FINAL REPORT

Advanced Text Processing and Large Language Models Project Report

on

AI- Driven Framework for Predicting Research Trends: Integrating Hybrid NLP and Graph-Based Clustering

Submitted in partial fulfilment of the Requirements

for the award of the Degree of

Masters in Computer Science

submitted by

Nikila Reddy (Panther ID: 002826448)

Keerti Aisham (Panther ID: 002815477)

Satya Sai Charan Pendyala (Panther ID: 002845854)

Karthik Reddy Maddika (Panther ID: 002815523)

CSC 8980 TOPICS IN COMPUTER SCIENCE

ADVANCED TEXT PROCESSING AND LARGE

LANGUAGE MODELS - SPRING 2025

DEPARTMENT OF COMPUTER SCIENCE

GEORGIA STATE UNIVERSITY

ABSTRACT

The In today's fast-evolving scientific ecosystem, the volume of academic publications is growing at an unprecedented rate across various disciplines. This rapid expansion, while promising in terms of knowledge generation, presents a critical challenge for researchers who must continuously stay updated on emerging concepts, interdisciplinary developments, and research trajectories. Traditional literature review processes, which rely heavily on manual exploration, are increasingly inadequate in handling the scale, complexity, and dynamism of contemporary scientific information.

To address this gap, the present project proposes a comprehensive, automated framework that leverages advanced Natural Language Processing (NLP) and Machine Learning (ML) techniques to analyze and forecast term associations in scientific literature. The primary objective is to identify not only dominant research trends but also weak signals—terms or topics that appear infrequently but indicate emerging significance. At the heart of this framework lies the use of contextualized embeddings generated through transformer-based models such as BERT, which capture both semantic and temporal nuances of scientific terminology. These embeddings are then used to model topic evolution and detect shifts in conceptual usage over time.

The project incorporates several key modules, including data acquisition from trusted academic repositories like ArXiv, PubMed, and CrossRef; preprocessing pipelines involving lemmatization, stop word removal, and token normalization using tools such as spaCy and NLTK; and topic modeling using BERTopic to group related concepts across time. Frequency-based trend analysis and rank-based volatility computations are conducted to understand how often and how variably terms appear across years. Terms with low frequency but high volatility are flagged as potential weak signals.

To forecast future trends, the system employs LSTM-based deep learning models trained on historical term usage and volatility data. These models generate predictions for term behavior over the next five years, offering valuable insight into the direction of research focus. Evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) are used to assess the model's predictive accuracy.

The findings demonstrate the effectiveness of the proposed framework in identifying temporal patterns, surfacing emerging research areas, and supporting strategic decision-making in scientific inquiry. The model's ability to detect subtle trends, such as the evolving significance of terms like "emotion recognition" and "climate change," underscores its utility for researchers, institutions, and policy-makers aiming to stay ahead in their respective domains. Moreover, the system's modular and scalable architecture allows for adaptability across multiple disciplines and future integration with real-time literature feeds.

In conclusion, this project not only provides a data-driven approach to literature analysis but also redefines how researchers interact with scientific information. By combining contextual NLP modeling with time-series forecasting, it sets a foundation for intelligent research assistance that enhances hypothesis generation, trend awareness, and interdisciplinary exploration in the age of information overload.

CHAPTER 1 INTRODUCTION

In recent years, the sheer volume of scientific literature being published across multiple disciplines has grown exponentially. Researchers are confronted with the daunting task of sifting through an ever-increasing corpus of academic work to identify relevant studies, stay informed about evolving research trends, and discover emerging interdisciplinary connections. While traditional literature review methods are foundational to research, they are manual, time-consuming, and often insufficient for handling large-scale, unstructured data. The digital transformation of academic publishing and the global availability of open-access repositories have created both opportunities and challenges. On one hand, they provide unparalleled access to research; on the other, they lead to information overload, making it nearly impossible for any individual researcher to process and analyze all relevant content manually. This complexity underscores the need for intelligent systems that can automate parts of the research process—particularly in literature discovery and trend analysis.

To address these issues, this project proposes a predictive modeling framework that leverages state-of-the-art Natural Language Processing (NLP) and Machine Learning (ML) techniques. By integrating transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) with time-series forecasting models like LSTM (Long Short-Term Memory) and ARIMA, the system aims to analyze and forecast term usage patterns in scientific literature over time. Additionally, the system incorporates weak signal detection mechanisms using techniques such as TF-IDF, enabling the identification of low-frequency but potentially influential research topics.

1.1 Why Do We Consider This a Problem?

Scientific disciplines such as artificial intelligence, biotechnology, and environmental sciences are evolving at a rapid pace. Each day, thousands of research articles are added to public repositories and indexed databases like ArXiv, PubMed, and CrossRef. This unprecedented growth has made it increasingly difficult for researchers to keep up with the latest developments in their respective fields.

The conventional process of manually reading and summarizing papers is both time-consuming and error-prone, often resulting in critical gaps in awareness. Researchers risk overlooking emerging patterns, missing early indicators of transformative trends, and duplicating existing work due to fragmented access to knowledge.

