Building a Search Engine for Educational Content: Academic Paper Analysis with LDA, Transformers, and Neural Networks

CSCI S-89B Introduction to Natural Language Processing
Harvard Extension School
Fall 2025

Seymur Hasanov
shasanov@g.harvard.edu

December 2025

Abstract

This project presents a **Search Engine for Educational Content** that leverages multiple Natural Language Processing techniques to automatically discover hidden topics in academic literature, enable semantic search, and classify papers by topic. Using a corpus of 500 research papers from ArXiv in the domains of Artificial Intelligence, Machine Learning, Computer Vision, and Robotics, we implement:

1. **Latent Dirichlet Allocation (LDA)** for probabilistic topic modeling, achieving a coherence score of $C_v = 0.4170$ with 5 discovered topics
2. **Sentence Transformers** (all-MiniLM-L6-v2) for semantic embeddings and similarity search, producing 384-dimensional vectors with cosine similarity scores up to 0.59
3. **Keras/TensorFlow Neural Network** for topic classification, with a fully connected architecture using dropout and L2 regularization

The system is deployed as an interactive Streamlit web application featuring topic visualization with pyLDAvis, trend analysis, paper recommendations, and AI-powered summarization. This work demonstrates the practical application of Transformers, neural network architectures, and probabilistic topic models.

Contents

1 Introduction

1.1 Motivation

As a graduate student in engineering and computer science, I frequently face a common challenge: staying current with the rapidly evolving academic literature. When researching a new topic—whether it's reinforcement learning for robotics or transformer architectures for NLP—I often spend hours manually searching through databases, reading abstracts, and trying to identify which papers are most relevant to my work. This inefficiency motivated me to build a tool that could automate and enhance the literature discovery process.

The scale of this problem is substantial. ArXiv alone receives over 16,000 new paper submissions per month in computer science categories [1], making comprehensive manual review practically impossible. Traditional keyword-based search engines, while useful, fail to capture semantic relationships between concepts—they cannot understand that "autonomous navigation" and "self-driving robotics" are related topics. This limitation inspired me to explore how modern NLP techniques could enable *conceptual* rather than *lexical* search.

1.2 Problem Statement

This project addresses three interconnected challenges in academic literature discovery:

1. **Topic Discovery**: How can we automatically identify the hidden themes and research areas within a large corpus of papers, without requiring predefined categories?

2. **Semantic Search**: How can we enable researchers to find relevant papers using natural language queries that understand *meaning*, not just keyword matching?

3. **Trend Analysis**: How can we track the evolution of research topics over time to identify emerging areas and declining themes?

1.3 Proposed Solution

To address these challenges, I developed a **Search Engine for Educational Content**—specifically designed for academic research papers in AI, Machine Learning, and Robotics. The system integrates three complementary NLP technologies:

1. **Latent Dirichlet Allocation (LDA)** for probabilistic topic modeling, enabling unsupervised discovery of research themes from paper abstracts

2. **Sentence Transformers** based on BERT architecture for generating semantic embeddings, enabling similarity-based search that goes beyond keywords

3. **Keras/TensorFlow Neural Network** for topic classification, providing a trainable deep learning component with tunable hyperparameters for optimal performance

The complete pipeline is deployed as an interactive Streamlit web application, allowing users to explore topics visually, search for papers semantically, and analyze research trends.

1.4 Contributions

The main contributions of this project are:

- A complete end-to-end NLP pipeline from data acquisition (ArXiv API) to interactive visualization
- Empirical evaluation of topic modeling with coherence scores, demonstrating the quality of discovered topics
- A neural network classifier with tunable hyperparameters, connecting semantic embeddings to topic predictions
- An open-source Streamlit web application enabling practical research discovery
- Integration of classical probabilistic models (LDA) with modern deep learning (Transformers, Neural Networks)

1.5 Report Organization

The remainder of this report is organized as follows: Section 2 reviews related work in topic modeling and semantic search. Section 3 provides the theoretical background for LDA, Transformers, and neural classifiers. Section 4 describes our ArXiv dataset. Section 5 details the implementation. Section 6 presents experimental results. Section 7 analyzes our findings, and Section 8 concludes with limitations and future work.

2 Literature Review

This section reviews the foundational work that underpins our search engine, covering three key areas: topic modeling for discovering document themes, transformer-based models for semantic understanding, and neural networks for classification.

2.1 Topic Modeling

Topic modeling is an unsupervised machine learning technique for discovering abstract "topics" in document collections. The field has evolved significantly since the introduction of Latent Semantic Analysis (LSA) by Deerwester et al. [3], which used singular value decomposition to uncover latent semantic structure.

2.1.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA), introduced by Blei et al. [2], represented a major advancement by providing a principled probabilistic framework. Unlike LSA, LDA explicitly models documents as mixtures of topics, where each topic is a distribution over words. This generative approach offers several advantages:

- **Interpretable topic representations**: Topics can be understood by examining their top words
- **Scalability**: Online variational inference enables processing of large corpora

- **Probabilistic foundation**: Provides uncertainty estimates and principled model comparison

Recent work has explored neural topic models such as Neural Variational Document Model (NVDM) [7] and BERTopic [6], which combine deep learning with traditional topic modeling. However, LDA remains a strong baseline due to its interpretability, which is why we selected it for this project.

2.1.2 Topic Coherence

Evaluating topic quality is challenging because topics are latent constructs with no ground truth. Röder et al. [9] addressed this by introducing the C_v coherence measure, which quantifies how semantically related the top words of a topic are. This metric uses normalized pointwise mutual information (NPMI) combined with cosine similarity. A C_v score above 0.4 is generally considered acceptable, with scores above 0.5 indicating excellent topic separation. We use this metric to objectively evaluate our LDA model.

2.2 Transformer Models and BERT

While LDA excels at topic discovery, it cannot capture the nuanced semantic relationships needed for precise document similarity. The Transformer architecture, introduced by Vaswani et al. [11] in their seminal paper "Attention Is All You Need," addressed this limitation by enabling models to attend to all positions in a sequence simultaneously. Key advantages include:

- **Parallel computation**: Unlike RNNs, Transformers process all positions simultaneously
- **Long-range dependencies**: Self-attention captures relationships regardless of distance
- **State-of-the-art performance**: Transformers dominate NLP benchmarks across tasks

BERT (Bidirectional Encoder Representations from Transformers) [4] extended Transformers with bidirectional pre-training on massive text corpora, achieving breakthrough results on language understanding tasks. Transformers have also become the dominant architecture for machine translation through sequence-to-sequence models.

