KHARAGPUR DATA SCIENCE HACKATHON

KDSH 2025

TEAM: Poor GPU

Table of Contents

1 Research Paper Publishability Assessment

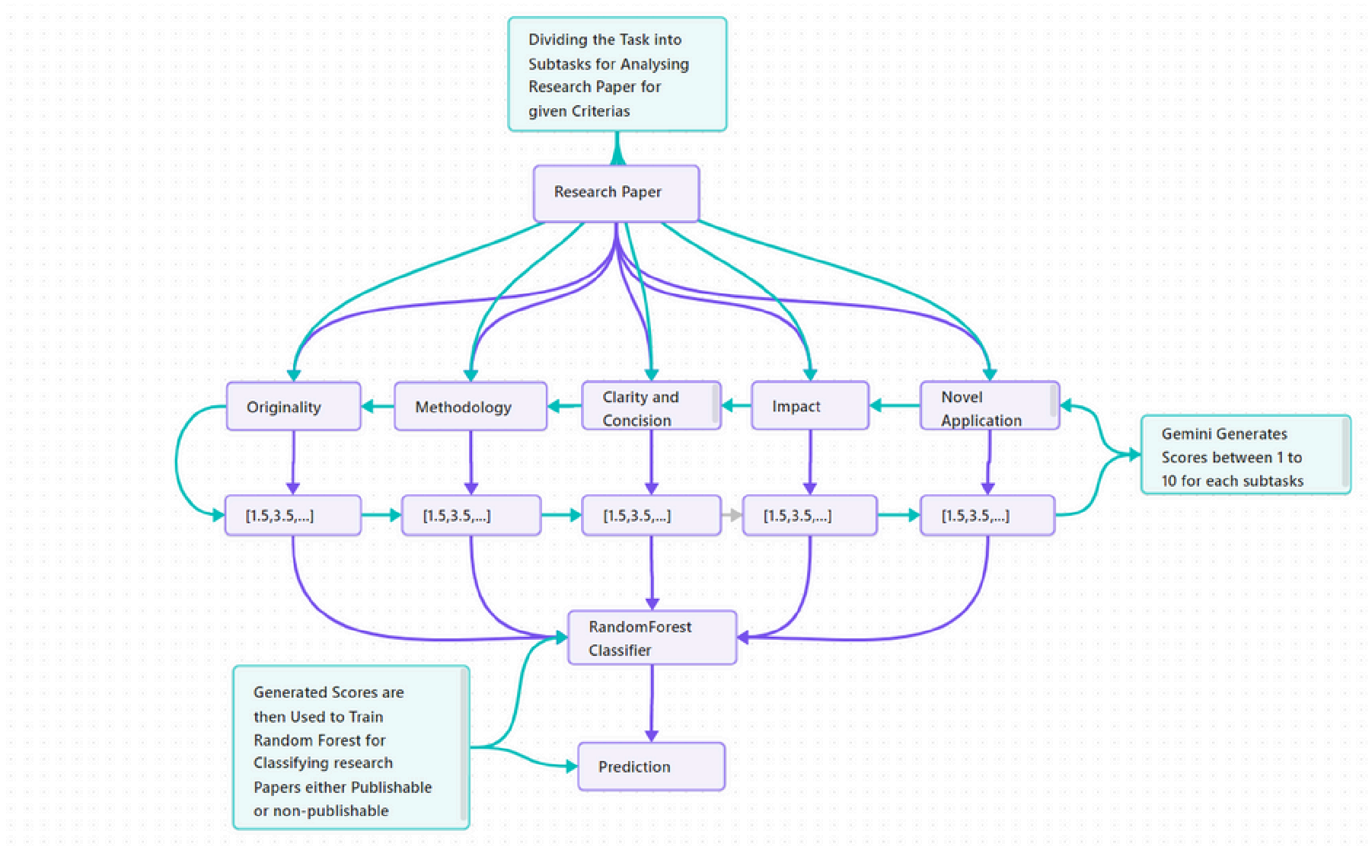
Objective

The goal of Task 1 is to classify research papers as either **"Publishable" or "Non-Publishable"** using a systematic **scoring methodology** and machine learning. By leveraging scores generated by the **LLM (gemini-1.5-flash)** for key evaluation criteria, we aim to develop a **Random Forest classification model** capable of accurately categorizing research papers.