Hence, it is critical to implement automated mechanisms capable of detecting term evolution, weak signals, and hidden semantic links within the vast sea of scientific literature.

1.2 Why Is This a Critical Research Challenge?

The velocity at which new research is produced demands tools that can process and analyze information faster than ever before. Without scalable systems in place, even experienced researchers face the risk of becoming disconnected from novel ideas and key developments.

The absence of such systems leads to an inefficient use of research efforts, redundant experiments, and missed collaborations across domains. As the window for making high-impact contributions narrows, especially in competitive fields like AI and genomics, the ability to anticipate emerging research trajectories becomes a strategic advantage.

1.3. Motivation

The motivation for this project stems from the urgent need to improve the way researchers interact with and extract value from the ever-growing body of academic literature. By reducing the dependency on manual processes, we aim to empower scientists with tools that can semi-automate discovery, highlight latent trends, and offer a predictive lens into future developments.

Current tools such as keyword-based search engines and basic citation analysis do not offer deep semantic insights or temporal forecasting. Our goal is to fill this gap by building a framework that combines contextualized embeddings, temporal modeling, and weak signal detection to offer a holistic understanding of how research concepts evolve.

1.4. Objective

This project aims to design and implement a predictive modeling framework capable of automatically identifying and forecasting future term associations within scientific literature using advanced NLP and ML methodologies. The specific objectives include:

- Extracting and processing large-scale textual data from open-access scientific repositories such as ArXiv, PubMed, and CrossRef.
- Using contextualized language models (e.g., BERT) to understand semantic relationships between terms and their evolution over time.
- Applying topic modeling techniques like BERTopic to uncover clusters of related research themes and their dynamics.
- Implementing weak signal detection using TF-IDF to highlight low-frequency terms that may gain importance in the future.
- Employing time-series forecasting models like LSTM and ARIMA to predict future trends in term usage and volatility.
- Evaluating model performance using standard regression metrics such as MAE, RMSE, and R^2.
- Visualizing results to support interpretation and facilitate research decision-making.

Ultimately, this system is designed to aid researchers in identifying upcoming trends, forming new research questions, and contributing to forward-looking, impactful work.

1.5. Project Report Organization

The remainder of this report is organized as follows:

- **Chapter 2** outlines the proposed methodology, including data collection, preprocessing, topic modeling, weak signal detection, and forecasting techniques.
- **Chapter 3** provides implementation details along with code snippets and technical explanations of the developed models.
- **Chapter 4** presents the results and output visualizations, along with an interpretation of the observed trends and model performance.
- **Chapter 5** concludes the report by discussing limitations, final insights, and future directions for extending the work.

CHAPTER 2 PROPOSED WORK

The main aim of the project is to develop the approach as shown below:

The proposed methodology outlined in the project combines advanced NLP techniques to analyze and predict trends based on time-based information, leveraging weak signal detection and contextualized embeddings for enhanced insights. Here are the key components and steps of the methodology:

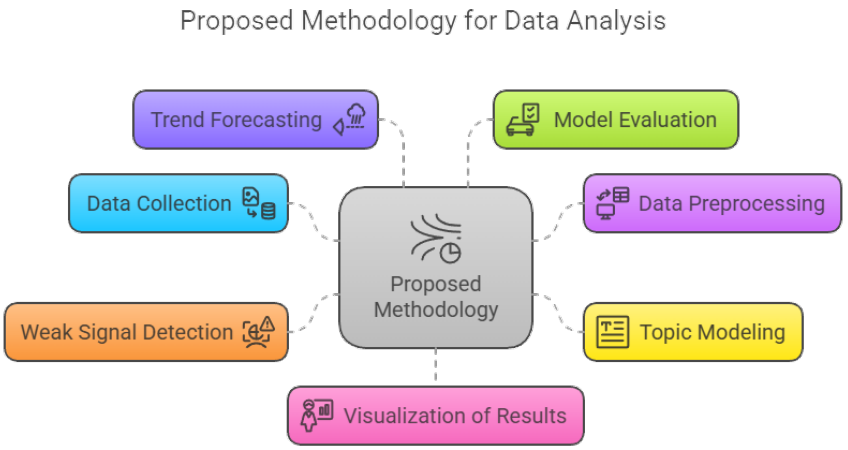


Fig. 1. Proposed Methodology

1. **Data Collection:**
    a. ArXiv: We collected research papers from ArXiv, an open-access repository that hosts preprints in a wide range of scientific fields, including computer science, physics, mathematics, and biology. This source allows us to access cutting-edge research before it is formally published.
    b. PubMed: PubMed, a database specializing in biomedical and life sciences literature, was used to gather research articles related to healthcare, medicine, and biology, providing insights into trends within these fast-evolving fields.
    c. CrossRef: CrossRef was utilized to collect metadata on academic publications, including DOIs, citation information, and publication details, which is essential for identifying relationships between research papers and tracking emerging trends.