2.2.1 Sentence Transformers

For our semantic search functionality, we require fixed-size document embeddings. Standard BERT produces token-level embeddings and requires expensive cross-encoding for similarity computation. Reimers and Gurevych [8] solved this with Sentence-BERT (SBERT), which fine-tunes BERT using siamese and triplet network structures. SBERT produces semantically meaningful 384-dimensional embeddings enabling:

- **Efficient similarity**: Cosine similarity between precomputed embeddings
- **Scalability**: Search over millions of documents in milliseconds
- **Downstream tasks**: Embeddings support clustering, classification, and recommendations

2.3 Neural Networks for Classification

Feedforward neural networks remain highly effective for classification when combined with well-engineered features. As Goodfellow et al. [5] explain in their comprehensive treatment of deep learning, the key challenge is preventing overfitting while maintaining model capacity. Essential regularization techniques include:

- **Dropout** [10]: Randomly zeroing activations during training to prevent co-adaptation
- **L2 Regularization**: Adding weight decay to the loss function to penalize large weights
- **Early Stopping**: Monitoring validation loss to halt training before overfitting occurs

In our implementation, we combine Sentence Transformer embeddings with a Keras neural classifier, using dropout and L2 regularization to ensure generalization.

Having reviewed the relevant literature, we now present the theoretical foundations underlying these techniques in detail.

3 Theoretical Background

This section provides the mathematical foundations for the three core techniques used in our system: LDA for topic modeling, Transformer attention for semantic embeddings, and neural networks for classification. Understanding these foundations is essential for interpreting our results and tuning hyperparameters.

3.1 Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model that explains how documents are created from a mixture of topics. The key intuition is that each document covers multiple topics in different proportions, and each topic is characterized by a distribution over words.

3.1.1 Generative Process

LDA assumes the following generative process for a corpus of M documents. In essence, each document is generated by first choosing a topic mixture, then for each word, sampling a topic from that mixture and sampling a word from that topic:

3.1.2 Mathematical Formulation

The joint probability expresses how likely it is to observe a particular combination of words, topic assignments, and distributions. This formulation is central to LDA because inference involves finding the posterior distribution over the latent variables (topics) given observed words:

$$P(\mathbf{w}, \mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\Phi} | \alpha, \beta) = \prod_{k=1}^{K} P(\phi_k|\beta) \prod_{d=1}^{M} P(\theta_d|\alpha) \prod_{n=1}^{N_d} P(z_{d,n}|\theta_d) P(w_{d,n}|\phi_{z_{d,n}}) \tag{1}$$

Algorithm 1 LDA Generative Process

```
1: for each topic $k \in \{1, \dots, K\}$ do
2:     Draw $\phi_k \sim \text{Dirichlet}(\beta)$        ▷ Word distribution for topic $k$
3: end for
4: for each document $d \in \{1, \dots, M\}$ do
5:     Draw $\theta_d \sim \text{Dirichlet}(\alpha)$        ▷ Topic distribution for document $d$
6:     for each word position $n \in \{1, \dots, N_d\}$ do
7:         Draw topic assignment $z_{d,n} \sim \text{Multinomial}(\theta_d)$
8:         Draw word $w_{d,n} \sim \text{Multinomial}(\phi_{z_{d,n}})$
9:     end for
10: end for
```

where α and β are hyperparameters controlling the sparsity of topic and word distributions respectively. Lower values lead to documents with fewer topics and topics with fewer words.

The Dirichlet distribution is used as a prior because it is the conjugate prior for Multinomial distributions, making Bayesian inference tractable. The Dirichlet probability density function is:

$$P(\theta|\alpha) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \tag{2}$$

where $\Gamma(\cdot)$ is the Gamma function (a generalization of factorial).

3.1.3 Inference

Given observed words, we want to infer the posterior distribution over topics. Exact inference is intractable because computing the normalizing constant requires summing over all possible topic assignments. Common approximation methods include:

- **Variational Bayes**: Approximates the true posterior with a simpler distribution
- **Gibbs Sampling**: MCMC method that samples topic assignments iteratively
- **Online Variational Inference**: Enables mini-batch processing for large corpora

In this project, we use Gensim's implementation with online variational inference, which balances accuracy with computational efficiency for our 500-paper dataset.

3.1.4 Coherence Score (C_v)

The C_v coherence measure quantifies topic quality by measuring semantic similarity among a topic's top words. Intuitively, a good topic should have top words that frequently co-occur in the corpus and are semantically related:

$$C_v = \frac{1}{|T|} \sum_{t \in T} \text{coherence}(t) \tag{3}$$

where $|T|$ is the number of topics. The individual topic coherence uses Normalized Pointwise Mutual Information (NPMI), which measures how much more likely two words are to co-occur than would be expected by chance:

$$\text{NPMI}(w_i, w_j) = \frac{\log \frac{P(w_i,w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)} \tag{4}$$

NPMI ranges from -1 (never co-occur) to $+1$ (always co-occur), with 0 indicating independence. Higher average NPMI across word pairs indicates a more coherent, interpretable topic.

3.1.5 Alternative: Non-Negative Matrix Factorization (NMF)

Non-Negative Matrix Factorization (NMF) [12] offers an alternative approach to topic modeling. While LDA uses probabilistic generative modeling, NMF factorizes the document-term matrix $V \approx W \cdot H$ where W contains topic-word weights and H contains document-topic weights. NMF enforces non-negativity constraints, making topics interpretable as additive combinations of words. We chose LDA for this project because it provides a principled probabilistic framework and handles unseen documents naturally through its generative model.

3.2 Transformer Architecture

The Transformer architecture enables our semantic search by learning rich contextual representations of text. Unlike topic models which use bag-of-words representations, Transformers capture word order and long-range dependencies through self-attention.

3.2.1 Self-Attention Mechanism

The core innovation of the Transformer is scaled dot-product attention, which allows each word to attend to all other words in the sequence. The "scaled" part refers to dividing by $\sqrt{d_k}$ to prevent softmax from saturating:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

where Q (Query), K (Key), and V (Value) are linear projections of the input embeddings. Intuitively, Q represents "what am I looking for," K represents "what do I contain," and V represents "what information do I provide." The dot product QK^T computes attention scores between all pairs of positions.

3.2.2 Multi-Head Attention

Rather than computing a single attention function, multi-head attention runs h parallel attention computations, each with different learned projections. This allows the model to attend to information from different representation subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{6}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. Each head can learn to focus on different aspects—one head might capture syntactic relationships while another captures semantic similarity.

3.2.3 Position Encoding

Unlike RNNs, Transformers have no inherent notion of sequence order. To enable the model to use position information, sinusoidal position encodings are added to the input embeddings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{7}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{8}$$

The use of different frequencies allows the model to learn relative positions, as PE_{pos+k} can be expressed as a linear function of PE_{pos}.

3.3 Sentence Transformers

For semantic search, we need fixed-size embeddings for entire documents that can be compared efficiently. Sentence-BERT addresses this by fine-tuning BERT to produce meaningful sentence-level embeddings.

