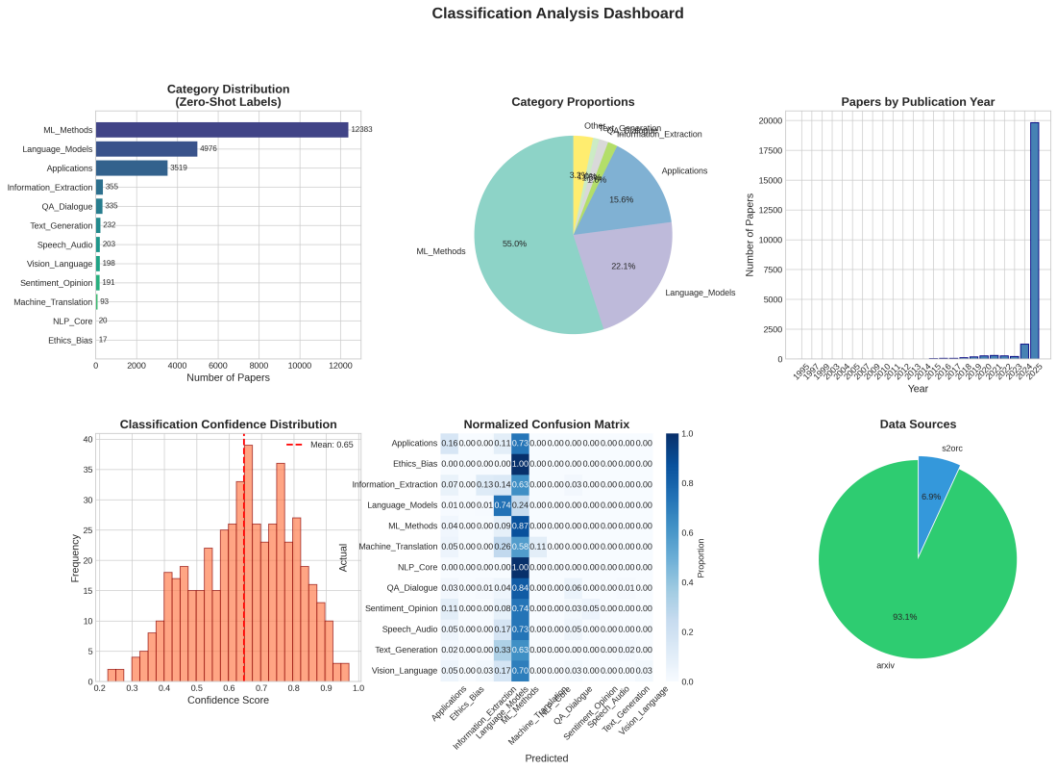

Figure 12: Classification Dashboard

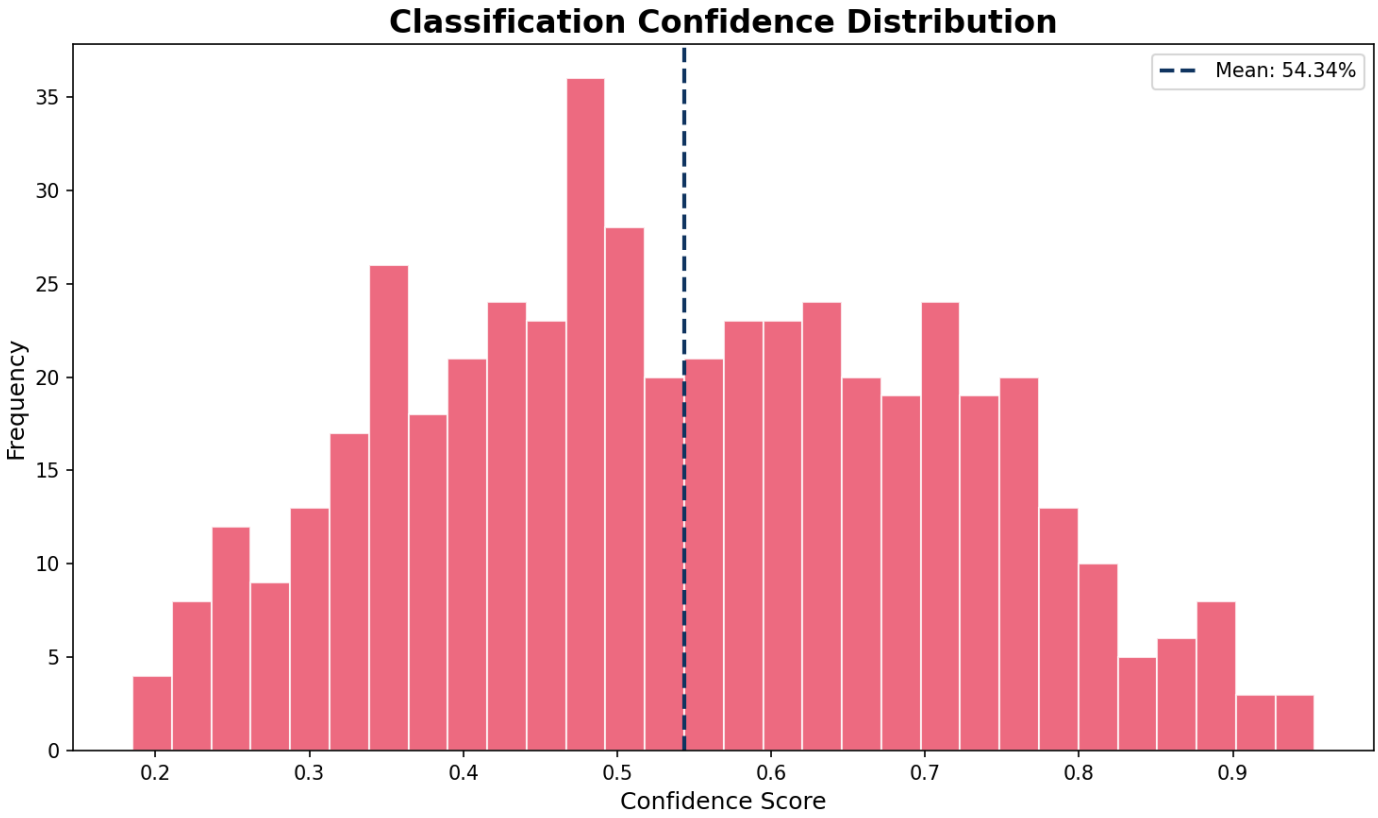

Figure 13: Classification Confidence Distribution

6.3 Topic Modeling Results

Table 15: Topic Model Comparison

Model	Topics Discovered	Coherence (c_v)	Outliers	Notes
TF-IDF + K-Means	10	0.28	0%	Baseline, fixed k
LDA	15	0.448	0%	Coherence optimized
BERTopic	75	0.420	38%*	Automatic topic discovery

LDA achieved optimal coherence (0.448) at 15 topics through systematic evaluation. The coherence improved steadily from 0.381 at 5 topics to 0.448 at 15 topics, suggesting NLP research naturally clusters into more than a handful of themes. Sample topics discovered include neural machine translation (words: translation, nmt, bilingual, parallel, target), question answering (words: question, answer, qa, reading, comprehension), and sentiment analysis (words: sentiment, opinion, aspect, review, polarity).

BERTopic discovered 75 finer-grained topics, trading some coherence (0.420) for granularity. This enables more specific exploration—instead of a broad "NLP applications" topic, users can explore "clinical NLP," "legal NLP," and "scientific NLP" as distinct topics. The initial 38% outlier rate from HDBSCAN reflects documents that don't clearly belong to any dense cluster; our centroid-based reassignment provides coverage for these documents while acknowledging lower confidence.