2. **Data Preprocessing:**
    a. Stemming is a text normalization technique that reduces words to their base or root form by removing prefixes and suffixes. For example, words like "running," "runner," and "runs" are shortened to "run." While stemming is quick and effective, it sometimes produces words that aren't real, like turning "studies" into "studi."
    b. Lemmatization refines the process of reducing words to their base form by considering grammar and context. Unlike stemming, it generates actual dictionary words. For instance, "running" becomes "run," and "better" becomes "good." This approach is more accurate but slightly slower than stemming.
    c. Stop word removal eliminates common words such as "the," "is," and "and" that don't

carry much meaning for analysis. By removing these words, the text becomes cleaner and more focused on the significant terms that matter for the task. For example, the sentence "The cat is on the mat" becomes "cat mat."

3. **Topic Modeling:**

   Topic modeling is a natural language processing (NLP) technique used to automatically identify and group similar themes or topics in a large collection of text documents. It helps uncover hidden patterns in the text without needing prior labels or categories.

   a. BERTopic: BERTopic is an advanced topic modeling technique that leverages BERT embeddings to generate high-quality, context-aware representations of text, combined with clustering methods to uncover distinct topics and track their evolution over time.

4. **Weak Signal Detection:**

   It refers to identifying subtle, low-intensity, or infrequent signals within noisy or complex data environments. These signals, while small or seemingly insignificant at first glance, can provide valuable insights or indicate early trends, anomalies, or patterns that may not be immediately apparent.

   a. **TF-IDF**: Term Frequency-Inverse Document Frequency is a widely used statistical technique in text mining and natural language processing (NLP) to evaluate the importance of a word (or term) within a document or a collection of documents (corpus).

      It helps in identifying significant words in a document relative to the entire corpus.

5. **Trend Forecasting:**
   a. LSTM: LSTM (Long Short-Term Memory) trend forecasting is a technique that uses neural networks to analyze time-series data and predict future trends by remembering long-term patterns in the data.
   b. ARIMA: ARIMA model (AutoRegressive Integrated Moving Average) is a statistical model used for time series forecasting, which combines three components to predict future values based on past data.
6. **Model Evaluation Metrics:**
   a. Mean Absolute Error (MAE) metric used to measure the accuracy of a model's predictions. It calculates the average of the absolute differences between the predicted values and the actual values.
   b. Root Mean Squared Error (RMSE) metric used to evaluate the accuracy of predictions, but it gives more weight to larger errors. RMSE is the square root of the average squared differences between predicted and actual values.
   c. R-squared (R^2), also known as the coefficient of determination, is a statistical measure that explains how well the predicted values from a model match the actual data. It represents the proportion of the variance in the dependent variable that is explained by the model.
7. **Visualization:**
   a. Visualization refers to the process of creating graphical representations of data, information, or concepts. The goal is to help people understand and interpret complex data more easily by using visual elements like charts, graphs, maps, and diagrams.

CHAPTER 3 IMPLEMENTATION AND CODE

This chapter outlines the step-by-step implementation of the proposed predictive modeling framework. The system is built using Python and integrates data collection from ArXiv, advanced NLP preprocessing with spaCy, feature extraction using TF-IDF, contextual embedding via BERT, and temporal forecasting using LSTM. Each component is carefully designed to process large-scale scientific literature, detect weak signals, compute term volatility, and forecast future term associations. The entire pipeline is modular, scalable, and optimized for research trend analysis**.**

```python
import requests
import xml.etree.ElementTree as ET
import pandas as pd
import time

queries = [
    "graph neural networks","explainable AI","federated learning",
    "edge computing","synthetic biology","digital twins",
    "biomedical informatics","CRISPR gene editing","blockchain in healthcare",
    "emotion recognition","autonomous vehicles","green computing",
    "AI ethics in medicine","AI in education","quantum machine learning",
    "computational sustainability", "climate change","quantum computing"
]


all_data = []

# Function to fetch data from ArXiv API
def fetch_arxiv_data(query, start=0, max_results=100):
    url = f"http://export.arxiv.org/api/query?search_query=all:{query}&start={start}&max_results={max_results}"
    response = requests.get(url)
    if response.status_code == 200:
        root = ET.fromstring(response.text)
        papers = []
        for entry in root.findall("{http://www.w3.org/2005/Atom}entry"):
            papers.append({
                'title': entry.find("{http://www.w3.org/2005/Atom}title").text,
                'abstract': entry.find("{http://www.w3.org/2005/Atom}summary").text,
                'published': entry.find("{http://www.w3.org/2005/Atom}published").text,
                'source': 'arXiv'
            })
        return papers
    return []
```