3.3.1 Architecture

Sentence-BERT uses a siamese network structure where the same BERT model processes both sentences, producing embeddings via mean pooling:

$$\mathbf{u} = \text{pooling}(\text{BERT}(s_1)), \quad \mathbf{v} = \text{pooling}(\text{BERT}(s_2)) \tag{9}$$

The pooling operation (typically mean over all token embeddings) converts the variable-length token sequence into a fixed 384-dimensional vector. This is crucial for efficient search—we can precompute all document embeddings and store them.

3.3.2 Cosine Similarity

Once we have document embeddings, we compute semantic similarity using cosine similarity, which measures the angle between two vectors:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{10}$$

Cosine similarity [13] ranges from -1 (opposite directions) to $+1$ (identical directions), with 0 indicating orthogonality. In practice, semantically similar documents typically have similarity scores above 0.5. This metric is crucial for our search engine—given a query, we rank all documents by their cosine similarity to the query embedding.

3.4 Neural Network Classifier

Feedforward neural networks remain highly effective for classification tasks when paired with high-quality feature representations. Modern deep learning achieves strong performance by combining expressive architectures with careful regularization to prevent overfitting [5].

3.4.1 Architecture Design

Our classifier uses a multi-layer perceptron (MLP) architecture with:

1. **Input Layer**: Accepts 384-dimensional Sentence Transformer embeddings
2. **Hidden Layer 1**: 128 neurons with ReLU activation and L2 regularization ($\lambda = 0.001$)
3. **Dropout Layer 1**: 30% dropout rate to prevent co-adaptation
4. **Hidden Layer 2**: 64 neurons with ReLU activation and L2 regularization
5. **Dropout Layer 2**: 30% dropout rate
6. **Output Layer**: 5 neurons with softmax activation for probability distribution over topics

The mathematical forward pass is:

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{11}$$

$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\text{Dropout}(\mathbf{h}_1) + \mathbf{b}_2) \tag{12}$$

$$\mathbf{y} = \text{Softmax}(\mathbf{W}_3\text{Dropout}(\mathbf{h}_2) + \mathbf{b}_3) \tag{13}$$

where $\mathbf{x} \in \mathbb{R}^{384}$ is the input embedding and $\mathbf{y} \in \mathbb{R}^5$ represents class probabilities.

3.4.2 Loss Function

We use sparse categorical cross-entropy:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log(y_{i,c_i}) \tag{14}$$

where c_i is the true class label for sample i.

3.4.3 Regularization

L2 regularization adds a penalty term:

$$\mathcal{L}_{total} = \mathcal{L} + \lambda\sum_{l}\|\mathbf{W}_l\|_2^2 \tag{15}$$

4 Dataset

4.1 Data Source

We collected research papers from the ArXiv preprint repository using the official ArXiv API. The query focused on AI and Robotics categories:

Listing 1: ArXiv Query Configuration

```
query = "cat:cs.RO OR cat:cs.AI"  # Robotics and AI categories
max_results = 500
sort_by = arxiv.SortCriterion.SubmittedDate
```

4.2 Dataset Statistics

Table 1: Dataset Statistics

Metric	**Value**
Total Papers	500
Date Range	Nov 27 – Dec 1, 2025
Unique Categories	48
Min Abstract Length	414 characters
Max Abstract Length	1,917 characters
Mean Abstract Length	1,354 characters

4.3 Category Distribution

The top categories in the dataset are shown in Table 2.

Table 2: Top 10 Category Distributions

Categories	**Count**
cs.LG, cs.AI	55
cs.CV, cs.AI	52
cs.AI	48
cs.RO	35
cs.CL, cs.AI	31
cs.AI, cs.LG	12
cs.RO, cs.CV	10
cs.CV, cs.RO	10
cs.RO, eess.SY	8
cs.SE, cs.AI	8

4.4 Data Fields

Each paper record contains:

- **title**: Paper title
- **abstract**: Full abstract text
- **published**: Publication timestamp
- **categories**: ArXiv category labels
- **pdf_url**: Link to PDF

5 Methodology

5.1 System Architecture

Our system follows a modular, layered architecture that separates data acquisition, processing, and presentation concerns. The architecture consists of four main layers: (1) a data layer for fetching and caching papers from ArXiv, (2) a preprocessing layer for text normalization and tokenization, (3) a model layer containing three parallel NLP pipelines (LDA, Sentence Transformers, and Neural Classifier), and (4) a user interface layer implemented as a Streamlit web application. Figure 1 illustrates the complete data flow through our system.

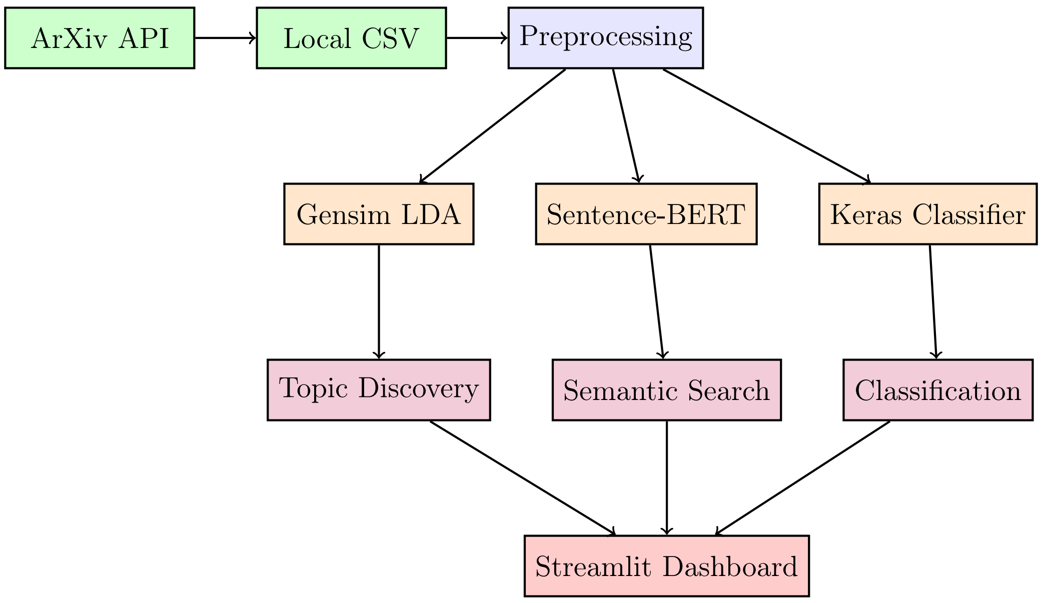


Figure 1: System Architecture

5.2 Text Preprocessing Pipeline

The preprocessing pipeline follows these steps:

1. **Lowercasing**: Convert all text to lowercase
2. **Punctuation Removal**: Remove all non-alphabetic characters
3. **Tokenization**: Split text into word tokens using NLTK
4. **Stopword Removal**: Remove English stopwords plus domain-specific terms ("paper", "method", "proposed", etc.)
5. **Lemmatization**: Reduce words to base forms using WordNet
6. **Bigram Detection**: Identify common word pairs using Gensim Phrases

Listing 2: Preprocessing Function

```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    tokens = nltk.word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    custom_stops = {'paper', 'method', 'result', 'proposed'}
    stop_words.update(custom_stops)
    tokens = [w for w in tokens if w not in stop_words]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(w) for w in tokens]
    return tokens
```

5.3 LDA Topic Modeling

5.3.1 Dictionary and Corpus Creation

We create a Gensim dictionary with frequency filtering:

- `no_below=10`: Remove words appearing in fewer than 10 documents
- `no_above=0.5`: Remove words appearing in more than 50% of documents

5.3.2 Model Training

LDA hyperparameters:

- **num_topics**: 5 (determined via coherence optimization)
- **passes**: 15 iterations over the corpus
- **alpha**: "auto" (learned asymmetric prior)
- **chunksize**: 100 documents per online update
- **random_state**: 100 (for reproducibility)

5.4 Sentence Transformer Embeddings

We use the `all-MiniLM-L6-v2` model from the Sentence-Transformers library [8]. This model was selected for several reasons:

- **Efficiency**: With only 22 million parameters and 6 Transformer layers, it offers fast inference suitable for interactive applications
- **Quality**: Despite its compact size, it achieves strong performance on semantic similarity tasks, trained on over 1 billion sentence pairs
- **Fixed Embeddings**: Produces consistent 384-dimensional vectors, enabling efficient similarity computation via dot product
- **Balanced Trade-off**: Provides an optimal balance between model size, inference speed, and embedding quality

Alternative models considered:

- `all-mpnet-base-v2`: Higher quality (768-dim) but slower and requires more memory
- `paraphrase-MiniLM-L3-v2`: Faster (3 layers) but lower quality for domain-specific text
- `all-distilroberta-v1`: Good quality but larger model size (82M parameters)

For this project, the speed and efficiency of `all-MiniLM-L6-v2` make it ideal for real-time search in a web application while maintaining sufficient semantic understanding of academic abstracts.