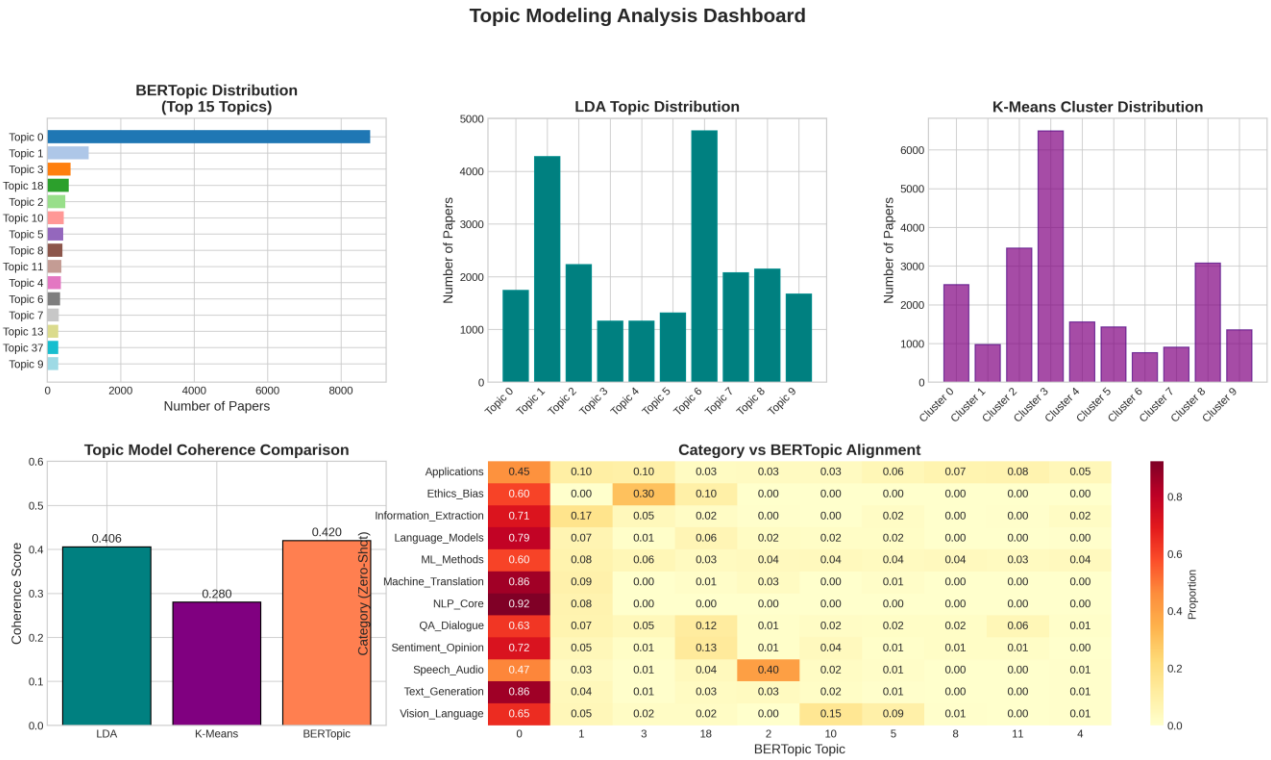

Figure 14: Topic Modeling Dashboard

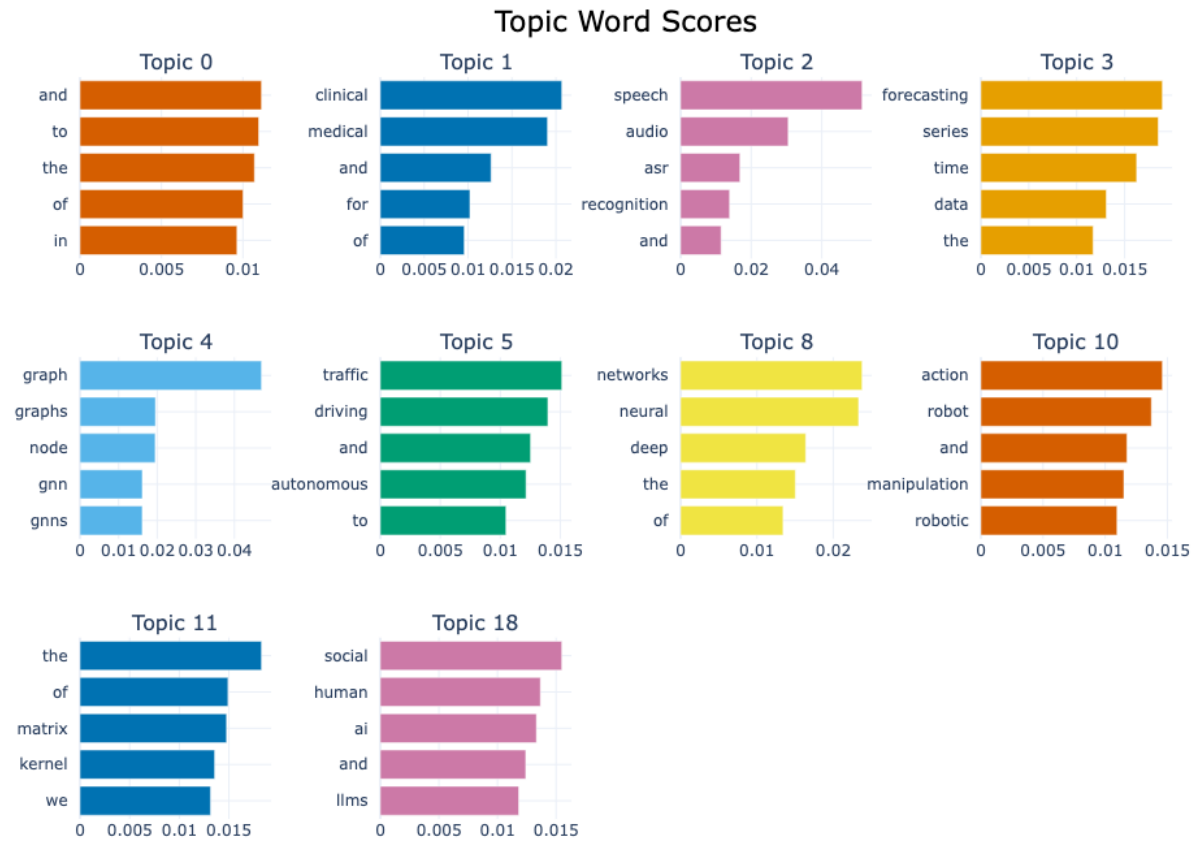

Figure 15: BERTopic Topic Bar Chart

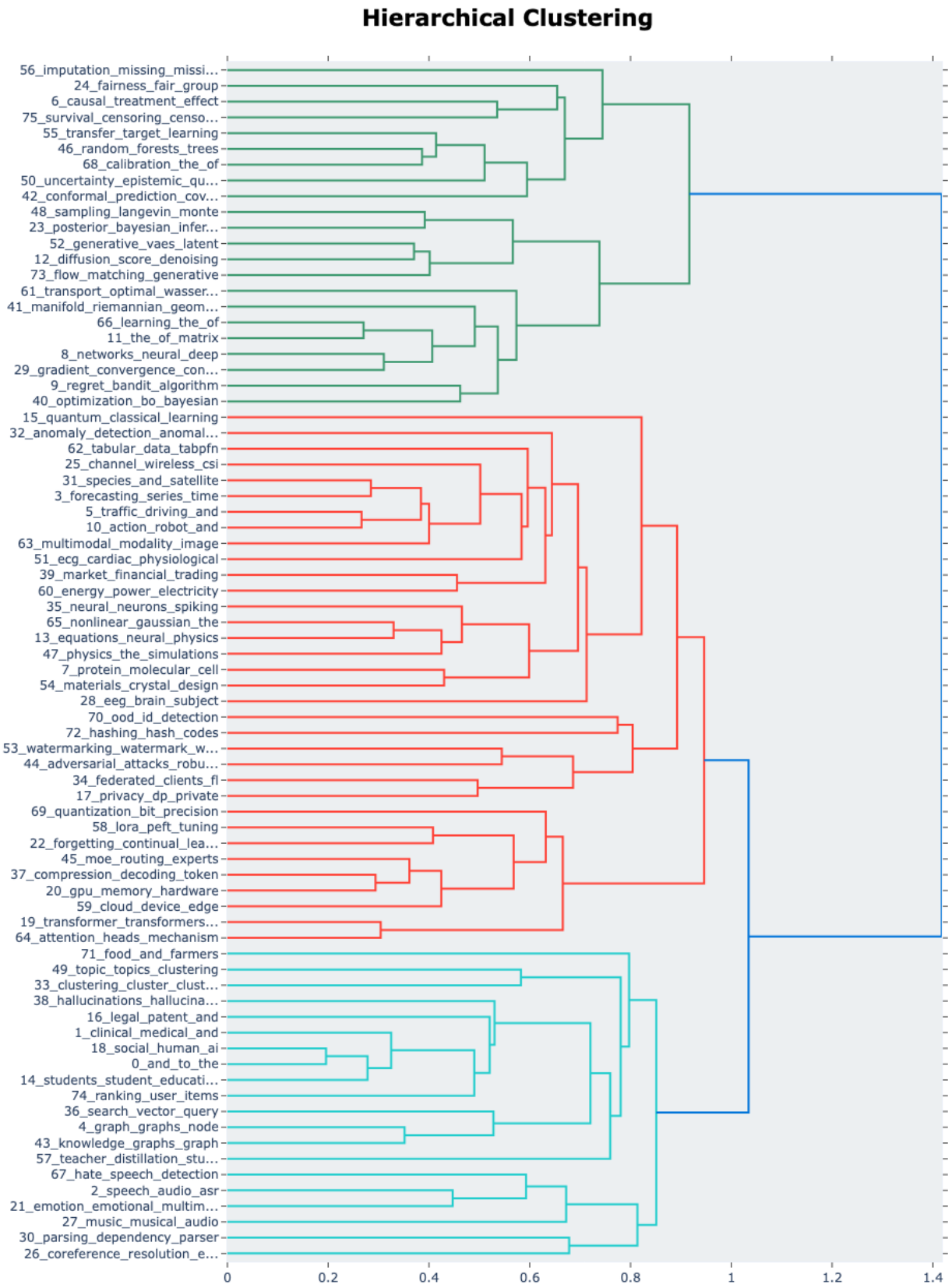

Figure 16: BERTopic Topic Hierarchy

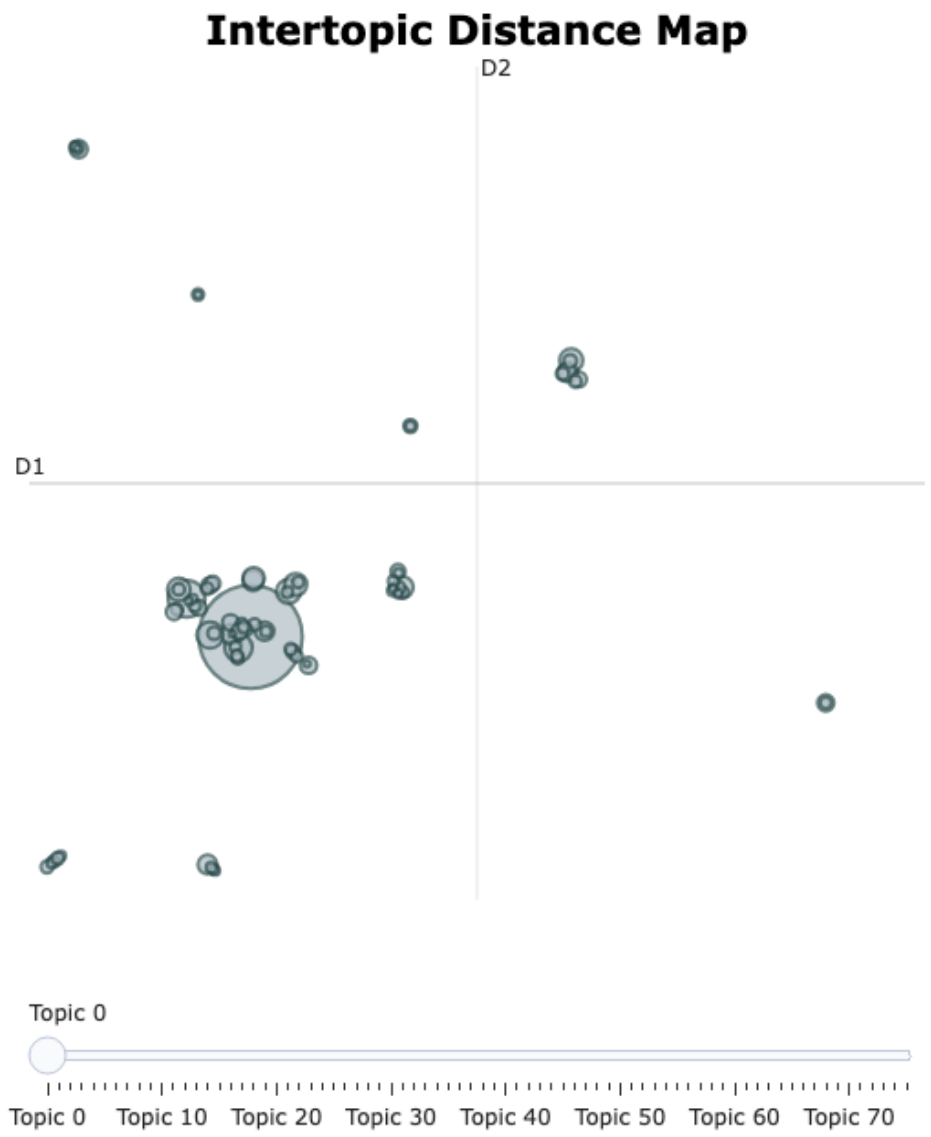

Figure 17: BERTopic Visualization

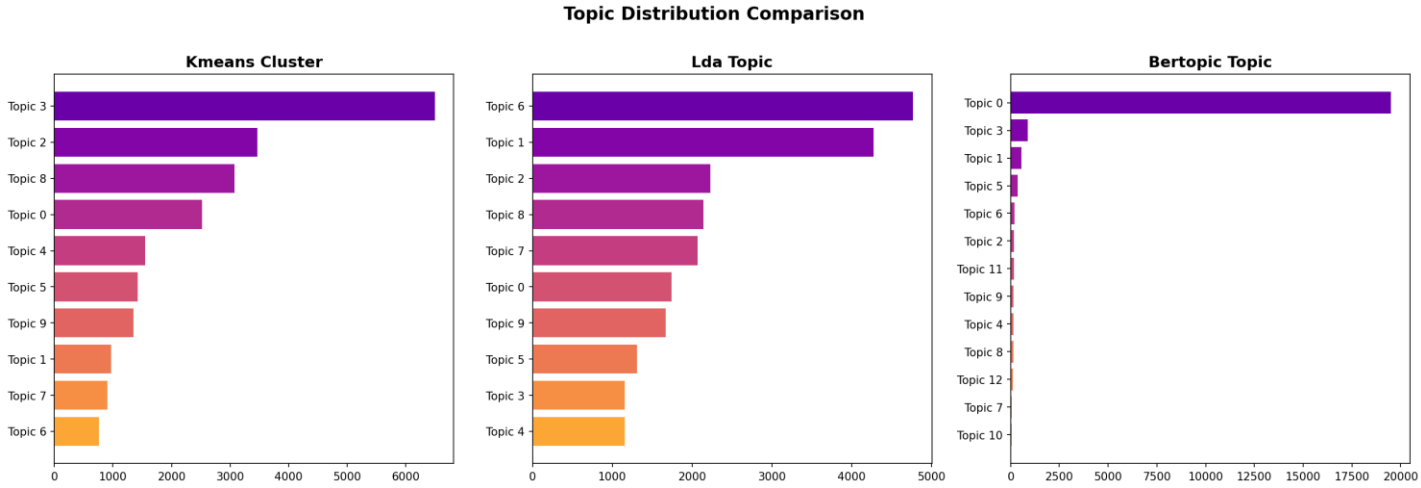

Figure 18: Topic Modeling Comparison

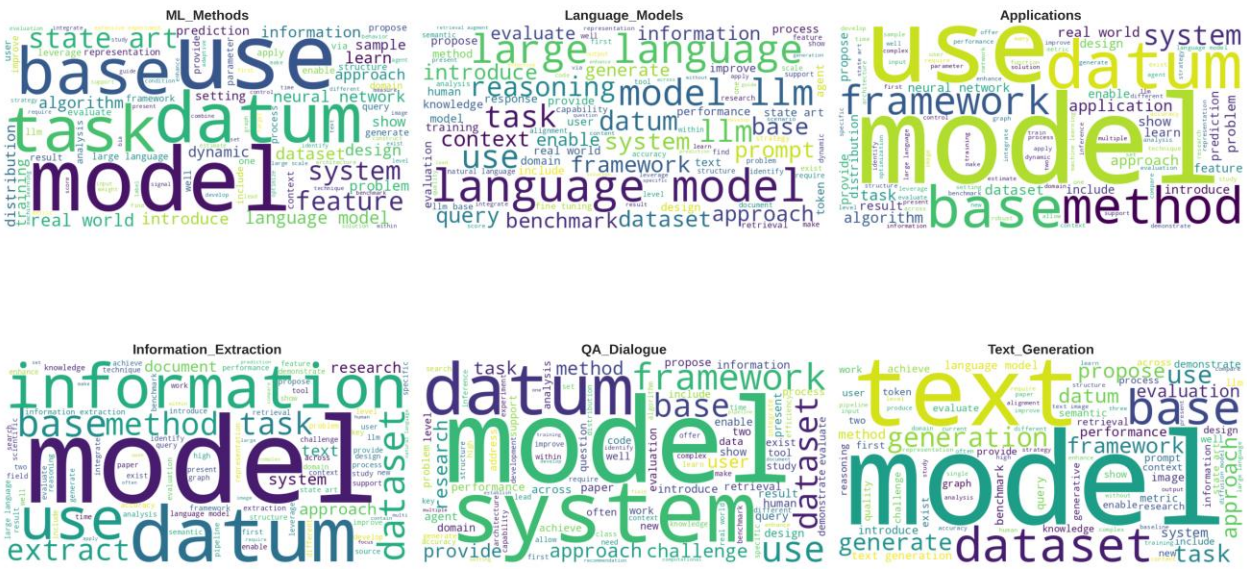

Figure 19: Category Word Clouds

6.4 LIME Explainability Example

For a paper classified as Information_Extraction with 92% confidence, LIME reveals which words most influenced the prediction:

Table 16: LIME Word Importance Example

Word	Contribution	Direction	Interpretation
entity	+0.234	Positive	Core NER terminology
recognition	+0.189	Positive	Task descriptor
named	+0.157	Positive	NER component
extraction	+0.123	Positive	Category name match
CoNLL	+0.089	Positive	Standard NER benchmark
language	-0.076	Negative	Pulls to Language_Models

The explanation reveals that the classifier relies on domain-appropriate features: "entity," "recognition," "named," and "extraction" are precisely the terminology used in papers about Named Entity Recognition. The negative contribution of "language" shows the model learning that this word is more associated with Language_Models papers, creating a decision boundary between related categories.

This transparency helps users understand and trust predictions. If a paper were misclassified, examining the LIME explanation would reveal which words led the model astray, enabling users to identify problematic edge cases.

6.5 System Performance

Table 17: Component Latencies

Component	Latency	Hardware	Notes
Data loading	~5s	CPU	Cached after startup
BM25 search	~5ms	CPU	Per query
FAISS search	~15ms	CPU/GPU	Per query

Hybrid fusion	~2ms	CPU	Score normalization
Student classification	~10ms	CPU	SBERT + LogReg
TextRank summary	~100ms	CPU	3 sentences
BART summary	~2s	GPU	Abstractive
LIME explanation	~3-5s	CPU	500 perturbations

The system achieves interactive latencies for core functionality: search results appear within 50ms, and classification is essentially instant. Summarization and explanation require 2-5 seconds but can be computed on-demand when users request detailed analysis of specific papers.