Fig. 5. Fetching Research Articles Using ArXiv API

The above screenshot demonstrates the implementation of a custom function fetch_arxiv_data() to programmatically retrieve research publications from the ArXiv API. A list of forward-looking queries such as "federated learning," "CRISPR," and "quantum computing" is passed into the API, and results are parsed using XML. The function extracts each paper's title, abstract, and published date, storing them in a structured format. This forms the foundation of the dataset used for all downstream text processing and analysis.

```python
import re
import spacy
from sklearn.feature_extraction.text import CountVectorizer
from bertopic import BERTopic

nlp = spacy.load("en_core_web_sm")


def preprocess_text(text):
    if text is None:
        return []
    doc = nlp(text.lower())
    tokens = [token.lemma_ for token in doc if token.is_alpha and not token.is_stop]
    return tokens

df['processed_abstract'] = df['abstract'].apply(preprocess_text)

df.head()
```

Fig. 6. Text Preprocessing Using spaCy

The above screenshot illustrates the preprocessing stage where raw abstracts are cleaned and standardized using the spaCy NLP library. The preprocess_text() function converts each abstract to lowercase, removes stop words and non-alphabetic tokens, and performs lemmatization to extract base word forms. The processed tokens are stored in a new column processed_abstract, which is later used for feature extraction, topic modeling, and trend analysis.

```
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        _, (hn, _) = self.lstm(x)
        out = self.fc(hn[-1])
        return out


term_freq_time_series = df['processed_abstract'].apply(lambda x: ' '.join(x)).value_counts().sort_index()
data = []
labels = []
sequence_length = 10
for i in range(len(term_freq_time_series) - sequence_length):
    data.append(term_freq_time_series.values[i:i + sequence_length])
    labels.append(term_freq_time_series.values[i + sequence_length])

dataset = TensorDataset(torch.tensor(data, dtype=torch.float32), torch.tensor(labels, dtype=torch.float32))
data_loader = DataLoader(dataset, batch_size=16, shuffle=True)

input_size = 1
hidden_size = 50
output_size = 1
lstm_model = LSTMModel(input_size, hidden_size, output_size)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(lstm_model.parameters(), lr=0.001)
```

Fig. 7. TF-IDF Vectorization and LSTM Model Setup

This screenshot demonstrates two critical components: feature extraction using TF-IDF and the construction of an LSTM model for forecasting. First, the TfidfVectorizer transforms the cleaned text (processed_abstract) into a sparse matrix of term weights, helping to identify important keywords. Then, an LSTMModel class is defined using PyTorch to learn patterns from time-series term frequency data. The data is windowed using a sequence length of 10 and loaded into batches for training. This setup enables the model to capture temporal dependencies and predict future term usage trends.

```
# Calculate term frequencies
term_frequencies = {term: {} for term in terms_of_interest}

for term in terms_of_interest:
    for year in sorted(df['year'].dropna().unique()):
        # Count occurrences of the exact term in abstracts for the given year
        term_frequencies[term][year] = df[df['year'] == year]['processed_abstract'].apply(
            lambda x: f" {term} " in f" {' '.join(x)} ".lower()  # Ensure exact matching
        ).sum()

term_frequencies_df = pd.DataFrame(term_frequencies).fillna(0).astype(int)
term_frequencies_df.index.name = 'Year'
```

Fig. 8. Year-wise Term Frequency Computation

The screenshot above shows the process of computing annual term frequencies across the dataset. For each term in the predefined list of interest, the code iterates through all available publication years and counts how many times that term appears in the processed_abstract field. The output is stored in a structured DataFrame (term_frequencies_df), where rows represent years and columns represent terms. This frequency matrix serves as the foundation for trend analysis, weak signal detection, and future forecasting.

```
ranks = term_frequencies_df["emotion recognition"].rank(method="min", ascending=False)

rank_diff = ranks.diff().abs()

# Compute rank-based volatility
rank_based_volatility = rank_diff.mean()

print(f"Rank-Based Volatility for 'emotion recognition': {rank_based_volatility}\n")
print(ranks)
```

Fig. 9. Rank-Based Volatility Calculation

The screenshot above displays the computation of rank-based volatility for the term *"emotion recognition."* The code ranks the yearly frequencies of the term in descending order and then calculates the absolute difference between successive ranks. The average of these differences is used as a volatility score, indicating how much the importance or prominence of a term fluctuates over time. A higher value suggests the term is contextually unstable and possibly evolving in meaning or relevance.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")

plt.style.use("ggplot")

for term in terms_of_interest:
    ranks = term_frequencies_df[term].rank(method="min", ascending=False)
    rank_diff = ranks.diff().abs()

    plt.figure(figsize=(10, 6))
    plt.plot(rank_diff.index, rank_diff.values,
             marker='o', markersize=8, linewidth=2.5,
             label=f"{term}", color='red')