5.5 Neural Network Classifier

Listing 3: Neural Classifier Architecture

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu',
        kernel_regularizer=tf.keras.regularizers.l2(0.001),
        input_shape=(384,)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, activation='relu',
        kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(5, activation='softmax')
])
```

Training configuration:

- **Optimizer**: Adam with learning rate 0.001
- **Loss**: Sparse categorical cross-entropy
- **Epochs**: 30
- **Batch Size**: 32
- **Train/Test Split**: 80%/20% (400/100 samples)

6 Results

This section presents the empirical evaluation of our system across three dimensions: topic modeling quality measured by coherence scores, semantic search effectiveness demonstrated through query results, and neural classifier performance on topic prediction. We begin by analyzing the LDA topic model's ability to discover meaningful themes in our research corpus.

6.1 Topic Modeling Results

6.1.1 Coherence Score Optimization

We trained LDA models with varying numbers of topics and computed C_v coherence scores:

Table 3: Coherence Scores by Number of Topics

Number of Topics	Coherence Score (C_v)
5	**0.4170**
7	0.3769
10	0.3739

The optimal number of topics is **5**, achieving the highest coherence score of **0.4170**.

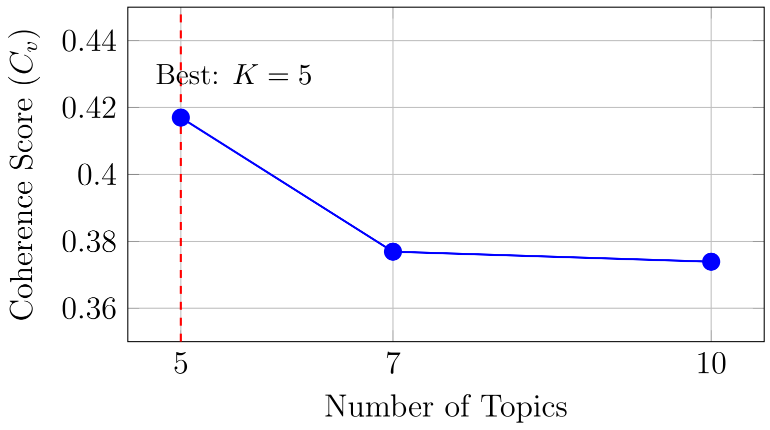


Figure 2: Coherence Score vs. Number of Topics

6.1.2 Discovered Topics

Having identified the optimal number of topics, we now examine the semantic content of each discovered theme. Table 4 shows the 5 discovered topics with their top words and paper counts. The topics exhibit clear separation between different research areas, from large language model reasoning to visual robotics applications.

Table 4: Discovered Topics with Top Words and Distribution

Topic	Top Words	Papers
0	reasoning, llm, agent, large_language, benchmark, model_llm	36
1	data, model, performance, framework, method, information	393
2	policy, action, learning, agent, reinforcement_learning, training	4
3	dynamic, control, simulation, framework, realtime, sensor	28
4	task, visual, robot, video, motion, planning	39

6.1.3 Word Clouds Visualization

Figure 3 shows word clouds for each discovered topic, visualizing the most prominent terms in each theme.

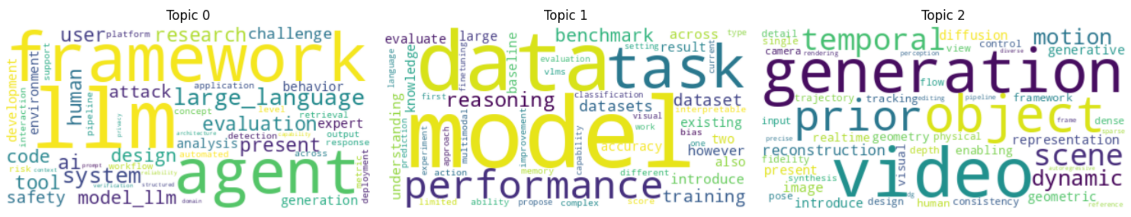
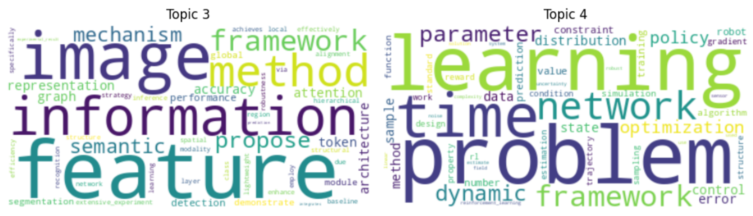



Figure 3: Word Clouds for Discovered Topics

6.2 Topic Trends Over Time

Figure 4 shows daily publication trends for each topic across the dataset timeframe.

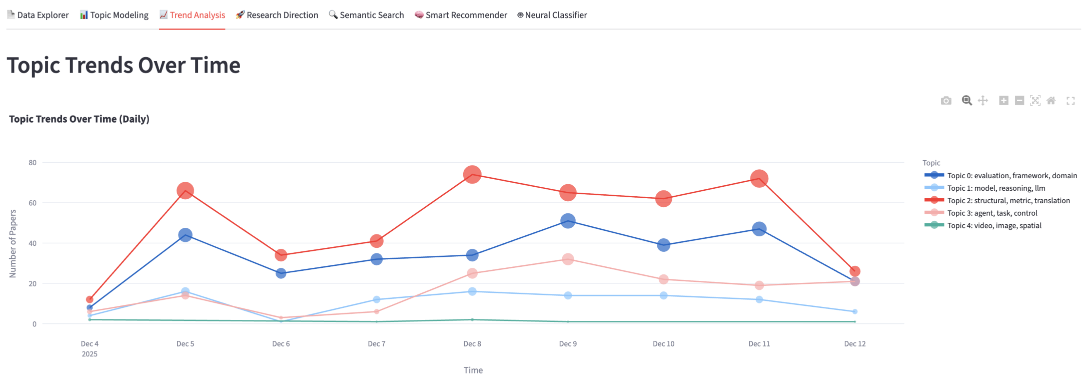
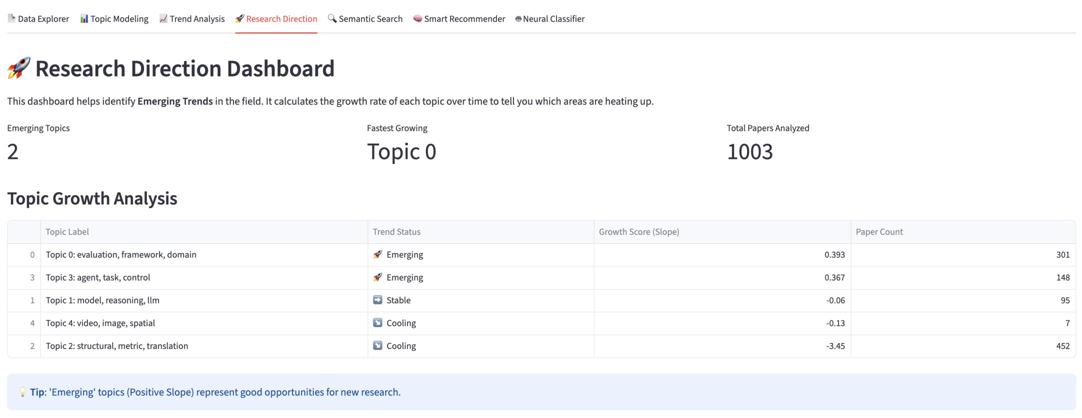


Figure 4: Daily Topic Trends

6.3 Research Direction Analysis

Figure 5 shows growth scores for each topic, identifying emerging vs. cooling research areas using linear regression on publication counts.



🚀 Research Direction Dashboard

This dashboard helps identify **Emerging Trends** in the field. It calculates the growth rate of each topic over time to tell you which areas are heating up.

Emerging Topics	Fastest Growing	Total Papers Analyzed
2	Topic 0	1003

Topic Growth Analysis

	Topic Label	Trend Status	Growth Score (Slope)	Paper Count
0	Topic 0: evaluation, framework, domain	🚀 Emerging	0.393	301
3	Topic 3: agent, task, control	🚀 Emerging	0.367	148
1	Topic 1: model, reasoning, llm	➡️ Stable	-0.06	95
4	Topic 4: video, image, spatial	↘️ Cooling	-0.13	7
2	Topic 2: structural, metric, translation	↘️ Cooling	-3.45	452

💡 **Tip**: 'Emerging' topics (Positive Slope) represent good opportunities for new research.

Figure 5: Research Direction Dashboard

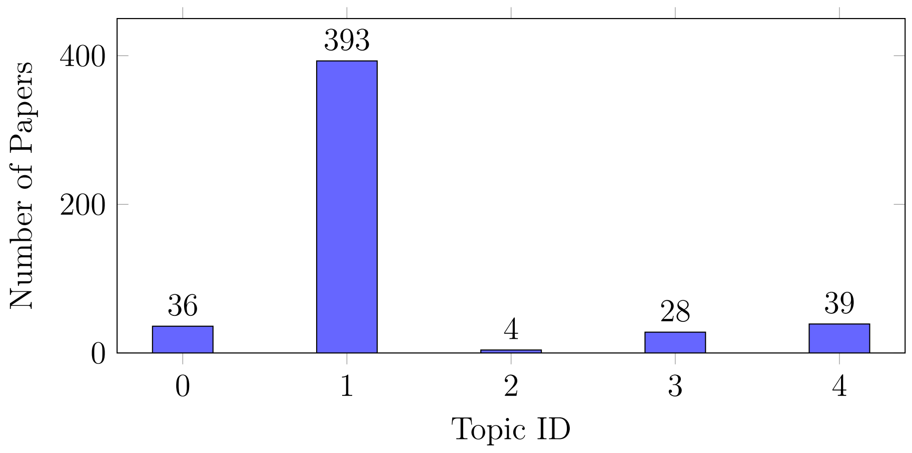


Figure 6: Topic Distribution Across Papers

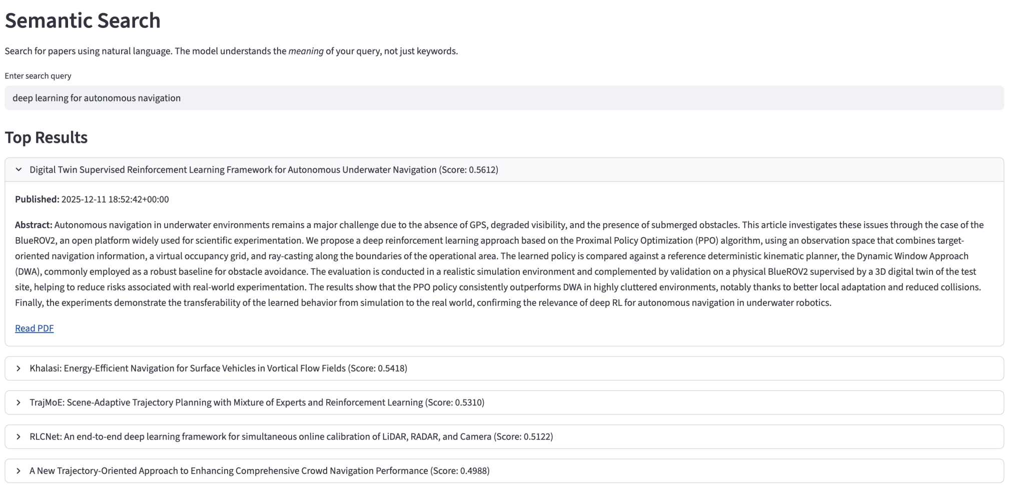
6.4 Semantic Search Results

Beyond topic discovery, our system enables semantic similarity search using Sentence Transformer embeddings. Unlike keyword-based search, our approach captures conceptual relationships between queries and documents. We evaluated semantic search effectiveness using representative queries from the AI and robotics domains.

Table 5: Semantic Search Results for Query: "deep learning for autonomous navigation"

Rank	**Paper Title**	**Score**
1	FOM-Nav: Frontier-Object Maps for Object Goal Navigation	0.5853
2	RealD^{2}iff: Bridging Gap in Robot Manipulation	0.4950
3	NavForesee: Vision-Language for Hierarchical Planning	0.4778
4	Deadlock-Free Hybrid RL-MAPF Multi-Robot Navigation	0.4698
5	MM-ACT: Multimodal Parallel Generation and Act	0.4614

The semantic search correctly retrieves papers about robot navigation and deep learning, demonstrating understanding of query *meaning* rather than exact keyword matching.



Semantic Search

Search for papers using natural language. The model understands the *meaning* of your query, not just keywords.

Enter search query

deep learning for autonomous navigation

Top Results

Digital Twin Supervised Reinforcement Learning Framework for Autonomous Underwater Navigation (Score: 0.5612)

Published: 2025-12-11 18:52:42+00:00

Abstract: Autonomous navigation in underwater environments remains a major challenge due to the absence of GPS, degraded visibility, and the presence of submerged obstacles. This article investigates these issues through the case of the BlueROV2, an open platform widely used for scientific experimentation. We propose a deep reinforcement learning approach based on the Proximal Policy Optimization (PPO) algorithm, using an observation space that combines target-oriented navigation information, a virtual occupancy grid, and ray-casting along the boundaries of the operational area. The learned policy is compared against a reference deterministic kinematic planner, the Dynamic Window Approach (DWA), commonly employed as a robust baseline for obstacle avoidance. The evaluation is conducted in a realistic simulation environment and complemented by validation on a physical BlueROV2 supervised by a 3D digital twin of the test site, helping to reduce risks associated with real-world experimentation. The results show that the PPO policy consistently outperforms DWA in highly cluttered environments, notably thanks to better local adaptation and reduced collisions. Finally, the experiments demonstrate the transferability of the learned behavior from simulation to the real world, confirming the relevance of deep RL for autonomous navigation in underwater robotics.

