Automated Framework for Research Paper Publishability Assessment and Conference Selection

KDSH:Team Admirals

January 14, 2025

Colab Notebook: Our Code: Colab Notebook Link

This report presents an innovative automated framework that addresses two critical challenges in academic publishing: assessing paper publishability and recommending appropriate conferences. Using a combination of advanced natural language processing techniques, including SciBERT, LDA topic modeling, and neural networks, The framework processes research papers through a multi-stage pipeline, leveraging both semantic understanding and contextual relevance to provide justified recommendations. Our solution significantly reduces the manual effort required in the paper submission process while maintaining high accuracy and providing transparent justifications for its decisions.

1 Problem Definition and Objectives

The primary objective of this work is twofold:

1. **Publishability Assessment:** Automatically classify research papers based on their content and quality, determining if they are suitable for publication.

2. **Conference Selection:** After classifying papers as "Publishable," recommend the most appropriate academic conference for submission, with a detailed justification based on the paper's content.

.

2 Literature Review

Most of the works done for this are kind of trial and error but for implementing KNN on LDA's Perplexity data we took idea from Research paper classification systems based on TF-IDF and LDA schemes Sang Woom Kim and Joon Mil Gil

3 Data Preprocessing

The dataset provided contains 150 research papers, with each paper consisting of an abstract, keywords, and a detailed body. The preprocessing steps include:

- Extracting the research papers from given drive using Pathway's Data Connectors
- Converting the data into simple Pandas dataframe
- Splitting whole paper into section by implementing a regex solution:The `structure_text` function processes extracted text by detecting sections like *Abstract* and numbered headings. It uses regular expressions to segment text, appending content to the appropriate section in a dictionary. The function organizes the document into structured, easily navigable sections.
- Then finally doing text preprocessing by removing irrelevant part of speech (POS) using spaCy module.

4 Task 1

In this task, we employ various machine learning techniques to classify a documents to publishable or non publishable, beginning with a pretrained SciBERT model and proceeding through several stages of processing, topic modeling, clustering, and classification. Below, we detail the steps involved in the process.

4.1 Step 1: Pretrained SciBERT Model

The first step in this task is to use a pretrained SciBERT model, a transformer-based model specialized for scientific text, for text classification. This serves as a benchmark model for our task, providing an initial performance measure against which other models will be compared.

4.2 Step 2: Topic Modeling with Latent Dirichlet Allocation (LDA)

Next, we apply Latent Dirichlet Allocation (LDA) for unsupervised topic modeling. Given a document d_i, LDA provides a distribution over topics $p(t_j|d_i)$, where t_j represents topic j and $p(t_j|d_i)$ is the probability that document d_i is related to topic t_j. The process involves the following steps:

$$X = \text{CountVectorizer}(d_i) \quad \text{(document-term matrix)}$$

$$\mathbf{z}_i = \text{LDA}(X_i) \quad \text{(topic distribution for document } d_i\text{)}$$

Here, X_i is the document-term matrix for each document, and $\mathbf{z}_i$ is the topic distribution output by LDA for each document.

4.3 Step 3:Stopwords Removal

$$\text{cleaned}(d_i) = \text{remove\_stopwords}(\text{tokenize}(d_i))$$

4.4 Step 4: Extracting Dominant Topics

After applying LDA, we extract the dominant topic for each document d_i by finding the topic with the highest probability $\mathbf{z}_{ij}$ for each document:

$$\text{dominant\_topic}(d_i) = \arg\max_j(\mathbf{z}_{ij})$$

The dominant topic for each document is stored for subsequent analysis.

4.5 Step 5: K-Means Clustering Based on Topic Distributions

Using the topic distributions $\mathbf{z}_i$ obtained from LDA, we apply **K-Means clustering** to group similar documents together

4.6 Step 6: Extracting Proper Nouns and Calculating Cosine Similarity

In this step, we extract proper nouns from each document and calculate the cosine similarity between them to analyze their relationships. For two vectors $\mathbf{a}$ and $\mathbf{b}$, the cosine similarity $\cos(\theta)$ is given by:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

4.7 Step 7: Cosine Similarity Among Topic Keywords

We also compute the cosine similarity between topic keywords extracted from LDA. Given the vector representation of topic keywords $\mathbf{v}_k$ for topic k, we calculate the similarity between the keywords for topics k_1 and k_2 as:

$$\cos(\theta_{k_1,k_2}) = \frac{\mathbf{v}_{k_1} \cdot \mathbf{v}_{k_2}}{\|\mathbf{v}_{k_1}\|\|\mathbf{v}_{k_2}\|}$$

This similarity helps us understand the relationships between the topics themselves.