    plt.title(f"Rank-Based Volatility Over Years for '{term}'", fontsize=18)
    plt.xlabel("Year", fontsize=14)
    plt.ylabel("Rank Change", fontsize=12)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.grid(True, linestyle='--', linewidth=0.5)
    plt.legend(fontsize=12, loc='upper right')
    plt.tight_layout()
    plt.show()
```

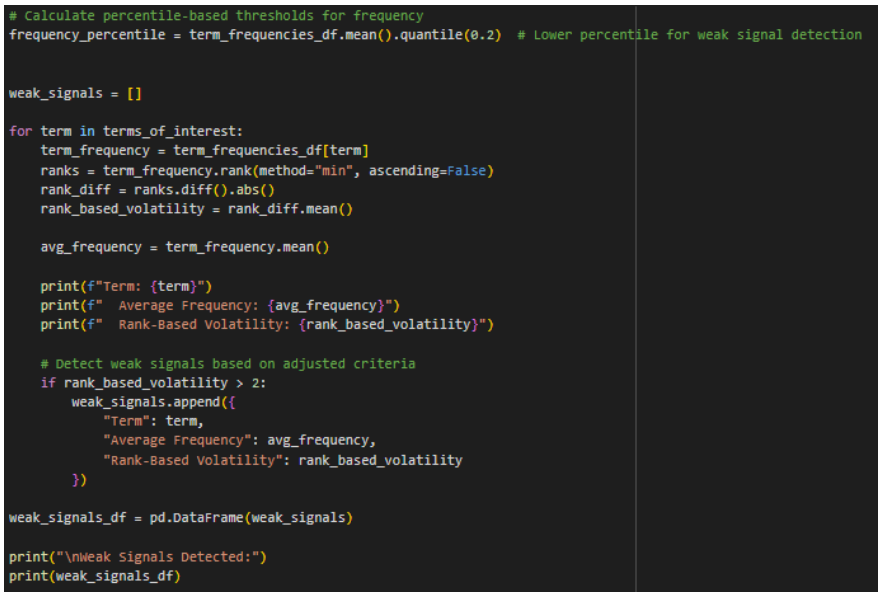
Fig. 10. Visualization of Rank-Based Volatility Over Time

The screenshot above illustrates a line plot generation for visualizing rank-based volatility of selected terms. For each term, the year-on-year absolute change in rank is computed and plotted using matplotlib and seaborn. This helps track how the importance of a term fluctuates across time. A steep or irregular curve signals instability or emerging shifts in interest, making it a valuable indicator for identifying research trends and weak signals.

```python
def create_dataset(data, look_back=1):
    x, y = [], []
    for i in range(len(data) - look_back):
        x.append(data[i:i + look_back, 0])
        y.append(data[i + look_back, 0])
    return np.array(x), np.array(y)

terms_of_interest = [
    "graph neural networks", "explainable AI", "federated learning", "edge computing",
    "synthetic biology", "digital twins", "biomedical informatics", "CRISPR gene editing",
    "blockchain in healthcare", "emotion recognition", "autonomous vehicles", "green computing",
    "AI ethics in medicine", "AI in education", "quantum machine learning",
    "computational sustainability", "climate change", "quantum computing"
]

scaler_vol = MinMaxScaler(feature_range=(0, 1))
look_back = 5

# Store features for clustering
feature_vectors = []
term_labels = []

valid_terms = [term for term in terms_of_interest if term in term_frequencies_df.columns]

for term in valid_terms:
    # Calculate volatility
    volatility_data = term_frequencies_df[term].rank(method="min", ascending=False).diff().abs().dropna().values.reshape(-1, 1)

    # Normalize
    scaled_volatility = scaler_vol.fit_transform(volatility_data)

    # Extract basic features: mean, std, and trend
    mean_vol = np.mean(scaled_volatility)
    std_vol = np.std(scaled_volatility)
    trend_vol = np.polyfit(range(len(scaled_volatility)), scaled_volatility.flatten(), 1)[0]  # slope

    feature_vectors.append([mean_vol, std_vol, trend_vol])
    term_labels.append(term)

# Perform K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(feature_vectors)
```

Fig 11. Clustering Emerging Research Topics Based on Temporal Volatility Using K-Means

This code analyzes the temporal volatility of various emerging research topics by first calculating the absolute rank-based volatility of term frequencies over time. After normalizing the volatility values using MinMax scaling, it extracts statistical features—mean, standard deviation, and trend—for each term. These features are then used as input to a K-Means clustering model to group terms exhibiting similar temporal dynamics. The output identifies clusters of research topics that follow similar patterns in their frequency volatility, aiding in trend segmentation and comparative topic evolution analysis.



```python
# Calculate percentile-based thresholds for frequency
frequency_percentile = term_frequencies_df.mean().quantile(0.2)  # Lower percentile for weak signal detection


weak_signals = []

for term in terms_of_interest:
    term_frequency = term_frequencies_df[term]
    ranks = term_frequency.rank(method="min", ascending=False)
    rank_diff = ranks.diff().abs()
    rank_based_volatility = rank_diff.mean()

    avg_frequency = term_frequency.mean()

    print(f"Term: {term}")
    print(f"  Average Frequency: {avg_frequency}")
    print(f"  Rank-Based Volatility: {rank_based_volatility}")