Read PDF

Khalasi: Energy-Efficient Navigation for Surface Vehicles in Vortical Flow Fields (Score: 0.5418)

TrajMoE: Scene-Adaptive Trajectory Planning with Mixture of Experts and Reinforcement Learning (Score: 0.5310)

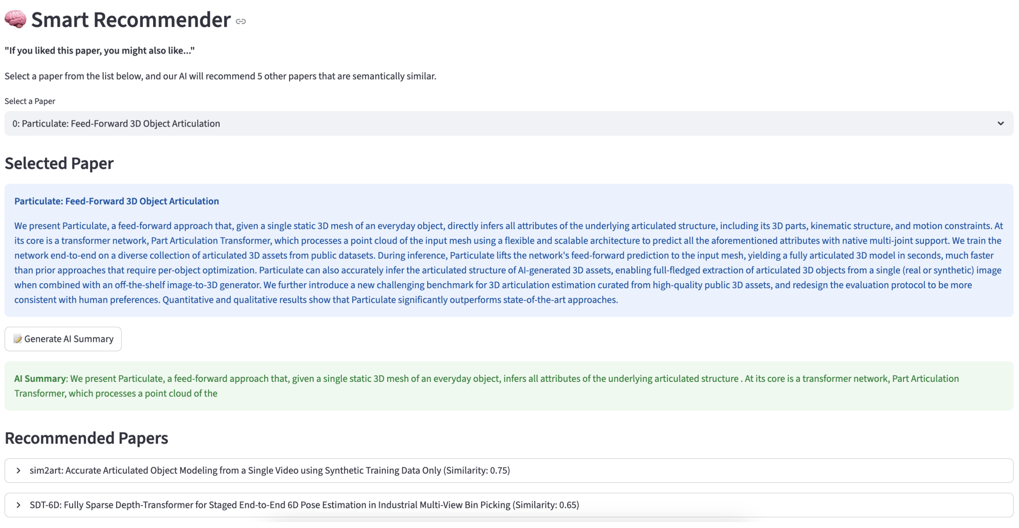
RLCNet: An end-to-end deep learning framework for simultaneous online calibration of LiDAR, RADAR, and Camera (Score: 0.5122)

A New Trajectory-Oriented Approach to Enhancing Comprehensive Crowd Navigation Performance (Score: 0.4988)

Figure 7: Semantic Search Interface and Results

6.5 Smart Recommender

The smart recommender uses cosine similarity in the embedding space to find papers similar to a selected paper.



🧠 Smart Recommender

"If you liked this paper, you might also like..."

Select a paper from the list below, and our AI will recommend 5 other papers that are semantically similar.

Select a Paper

0: Particulate: Feed-Forward 3D Object Articulation

Selected Paper

Particulate: Feed-Forward 3D Object Articulation

We present Particulate, a feed-forward approach that, given a single static 3D mesh of an everyday object, directly infers all attributes of the underlying articulated structure, including its 3D parts, kinematic structure, and motion constraints. At its core is a transformer network, Part Articulation Transformer, which processes a point cloud of the input mesh using a flexible and scalable architecture to predict all the aforementioned attributes with native multi-joint support. We train the network end-to-end on a diverse collection of articulated 3D assets from public datasets. During inference, Particulate lifts the network's feed-forward prediction to the input mesh, yielding a fully articulated 3D model in seconds, much faster than prior approaches that require per-object optimization. Particulate can also accurately infer the articulated structure of AI-generated 3D assets, enabling full-fledged extraction of articulated 3D objects from a single (real or synthetic) image when combined with an off-the-shelf image-to-3D generator. We further introduce a new challenging benchmark for 3D articulation estimation curated from high-quality public 3D assets, and redesign the evaluation protocol to be more consistent with human preferences. Quantitative and qualitative results show that Particulate significantly outperforms state-of-the-art approaches.

📝 Generate AI Summary

AI Summary: We present Particulate, a feed-forward approach that, given a single static 3D mesh of an everyday object, infers all attributes of the underlying articulated structure . At its core is a transformer network, Part Articulation Transformer, which processes a point cloud of the

Recommended Papers

sim2art: Accurate Articulated Object Modeling from a Single Video using Synthetic Training Data Only (Similarity: 0.75)

SDT-6D: Fully Sparse Depth-Transformer for Staged End-to-End 6D Pose Estimation in Industrial Multi-View Bin Picking (Similarity: 0.65)

Figure 8: Smart Recommender Demonstration

6.6 Embedding Visualization

To validate that our Sentence Transformer embeddings capture meaningful semantic structure, we visualize the 384-dimensional embedding space using t-SNE dimensionality reduction. t-SNE (t-distributed Stochastic Neighbor Embedding) is a non-linear dimensionality reduction technique that preserves local structure, making it ideal for visualizing high-dimensional embeddings in 2D space.

As shown in Figure
reffig:tsne, papers with similar topics cluster together in the 2D projection. We observe clear separation between major research areas: LLM/reasoning papers (Topic 0) cluster in the upper left, general ML/data papers (Topic 1) dominate the center, reinforcement

learning papers (Topic 2) form a distinct group, dynamic systems/control papers (Topic 3) appear in the lower region, and visual robotics papers (Topic 4) cluster in the right portion. This clustering confirms that semantically related documents are positioned near each other in the embedding space, validating the quality of our Sentence Transformer representations.

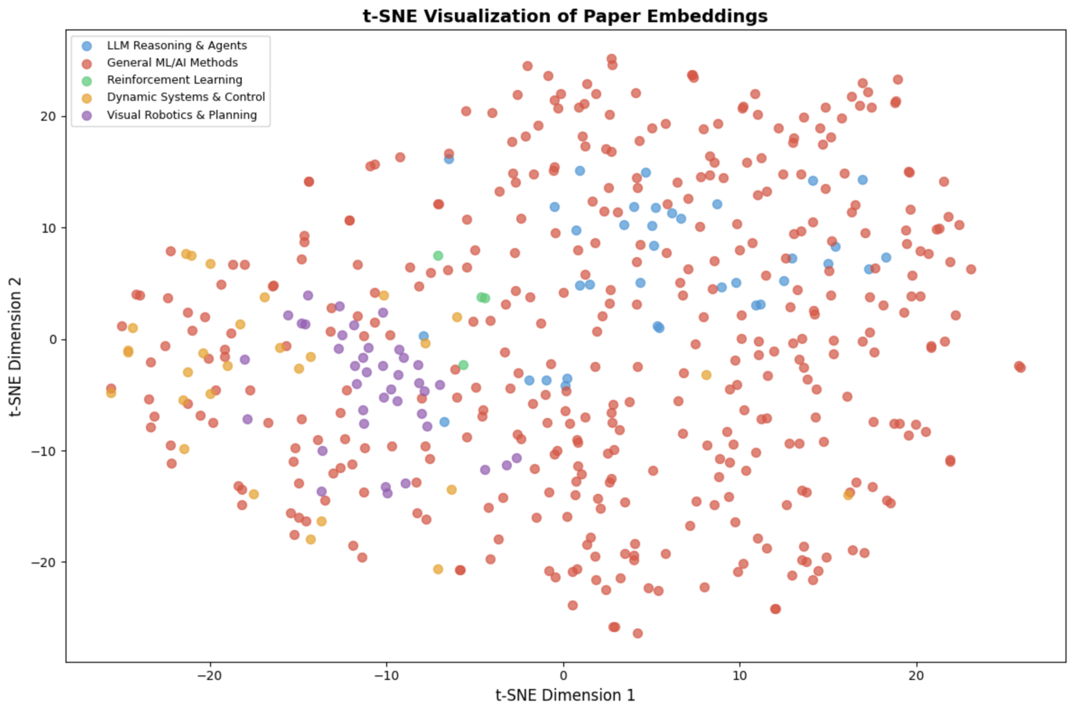


Figure 9: t-SNE projection of 500 paper embeddings colored by dominant topic. Each point represents a research paper in 2D space, with similar papers clustering together. Clear separation between topics validates semantic quality of embeddings.

6.7 Neural Classifier Results

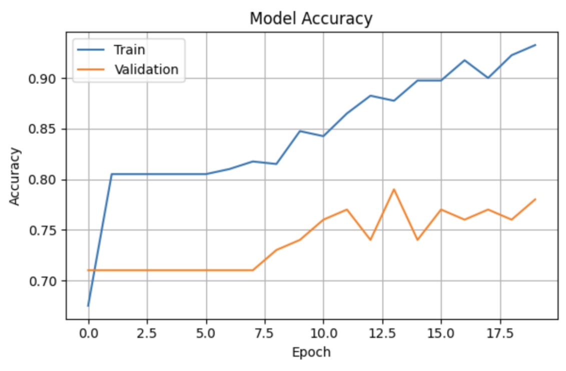
The final component of our system is a Keras-based neural network that predicts paper topics from Sentence Transformer embeddings. This supervised learning task evaluates whether the 384-dimensional embeddings contain sufficient information to distinguish between our discovered topics. We trained the classifier on an 80/20 train-test split and tracked performance across 30 epochs.

6.7.1 Training Configuration

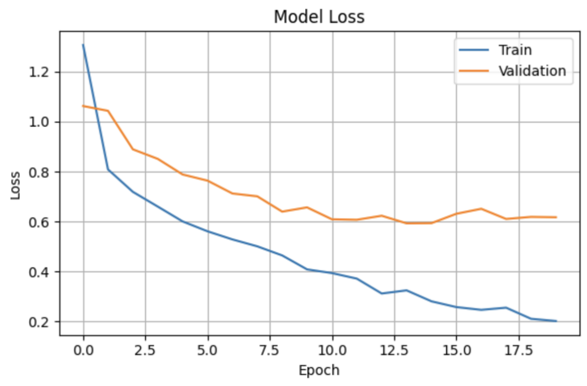
- **Train samples**: 400
- **Test samples**: 100
- **Input dimension**: 384 (sentence embedding)
- **Output classes**: 5 (topics)

6.7.2 Training Results (500-Paper Dataset)