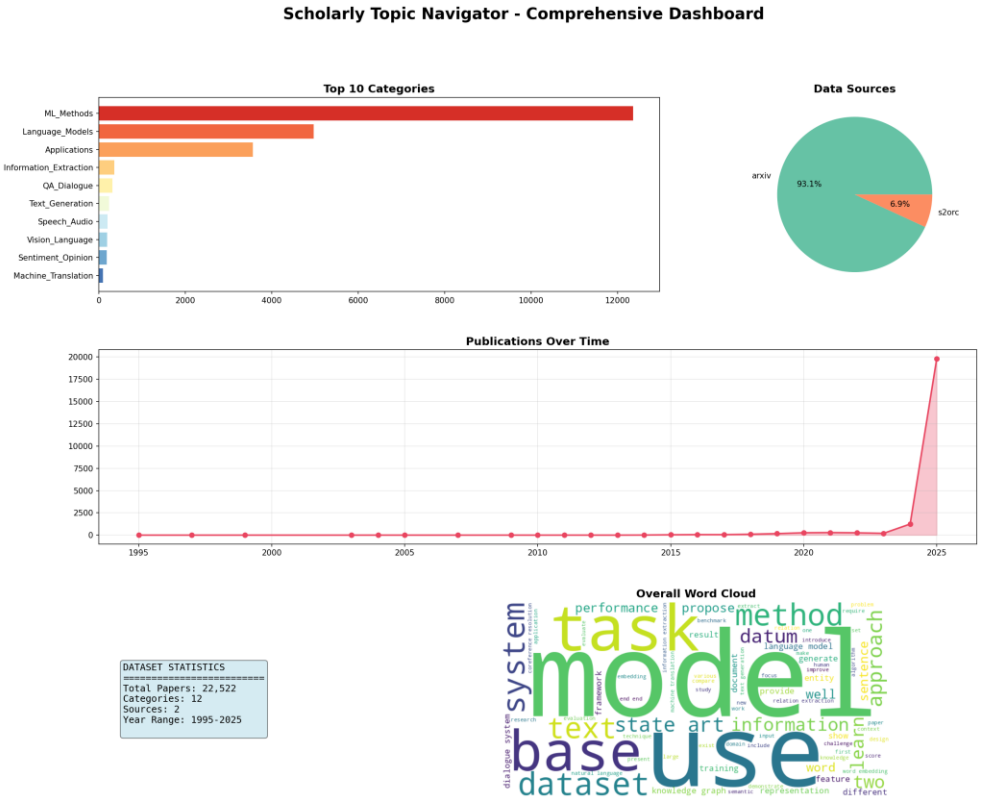

Figure 20: Comprehensive System Dashboard

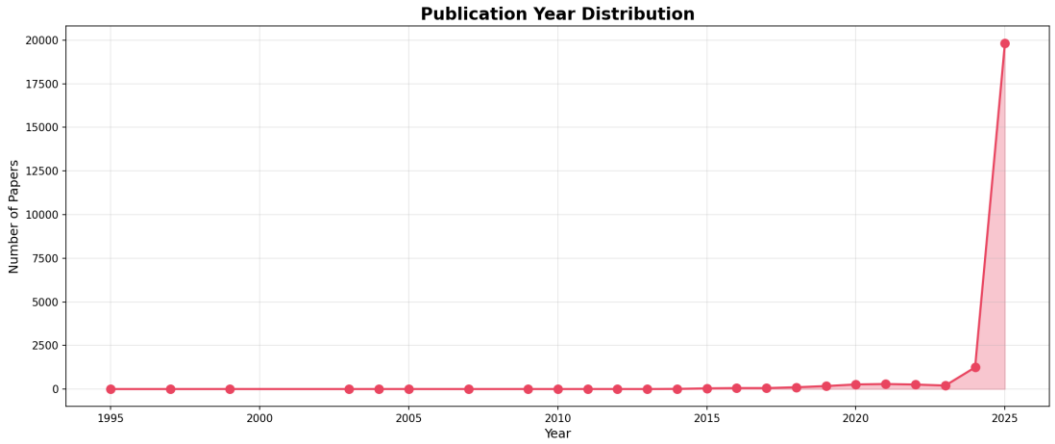

Figure 21: Paper Distribution by Year

7. Summary and Conclusions

7.1 Key Accomplishments

This paper presented the Scholarly Topic Navigator, an end-to-end NLP pipeline that demonstrates how multiple complementary techniques can be integrated into a practical tool for research literature discovery.