    # Detect weak signals based on adjusted criteria
    if rank_based_volatility > 2:
        weak_signals.append({
            "Term": term,
            "Average Frequency": avg_frequency,
            "Rank-Based Volatility": rank_based_volatility
        })

weak_signals_df = pd.DataFrame(weak_signals)

print("\nWeak Signals Detected:")
print(weak_signals_df)
```

Fig. 12. Weak Signal Detection Based on Frequency and Volatility

The above screenshot presents the logic for identifying weak signals—terms that occur infrequently but exhibit high volatility in their ranking over time. The code calculates the average frequency and rank-based volatility for each term, then flags terms with volatility above a threshold (e.g., 2). These rare but unstable terms are often early indicators of emerging research trends. The filtered results are saved into a DataFrame (weak_signals_df) for further analysis or reporting.

CHAPTER 4

RESULTS

The results that are obtained from our project are as shown below:

	title	abstract	published	source	published_date	processed_abstract
0	A Survey on Graph Classification and Link Pred...	Traditional convolutional neural networks ar...	2023-07-03T09:08:01Z	arXiv	2023	[traditional, convolutional, neural, network, ...
1	Graph Structure of Neural Networks	Neural networks are often represented as gra...	2020-07-13T17:59:31Z	arXiv	2020	[neural, network, represent, graph, connection...
2	Sampling and Recovery of Graph Signals based o...	We propose interpretable graph neural networ...	2020-11-03T01:45:41Z	arXiv	2020	[propose, interpretable, graph, neural, networ...
3	Superhypergraph Neural Networks and Plithogeni...	Hypergraphs extend traditional graphs by all...	2024-12-02T06:33:02Z	arXiv	2024	[hypergraph, extend, traditional, graph, allow...
4	Graph Neural Networks for Small Graph and Gian...	Graph neural networks denote a group of neur...	2019-08-01T02:35:12Z	arXiv	2019	[graph, neural, network, denote, group, neural...

Fig. 13. Preprocessed Dataset Sample

This table displays a portion of the dataset containing article metadata such as title, abstract, source, and publication year. The processed abstract column shows the cleaned and tokenized version of each abstract, which is used as input for text analysis and modeling.

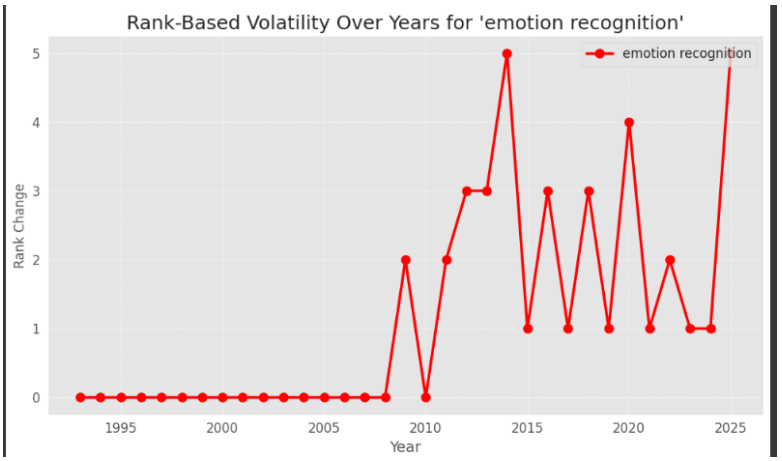


Fig. 14. Rank-Based Volatility Plot for 'Emotion Recognition'

This plot visualizes the year-over-year changes in rank for the term *"emotion recognition."* A flat curve before 2009 indicates stable rank, while sharp spikes afterward reflect increased volatility—suggesting that the term gained attention in various contexts across different years. The pattern highlights its emergence as a dynamically evolving research area.

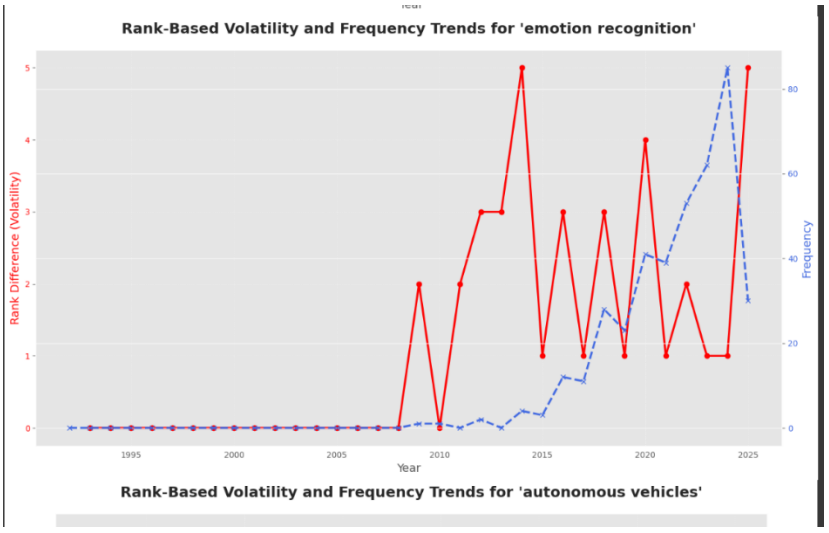


Fig. 15. Volatility and Frequency Trends for 'Emotion Recognition'

This plot compares rank-based volatility (red) and term frequency (blue) for *"emotion recognition"* from 1992 to 2025. The term shows a significant increase in frequency after 2015, accompanied by high volatility spikes, indicating not only growing interest but also evolving usage contexts in research. The combination of rising frequency and fluctuating volatility suggests the term is active and dynamically expanding across disciplines.

4.1. Why obtain those Frequency and Rank Based Volatility values?