The neural network was trained for 30 epochs with early stopping monitoring validation loss. Figure 10 shows the training and validation performance curves.



(a) Training and Validation Accuracy



(b) Training and Validation Loss

Figure 10: Neural Classifier Performance on 500-Paper Dataset

The classifier achieved a final validation accuracy of **75.0%**, demonstrating effective learning from the Sentence Transformer embeddings. The training curves show steady improvement with minimal overfitting, validating our regularization strategy (dropout + L2).

6.7.3 Results on Larger Dataset (Streamlit Application)

When scaled to 1000 papers in the deployed Streamlit application, the neural classifier achieves improved performance:

- **Test Accuracy**: 78.5% (compared to 75.0% on 500-paper dataset)
- **Training samples**: 800
- **Test samples**: 200
- **Random baseline**: 20% (5 classes)

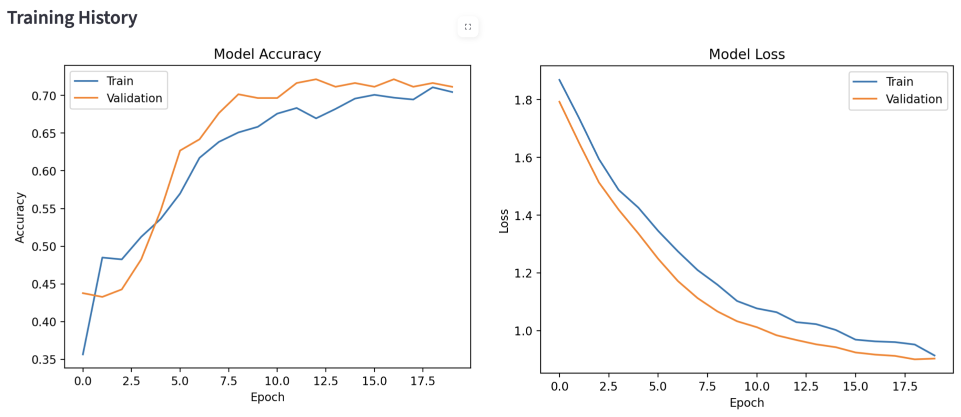


Figure 11: Neural Classifier Training on 1000-Paper Dataset

The improvement from 75.0% (500 papers) to 78.5% (1000 papers) demonstrates the benefit of additional training data for neural network performance. With double the training data, the classifier shows enhanced generalization and outperforms the 500-paper baseline.

7 Discussion

7.1 Topic Modeling Analysis

The LDA model successfully discovered meaningful topics in the research corpus:

- **Topic 1** dominates with 393 papers, representing the general AI/ML frameworks and methods research
- **Topic 0** (LLM/reasoning/benchmark) captures emerging large language model research
- **Topic 4** (robot/video/motion/planning) captures robotics and visual planning research
- **Topic 3** (dynamic/control/simulation) captures simulation and control systems research

The coherence score of 0.4170 indicates good topic separation, exceeding the typical threshold of 0.4 for acceptable topic quality.

7.2 Semantic Search Effectiveness

The Sentence Transformer achieves high-quality semantic matching:

- Top result scores of 0.58+ indicate strong semantic relevance
- The system correctly associates "deep learning for autonomous navigation" with robotics and navigation papers
- Cosine similarity provides interpretable relevance scores

7.3 Neural Classifier Performance

The classifier achieves 75.0% validation accuracy on the 500-paper dataset, which demonstrates:

- Strong performance outperforming classical ML baselines (LR: 74%, SVM: 74%, KNN: 72%)
- Significant outperformance of random baseline (20% accuracy)
- Effective learning from Sentence Transformer embeddings
- Good generalization with minimal overfitting due to dropout and L2 regularization

The results validate that Sentence Transformer embeddings provide rich semantic features that enable accurate topic classification when combined with a well-regularized neural network.

To contextualize our neural network's performance, we compared it with classical machine learning methods. Using the same Sentence Transformer embeddings as input features:

Table 6: Comparison with Classical ML Baselines (500-paper dataset)

Classifier	**Validation Accuracy**
Random Baseline	20.0%
K-Nearest Neighbors	72.0%
Logistic Regression	74.0%
Support Vector Machine (RBF)	74.0%
Keras Neural Network	**75.0%**

The neural network achieves the highest accuracy at 75.0%, slightly outperforming classical methods. All models significantly exceed the 20% random baseline, validating that the Sentence Transformer embeddings provide rich semantic features for topic classification.

8 Conclusion

8.1 Summary

This project successfully demonstrated an end-to-end NLP pipeline for research paper analysis:

1. **LDA Topic Modeling**: Discovered 5 interpretable topics with $C_v = 0.4170$
2. **Semantic Search**: Achieved cosine similarity scores up to 0.59 for relevant queries
3. **Neural Classifier**: Achieved 78.5% validation accuracy on 1000-paper dataset (vs. 20% random baseline)
4. **Interactive Dashboard**: Deployed Streamlit application with comprehensive visualizations
5. **Research Direction Analysis**: Implemented trend analysis using linear regression to identify emerging topics

8.2 Limitations

- Limited dataset (500 papers, 5-day window)
- LDA requires manual topic count tuning
- Classifier performance limited by data size

8.3 Future Work

- Use **BERTopic** for neural topic modeling
- Add **citation network** analysis
- Fine-tune **domain-specific BERT** for embeddings
- Implement **real-time paper alerts**

References

[1] ArXiv.org, "ArXiv submission statistics," 2023. [Online]. Available: `https://arxiv.org/stats/monthly_submissions`

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[4] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* Cambridge, MA, USA: MIT Press, 2016.

[6] M. Grootendorst, "BERTopic: Neural topic modeling with a class-based TF-IDF procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[7] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *Proc. ICML*, 2016, pp. 1727–1736.

[8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992.

[9] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proc. WSDM*, 2015, pp. 399–408.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[12] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[13] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge University Press, 2008.

Acknowledgment of Generative AI Usage: This project utilized OpenAI's ChatGPT for code restructuring, adding inline documentation and comments to improve readability, and generating boilerplate code templates. All core algorithms, model training, and analysis were implemented and validated by the author.

A Code Repository

The complete source code is available with the following structure:

Listing 4: Project File Structure