Table 18: Summary of Results

Component	Key Metric	Value	Notes
Data Collection	Final papers	22,522	From ArXiv + S2
Retrieval	MRR	0.83	BM25-weighted hybrid
Classification	Accuracy	67%	12-class taxonomy
Classification	Speedup	~50x	Student vs teacher
Topic Modeling	LDA Coherence	0.448	At 15 topics
Topic Modeling	BERTopic Topics	75	Fine-grained
Embedding	SBERT Balance	0.56 avg sim	Good discrimination

The system successfully demonstrates several key findings: hybrid retrieval benefits from careful weight tuning (counter to our initial hypothesis, BM25 remains important for scientific text); knowledge distillation is highly effective for deploying expensive models (50x speedup with acceptable accuracy); class imbalance fundamentally limits classifier performance on minority categories; and LIME provides valuable transparency for building user trust.

7.2 Challenges and Solutions

The development process encountered several significant challenges requiring engineering solutions:

- FAISS Segmentation Fault on macOS: A deep incompatibility between FAISS and Apple Silicon's OpenMP implementation caused segmentation faults during vector search. We resolved this by implementing a NumPy-based cosine similarity fallback. At 22,000 documents, the performance difference is negligible (<5ms), making this a pragmatic choice prioritizing stability over theoretical efficiency.
- SciBERT Semantic Collapse: High embedding similarity (0.87 average) caused all papers to appear similar for clustering, preventing meaningful topic discovery. This informed our selection of SBERT (0.56 similarity) for topic modeling while acknowledging SciBERT's value for similarity search where high similarity is desirable.
- Zero-Shot Threshold Issue: Initial confidence thresholds caused 95%+ papers to be classified as "Unclassified" because probability mass distributed across 12 categories produces low per-class confidence even for correct predictions. Resolution: always use argmax selection without thresholding, trusting relative rankings.
- Class Imbalance: The heavily skewed distribution (ML_Methods: 54.9% vs Ethics_Bias: 0.07%) leads to zero F1 for minority classes as the model learns to over-predict majority classes. This is documented as a fundamental limitation requiring techniques like oversampling or focal loss to address.
- BERTopic Outliers: HDBSCAN initially assigned 38% of papers to noise (Topic -1). We implemented centroid-based reassignment: for each discovered topic, compute the mean embedding of assigned documents, then assign noise documents to their nearest centroid.

7.3 Future Directions

Future work should address several opportunities for improvement:

- Data Expansion: Incorporate additional sources with abstract text, particularly ACL Anthology via web scraping, to expand venue coverage and include high-quality peer-reviewed papers.
- Class Imbalance: Apply oversampling (SMOTE), class-weighted training, or focal loss to improve minority class performance. Alternatively, hierarchical classification could group rare categories.
- Domain Adaptation: Fine-tune transformer encoders (SBERT, SciBERT) on our corpus for domain-specific classification, potentially improving performance on scientific terminology.

- Evaluation Methodology: Collect human relevance judgments for principled retrieval evaluation rather than relying on synthetic ground truth based on categories.
- Production Deployment: Migrate from Kaggle notebooks to production infrastructure with authentication, scaling, persistent storage, and scheduled data updates.
- Real-time Updates: Implement streaming updates to ingest new papers as they're published on ArXiv, keeping the system current with recent research.

The complete system demonstrates that careful combination of multiple NLP techniques can create practical tools for research literature discovery. Every parameter has a documented reason—the choices reflect tradeoffs between speed, accuracy, memory usage, and interpretability appropriate for a 22,000-paper corpus analyzed on consumer-grade hardware.

8. References

1. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT 2019.
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems.
3. Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP-IJCNLP 2019.
4. Cohan, A., Feldman, S., Beltagy, I., Downey, D., and Weld, D. S. (2020). SPECTER: Document-level Representation Learning using Citation-informed Transformers. Proceedings of ACL 2020.
5. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Proceedings of ACL 2020.
6. Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv preprint arXiv:2203.05794.
7. Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993-1022.
8. Robertson, S. and Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval, 3(4), 333-389.
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems.
10. Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why Should I Trust You?: Explaining the Predictions of Any Classifier. Proceedings of KDD 2016.
11. Johnson, J., Douze, M., and Jegou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data.
12. Hugging Face. (2024). Transformers: State-of-the-Art Natural Language Processing. https://huggingface.co/transformers
13. Sentence-Transformers. (2024). Sentence Embeddings using Siamese BERT-Networks. https://www.sbert.net/
14. Facebook Research. (2024). FAISS: A library for efficient similarity search. https://github.com/facebookresearch/faiss
15. spaCy. (2024). Industrial-Strength Natural Language Processing in Python. https://spacy.io/
16. Gensim. (2024). Topic modelling for humans. https://radimrehurek.com/gensim/
17. Streamlit. (2024). The fastest way to build data apps. https://streamlit.io/