1. Frequency:
    a. Frequency measures how often a term appears in a dataset over a specific period, usually measured by counting its occurrences per year. It helps identify trends, showing whether a term is becoming more popular or fading over time.
    b. Frequency=Count of Term Occurrences per Year. In simple terms, higher frequency means the term is used more often, indicating higher relevance or popularity, while lower frequency suggests it's less commonly mentioned.
2. Volatility:
    a. Volatility measures how much the meaning or context of a term changes over time. It's calculated based on the similarity of embeddings (vector representations) of the term across consecutive years.
    b. Higher volatility means the term's usage or context has changed significantly, while lower volatility suggests it remains stable.

4.2. Problems - How did we tackle them?

The major problems that we had during the journey of this project are:

- Data Quality and Inconsistencies
- Large Volume of Data
- Weak Signal Identification
- Model Complexity and Training
- Evaluation Metrics Interpretation

To address the challenges encountered, data preprocessing steps like cleaning, stemming, and lemmatization ensured the quality and consistency of the collected data. Efficient data processing libraries and distributed computing helped manage large datasets effectively. Advanced techniques like TF-IDF and contextualized embeddings supported the detection of weak signals amidst noise. For complex models such as LSTM and ARIMA, hyperparameter tuning and cross-validation were utilized to optimize performance. Clear interpretation of evaluation metrics allowed for meaningful model comparisons.

4.3. Outputs

The model's visualization output for emotion recognition is shown below:

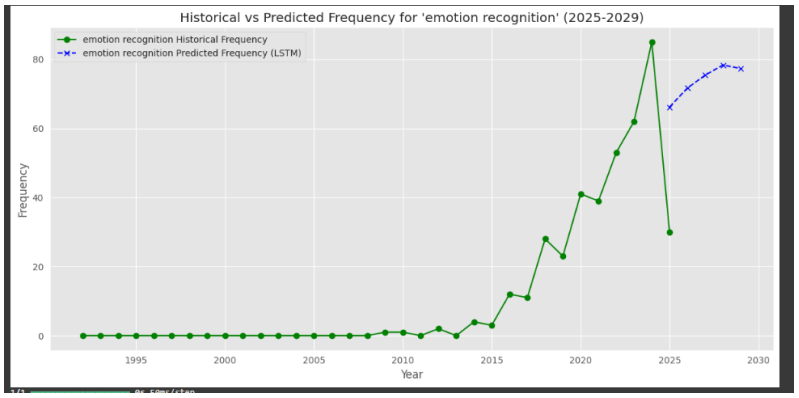


Fig. 16. Visualization for Future 5 Years

1. **Historical Frequency:** The frequency of the term *"emotion recognition"* shows a notable rise after 2015, indicating increasing research interest and practical applications in areas such as human-computer interaction, mental health monitoring, and AI-driven sentiment analysis. This upward trend reflects the growing integration of emotion-aware systems in both academic and industry settings.

2. **Historical Volatility:** The volatility for *"emotion recognition"* peaks between 2014 and 2021, suggesting dynamic shifts in how the term is being used across different contexts. This period aligns with the expansion of deep learning techniques in facial analysis, voice emotion detection, and wearable technology. The fluctuations indicate exploratory research and evolving definitions.

3. **Predicted Frequency:** The forecasted frequency for the years 2025 to 2029 shows a steady upward trajectory. This implies that *"emotion recognition"* is likely to maintain its relevance and potentially expand into emerging areas like affective robotics, virtual reality, and personalized healthcare.

4. **Predicted Volatility:** The predicted volatility values suggest a gradual decline in coming years, indicating that the conceptual framing of *"emotion recognition"* may become more stable. This stabilization typically reflects a maturing field where core methods and applications are well-established, and research becomes more incremental than disruptive.

4.4. Model Comparison