4.8 Step 8: SciBERT Embeddings and Neural Network Probabilities

Next, we use **SciBERT embeddings** to represent the documents in vector space, and then apply a **Neural Network** to obtain the probabilities of each document belonging to a particular class. Let $\mathbf{e}_i$ be the SciBERT embedding for document d_i, and $\mathbf{p}_i$ the output of the neural network:

$$\mathbf{p}_i = \text{NeuralNetwork}(\mathbf{e}_i)$$

The neural network outputs probabilities for each document, which are used for classification tasks.

4.9 Step 9: Decision Tree Classification

Finally, we apply a **Decision Tree** classifier to predict the target class for each document.

5 Task 2: Conference Classification and Justification

In this task, we aim to classify research papers into appropriate conferences based on their content, keywords, and contextual embeddings. This approach is the core of our report, receiving the highest weightage due to its complexity and importance. The methodology is detailed as follows:

5.1 Step 1: Making embeddings of research paper

Using the pretrained **SciBERT** model, we extract embeddings for all research papers. Let the textual content of a paper d_i be represented as:

$$\mathbf{e}_i = \text{SciBERT}(d_i)$$

where $\mathbf{e}_i$ is the embedding vector for the document d_i. This step provides the foundational representation of the papers in a high-dimensional embedding space.

5.2 Step 2(a): Generating Keywords for Each Conference

For each conference, we create a large list of approximately 200 keywords found from OpenAI Free LLM and arxiv.org. The keywords for conference c_j are represented as:

$$\mathcal{K}_j = \{k_{j1}, k_{j2}, \ldots, k_{jm}\}, \quad m \approx 200$$

where $\mathcal{K}_j$ is the set of keywords for conference c_j.

5.3 Step 2(b): Creating one more list of keywords

Which is more mutually exclusive in nature

5.4 Step 3: Computing Cosine Similarity Between Papers and Conference Keywords

To evaluate the relevance of a research paper d_i to conference c_j, we calculate the cosine similarity between the SciBERT embeddings of d_i and the embeddings of the keywords in $\mathcal{K}_j$:

$$\cos(\theta_{ij}) = \frac{\mathbf{e}_i \cdot \mathbf{v}_j}{\|\mathbf{e}_i\|\|\mathbf{v}_j\|}$$

where $\mathbf{v}_j$ represents the average embedding of all keywords in $\mathcal{K}_j$. This similarity is stored for further analysis.

5.5 Step 4: Computing Euclidean Distance as an Alternative Measure

In addition to cosine similarity, we compute the Euclidean distance d_{ij} between the SciBERT embedding of d_i and the average embedding of $\mathcal{K}_j$:

$$d_{ij} = \|\mathbf{e}_i - \mathbf{v}_j\|$$

This measure provides an alternative perspective on the proximity of a paper to a conference's thematic focus. This give one prediction

5.6 Step 5: Updating Keywords to Ensure Mutual Exclusivity

We refine the list of keywords for each conference by ensuring mutual exclusivity. For two conferences c_j and c_k, we enforce:

$$\mathcal{K}_j \cap \mathcal{K}_k = \emptyset$$

This ensures that each keyword distinctly represents only one conference, eliminating ambiguity in classification. This gives another prediction

5.7 Step 6: Averaging SciBERT Embeddings for Final Similarity Computation

We compute the average embedding $\bar{\mathbf{e}}_i$ for all research papers using the SciBERT model:

$$\bar{\mathbf{e}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}_i$$

where n is the total number of papers. Using $\bar{\mathbf{e}}$, we calculate the similarity with the updated keyword embeddings for final classification. This gives one more prediction

5.8 Step 7: Extracting Results Using the T5 Model

We are using t5 for classification on abstract and a small content text which helps in classifying. And it gives one more prediction

5.9 Step 8: Prioritizing Predictions

The final predictions are prioritized by assigning maximum weightage to the SciBERT-based model.The other models are listened to if atleast 3 are giving same output this is done because SciBert is the most robust model for this task

5.10 Step 9: Generating Justifications

To justify the classification, we combine the T5-generated summaries and the top keywords for the predicted conference. Using a mapping of conference abbreviations and descriptions, we create a justification for each classification. The justification format is as follows:

> *"It is classified into* ***[CONFERENCE]*** *(****[FULL NAME]****) because the work on* ***[SUMMARY]****, combined with the key themes of* ***[KEYWORDS]****, aligns well with the themes of* ***[CONFERENCE]****."*

Code Implementation for Justification

Below is a snippet of Python code used to generate the justifications programmatically:

```
from transformers import pipeline

# Step 1: Initialize the summarizer pipeline
summarizer = pipeline("summarization", model="t5-large", tokenizer="t5-large")

# Step 2: Generate a paper summary using T5 model
def generate_summary(text):
    summary = summarizer(text, max_length=100, min_length=50, do_sample=False)
    return summary[0]['summary_text']
```

```
# Step 3: Generate justification
def generate_justification(paper_text, topic_keywords, conference_abbr):
    selected_keywords = topic_keywords[:10]
    summary = generate_summary(paper_text)
    return f"It is classified into {conference_abbr.upper()} because the work on {
           f"combined with the key themes of {', '.join(selected_keywords)}, " \
           f"aligns well with the themes of {conference_abbr.upper()}."
```

This approach integrates embeddings, clustering, and summarization models into a cohesive framework, ensuring both accuracy and interpretability in conference classification.

6 Techstack

The following technologies and libraries were utilized throughout the project to ensure robust and efficient implementation:

- **Primary Frameworks and Libraries:**
  - **Transformers:** Hugging Face's `transformers` library for state-of-the-art NLP models.
  - **SciBERT:** A domain-specific BERT model tailored for scientific text processing.
  - **T5:** A transformer model for text-to-text tasks, used for summarization and classification.
  - **PyTorch:** For deep learning model development and fine-tuning.
  - **TensorFlow:** Supplemental deep learning framework for experimentation.
- **Data Processing and NLP:**
  - **pandas:** For structured data manipulation and analysis.
  - **NumPy:** For efficient numerical operations.
  - **re (Regex):** For pattern matching and text preprocessing.

  - **spaCy:** For tokenization, named entity recognition, and linguistic analysis.
  - **NLTK Toolkit:** For additional natural language processing utilities.

- **Machine Learning and Topic Modeling:**
  - **scikit-learn:** For clustering (e.g., K-means), decision trees, and evaluation metrics.
  - **Gensim:** For advanced topic modeling and Latent Dirichlet Allocation (LDA).

- **Repositories and Modules:**
  - **Pathway Modules:** Integrated pathway modules for pipeline management.
  - **GitHub Repository:** Hosted codebase for collaborative development and version control.

This diverse tech stack enabled the implementation of advanced NLP techniques, robust data processing, and efficient model training and evaluation.

7 Additional things for task 2

7.1 Parsing Quality of Research Papers

Accurate parsing of research papers was a cornerstone of the system. Leveraging the SciBERT model, the pipeline extracted key information, including abstracts, titles, and keywords, while ensuring minimal loss of context. Preprocessing steps such as tokenization, stopword removal, and stemming were employed to improve the quality of extracted text. Furthermore, advanced parsing techniques ensured compatibility with various document formats like PDFs and text files.

7.2 Efficiency of Retrieval

Efficient retrieval mechanisms were implemented using a combination of vector embeddings and similarity measures such as cosine similarity and Euclidean distance. This enabled rapid identification of relevant research papers and associated keywords. Batch processing capabilities were introduced to handle large datasets, achieving a throughput of 100 papers per hour. Optimization of retrieval queries reduced unnecessary computations and enhanced overall efficiency.

7.3 Latency

Resource optimization was a critical consideration to minimize latency and resource consumption.

- **Latency:** Average processing time per paper was reduced after more optimisations,

7.4 Reasoning Behind Conference Selection

The rationale for conference selection was built on a robust understanding of conference domains and thematic alignment:

- **Keyword Matching:** SciBERT embeddings and topic modeling were used to align research paper content with conference themes. Similarity measures provided a quantitative basis for classification.
- **Domain Knowledge:** A mapping of conferences (e.g., KDD, CVPR, NeurIPS) to their respective focus areas was utilized to ensure alignment. For instance, papers with keywords related to "deep learning" and "optimization" were directed to NeurIPS.
- **Justification Generation:** T5-based summarization added contextual depth by combining extracted keywords with summaries to generate well-rounded justifications for classification.