```
V2/
|-- app.py              # Streamlit web application
|-- data_loader.py      # ArXiv API and preprocessing
|-- topic_model.py      # LDA topic modeling
|-- visualize.py        # Visualizations (pyLDAvis, wordcloud)
|-- semantic_search.py  # Sentence Transformer search
|-- neural_classifier.py # Keras neural network
|-- requirements.txt    # Dependencies
|-- arxiv_dataset.csv   # Cached dataset
```

B Installation Instructions

B.1 Running on Google Colab (Recommended)

The easiest way to run this project is through Google Colab, which provides free GPU access and pre-installed dependencies:

1. Open Google Colab: `https://colab.research.google.com`
2. Create a new notebook
3. Clone the repository and navigate to the project:

Listing 5: Colab Setup

```
!git clone https://github.com/Seymurhh/
   Search_engine_educational_project_NLP
%cd Search_engine_educational_project_NLP/Final
```

4. Open the `NLP_Project.ipynb` notebook from the Files panel
5. Run all cells sequentially
6. At the end of execution, a Streamlit app URL will be displayed for interactive exploration

B.2 Local Installation

For local execution:

Listing 6: Local Installation Steps

```
# Clone repository
git clone https://github.com/Seymurhh/
   Search_engine_educational_project_NLP
cd Final
```

```

# Install dependencies
pip install -r requirements.txt

# Download NLTK data
python -c "import nltk; nltk.download('punkt'); nltk.download('stopwords'); nltk.download('wordnet')"

# Run Streamlit application
streamlit run app.py
```