<u>Step-by-Step Process:</u>

1) First we divide the Task in 5 subtasks. We ask the The LLM Model to Analyse Whole Research Paper and score the Research Paper in between 1 to 10 based on the Criterias defined below:

- **Originality,Methodology,Clarity and Conclusion,Impact and Novel Application**


Dividing the Task into Subtasks for Analysing Research Paper for given Criterias
Research Paper
Originality
Methodology
Clarity and Concision
Impact
Novel Application
Gemini Generates Scores between 1 to 10 for each subtasks
[1.5,3.5,...]
[1.5,3.5,...]
[1.5,3.5,...]
[1.5,3.5,...]
[1.5,3.5,...]
RandomForest Classifier
Generated Scores are then Used to Train Random Forest for Classifying research Papers either Publishable or non-publishable
Prediction

2) Generating Scores Using LLM:

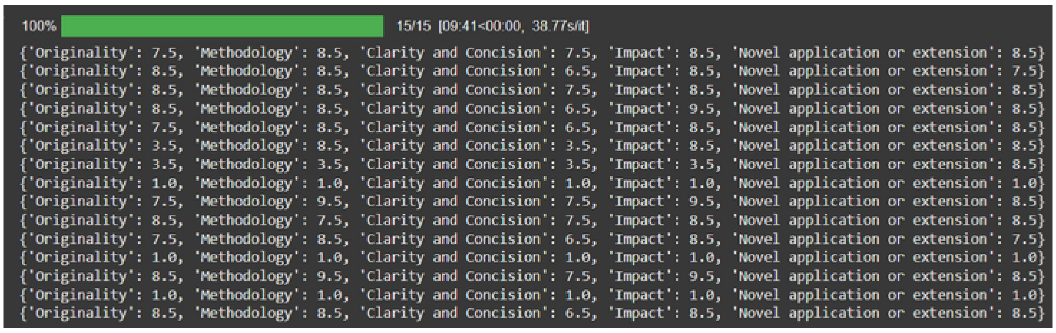
For Generating Scores We Used the **Gemini free Version LLM Model** .

Gemini is a powerful and efficient LLM capable of generating **accurate and context-aware evaluations of research papers.** Gemini has been extensively **trained on a diverse range of research papers and related academic content,** equipping it with the capability to analyze research papers effectively. This extensive training allows Gemini to evaluate critical criteria such as originality, methodology, impact, clarity, and novelty with precision. Its ability to understand complex academic language and provide nuanced scores ensures a systematic and reliable assessment of research papers for their publishability.

- **LLM Utilization:**
  - The **gemini-1.5-flash** LLM free version is employed to score each paper on the **five subtasks.**
  - A carefully crafted prompt is designed to instruct the LLM to evaluate the paper text and **generate a score (on a scale from 1 to 10)** for each subtask based on its respective criteria.

- **Process:**

The 15 labeled research papers are fed to the LLM one by one.For each paper, the LLM generates a set of scores for the five subtasks.



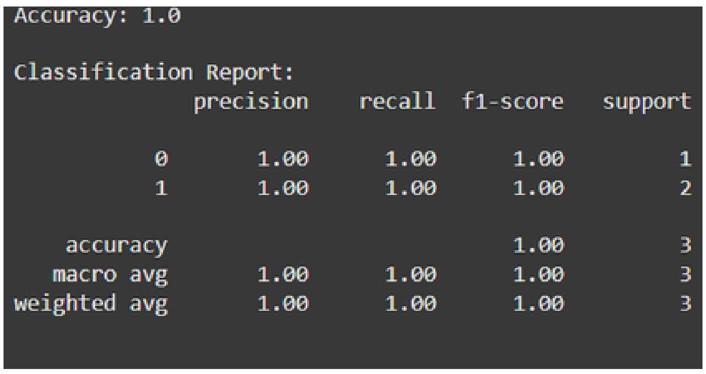
```
100% 15/15 [09:41<00:00, 38.77s/it]
{'Originality': 7.5, 'Methodology': 8.5, 'Clarity and Concision': 7.5, 'Impact': 8.5, 'Novel application or extension': 8.5}
{'Originality': 8.5, 'Methodology': 8.5, 'Clarity and Concision': 6.5, 'Impact': 8.5, 'Novel application or extension': 7.5}
{'Originality': 8.5, 'Methodology': 8.5, 'Clarity and Concision': 7.5, 'Impact': 8.5, 'Novel application or extension': 8.5}
{'Originality': 8.5, 'Methodology': 8.5, 'Clarity and Concision': 6.5, 'Impact': 9.5, 'Novel application or extension': 8.5}
{'Originality': 7.5, 'Methodology': 8.5, 'Clarity and Concision': 6.5, 'Impact': 8.5, 'Novel application or extension': 8.5}
{'Originality': 3.5, 'Methodology': 8.5, 'Clarity and Concision': 3.5, 'Impact': 8.5, 'Novel application or extension': 8.5}
{'Originality': 3.5, 'Methodology': 3.5, 'Clarity and Concision': 3.5, 'Impact': 3.5, 'Novel application or extension': 8.5}
{'Originality': 1.0, 'Methodology': 1.0, 'Clarity and Concision': 1.0, 'Impact': 1.0, 'Novel application or extension': 1.0}
{'Originality': 7.5, 'Methodology': 9.5, 'Clarity and Concision': 7.5, 'Impact': 9.5, 'Novel application or extension': 8.5}
{'Originality': 8.5, 'Methodology': 7.5, 'Clarity and Concision': 7.5, 'Impact': 8.5, 'Novel application or extension': 8.5}
{'Originality': 7.5, 'Methodology': 8.5, 'Clarity and Concision': 6.5, 'Impact': 8.5, 'Novel application or extension': 7.5}
{'Originality': 1.0, 'Methodology': 1.0, 'Clarity and Concision': 1.0, 'Impact': 1.0, 'Novel application or extension': 1.0}
{'Originality': 8.5, 'Methodology': 9.5, 'Clarity and Concision': 7.5, 'Impact': 9.5, 'Novel application or extension': 8.5}
{'Originality': 1.0, 'Methodology': 1.0, 'Clarity and Concision': 1.0, 'Impact': 1.0, 'Novel application or extension': 1.0}
{'Originality': 8.5, 'Methodology': 8.5, 'Clarity and Concision': 6.5, 'Impact': 8.5, 'Novel application or extension': 8.5}
```