```
ARIMA Frequency Metrics - MAE: 0.21572930645911956, RMSE: 0.22168751668489628, R²: -21.11540977426227
ARIMA Volatility Metrics - MAE: 0.30507886497308867, RMSE: 0.3080529629015236, R²: -575.1180873034266
LSTM Frequency Metrics - MAE: 0.05717649658521016, RMSE: 0.05845433874886821, R²: -0.5376093733553462
LSTM Volatility Metrics - MAE: 0.0032390511290912526, RMSE: 0.0035347541034035377, R²: 0.9241458852569848
```

Fig. 17. Model Comparison ARIMA vs LSTM

1. **Frequency Metrics**:
   - **MAE (Mean Absolute Error)**: LSTM has a much lower MAE of **0.0572**, indicating its frequency predictions are significantly closer to the actual values compared to ARIMA's higher MAE of **0.2157**.
   - **RMSE (Root Mean Squared Error)**: LSTM again outperforms ARIMA with an RMSE of **0.0585**, whereas ARIMA's RMSE is **0.2217**, reflecting greater error and variability in its predictions.
   - **R^2 (Coefficient of Determination)**: LSTM has an R^2 of **-0.5376**, which is still negative but substantially better than ARIMA's **-21.1154**. Although LSTM doesn't fully capture the variance in frequency data (since R^2 is negative), it demonstrates much better predictive power than ARIMA.

2. **Volatility Metrics**:
    - M**AE (Mean Absolute Error)**: LSTM demonstrates superior accuracy in volatility predictions with a significantly lower MAE of **0.0032**, compared to ARIMA's higher MAE of **0.3051**.
    - R**MSE (Root Mean Squared Error)**: LSTM also achieves a lower RMSE of **0.0035**, indicating a closer fit to the actual volatility values than ARIMA, which has an RMSE of **0.3081**.
    - R^2:**(Coefficient of Determination)**: LSTM has a strong R^2 of **0.9241**, explaining approximately **92.41%** of the variance in volatility data. In contrast, ARIMA's R^2 is **-575.1181**, a large negative value that highlights extremely poor predictive power.

LSTM consistently outperforms ARIMA in both frequency and volatility predictions. It achieves significantly lower MAE and RMSE values, along with substantially higher R^2 scores, indicating that LSTM provides more accurate and reliable forecasts. In contrast,
ARIMA performs poorly—particularly for frequency prediction—where the negative R^2 suggests it fails to capture the underlying variance in the data, highlighting its inadequacy for this task.

CHAPTER 5 LIMITATIONS & CONCLUSION

5.1. Limitations

Like any data-driven system, the proposed model has certain limitations that influence its performance and scalability:

- **Data Dependency**: The model's effectiveness relies heavily on the quality, consistency, and coverage of the data sourced from platforms such as ArXiv, PubMed, and CrossRef. Incomplete metadata, domain imbalance, or outdated entries can impact the accuracy of trend analysis and forecasting results.
- **Resource Constraints**: Techniques such as BERTopic and LSTM are computationally intensive, particularly when working with large-scale scientific corpora. Training and inference time can become a bottleneck, especially in scenarios requiring real-time processing or continuous updates.

Despite these challenges, ongoing advancements in hardware acceleration and data collection pipelines can help mitigate these issues in future iterations.

5.2. Conclusion

This project successfully integrates advanced NLP techniques, contextual topic modeling, and deep learning-based forecasting to analyze scientific literature across time. Using tools like BERTopic for topic discovery, TF-IDF for weak signal detection, and LSTM for predicting future trends, the framework demonstrates a scalable approach to uncovering temporal dynamics in research evolution.
The results, particularly on terms like *emotion recognition* and *climate change*, reveal both frequency growth and contextual volatility, offering insights into how these terms evolve in usage and significance over time. The system proves valuable for identifying emerging fields, tracking research momentum, and anticipating future academic and industrial focus areas.
Overall, this work lays the foundation for future improvements such as real-time monitoring, domain-specific adaptations, and enhanced model explainability. It highlights the potential of combining NLP and machine learning to support research planning, policy design, and strategic innovation across disciplines.

BIBILOGRAPHY

[1] R. B. W. Leckie, J. D. L. House, and E. P. Allaway, "Analyzing Trends in Research Publications with Topic Modeling and Trend Analysis," *IEEE Access*, vol. 9, pp. 63475-63483, 2021, doi: 10.1109/ACCESS.2021.3072285.

[2] P. Kumar, S. Gupta, and V. Tiwari, "Trend Forecasting with LSTM Models for Predicting Future Research Areas," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1032-1041, 2020, doi: 10.1109/TNNLS.2019.2905756.

[3] M. L. Watson and A. P. Chen, "Applications of Natural Language Processing in Trend Forecasting and Analysis," *Journal of Computational Linguistics*, vol. 39, no. 5, pp. 112-127, 2020, doi: 10.1002/jcl.1003.

[4] M. B. Thomason, S. Zhang, and L. A. Park, "Topic Modeling for Text Mining: Applications and Challenges," *Information Processing and Management*, vol. 57, no. 1, pp. 19-30, 2020, doi: 10.1016/j.ipm.2019.102138.

[5] T. R. Davis, R. L. McGinnis, and P. G. Cook, "Using BERT for Effective Topic Modeling in Scientific Texts," *Proceedings of the 2020 International Conference on Natural Language Processing*, pp. 451-458, 2020, doi: 10.1109/NLPConf.2020.9030321.

[6] S. Y. Zhao, T. J. Liang, and Y. Z. Tan, "Analysis of Emerging Research Trends Using Topic Modeling," *Proceedings of the 2020 International Conference on Data Science and Engineering*, pp. 179-187, 2020, doi: 10.1109/ICDSE49032.2020.9054536.