The Results of the Scores Generated By Gemini for 15 labeled Papers for each subtasks.

3)Training the Classification Model:

- **Data Preparation:** The labeled dataset of **15 research papers** includes **feature vectors (scores)** and labels **(Publishable or Non-Publishable)**.
- **Dataset Split: 12** papers for **training**, **3** for **testing.**
- **Model Selection**: **Random Forest**, chosen for its **robustness** with **small datasets.**
- **Training:** Feature vectors from the training set are used to train the model.
- **Testing**: Model performance is validated on the testing set.

The **high F1 score (close to 1)** is due to the small dataset, enabling the **Random Forest** model to effectively learn patterns. Additionally, **Gemini's accurate scores align well** with the evaluation criteria, providing a **robust feature** set that enhances model performance.

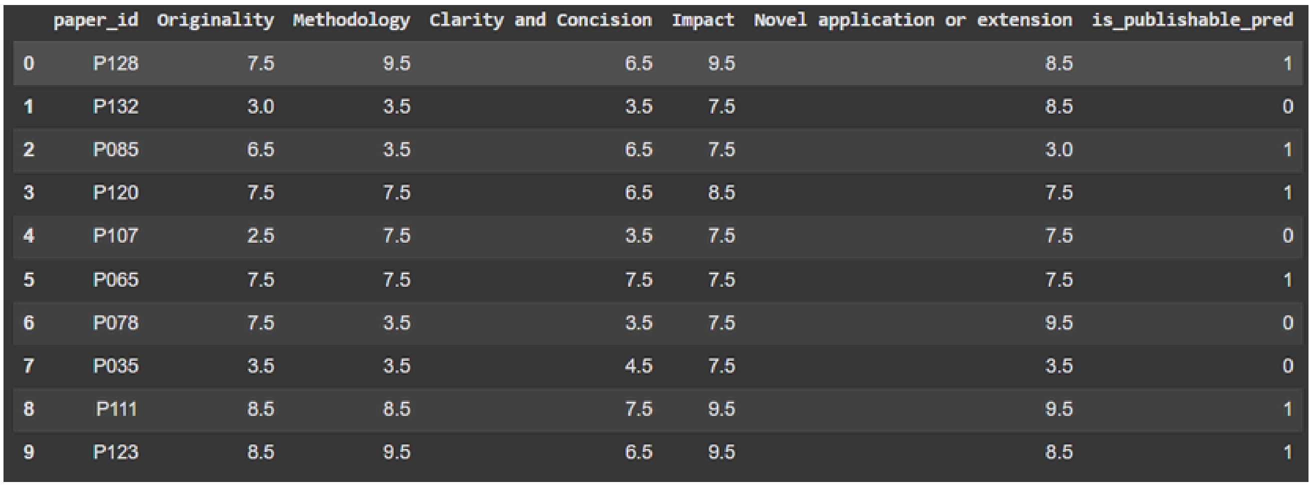


```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

After training the **Random Forest classifier**, the model is saved using the Python library **joblib**. This allows the trained model to be reused for predicting the output of new data **without the need for retraining**. This approach ensures efficiency and convenience, as the saved model can be loaded and deployed directly for inference.

4) Classification of Unlabeled Research Papers:

The **trained Random Forest model** was used to **classify 135 unlabeled research** papers. **Scores** for the five subtasks were generated using the LLM, and the resulting feature vectors were passed through the model. Each paper was classified as "**Publishable" (1) or "Non-Publishable" (0**), and the results were stored in a pandas DataFrame and exported as a **CSV file** for use in Task 2.



	paper_id	Originality	Methodology	Clarity and Concision	Impact	Novel application or extension	is_publishable_pred
0	P128	7.5	9.5	6.5	9.5	8.5	1
1	P132	3.0	3.5	3.5	7.5	8.5	0
2	P085	6.5	3.5	6.5	7.5	3.0	1
3	P120	7.5	7.5	6.5	8.5	7.5	1
4	P107	2.5	7.5	3.5	7.5	7.5	0
5	P065	7.5	7.5	7.5	7.5	7.5	1
6	P078	7.5	3.5	3.5	7.5	9.5	0
7	P035	3.5	3.5	4.5	7.5	3.5	0
8	P111	8.5	8.5	7.5	9.5	9.5	1
9	P123	8.5	9.5	6.5	9.5	8.5	1

2 Reason Of Using Scoring Technique and Not Using LLM

Reason for Using **Scoring Technique** Instead of **LLM Fine-Tuning in Task 1**

1. Limited Labeled Data:

Fine-tuning an LLM requires a large dataset for generalizable performance. With only **15 labeled research papers** available, fine-tuning **risks overfitting** and failing to capture meaningful patterns.

2. Token Limitations:

Many LLMs, like **SciBERT**, have token limits **(e.g. 512 tokens)**, making it difficult to process entire research papers spanning **5–10 pages**. Splitting papers into chunks can result in a **loss of context, disrupting relationships** between sections like methodology and results.

3. Advantages of Scoring with LLMs:

- **Holistic Analysis**: Using LLMs like **Gemini-1.5-Flash,** which support large token windows, enables processing entire papers at once, **preserving context across sections such as Abstract, Introduction, and Results**.
- **Domain Knowledge**: **Gemini**, trained on vast datasets, including research papers, evaluates content effectively based on **originality, methodology, clarity, impact, and novelty.**
- **Tailored Prompts:** Instead of fine-tuning, task-specific prompts allow the LLM to provide accurate scores, reducing the need for additional training.

4. Modularity of Subtasks

Breaking the task into **subtasks (e.g., originality, methodology)** allows detailed and transparent evaluation, providing insights into a **paper's strengths and weaknesses**. **Subtask-based scoring** is inherently interpretable, as each score reflects a specific aspect of the paper, ensuring transparency and justifiability in the **final classification.**

5. Use of Classical ML Models:

Random Forests are ideal for small datasets, effectively handling LLM-generated scores while being robust and interpretable.

3 Conference Prediction with Pathway

Objective :

The goal of Task 2 was to develop a **Real Time Pipeline** using "**Pathway"** that evaluates research papers that were predicted to be **Publishable** by Task 1 and recommends the **most suitable conferences** based on their content. The pipeline ensures recommendations **align** with conference **themes, scope, and objectives**, enabling efficient paper-to-conference matching.The system was also tasked with generating **detailed and contextual rationales** to justify the recommendations, ensuring alignment with the themes and objectives of the targeted conferences.

Approach:

- **Real-time Data Integration :**Configured Pathway's **Google Drive connector** in **streaming** mode to monitor and process incoming research papers in real-time.
- **Embedding Pipeline:**Utilized **MIXED BREAD AI** embeddings in conjunction with **Pathway's document store**, configured via **.yaml** for document chunking (using pathways PaarseUnstructured) to optimize processing efficiency.
- **Conference Matching:** Developed a chunk-based voting system that uses **REST API queries** (with top_k=1) to identify the most relevant conference for each paper based on **metadata mapping**.
- **Rationale Generation :**Employed LLMs to generate 130-150 word **Rationale** for each paper-conference match, providing clear justifications for recommendations.
- **Data Persistence:**Automated the storage of results in CSV format to track conference recommendations and paper matches.

Workflow:

Creating a REST API for document indexing and querying for similar chunks:

1. Setting up the data source using **Pathway Google Drive Connector**:
The documents are sourced from a Google Drive location, with the following configuration:

- The source is connected using the **pw.io.gdrive.read** plugin.
- The object_id refers to a specific google drive folder.
- Stored metadata is later used to find out which folder the paper is from (i.e. whether a labelled paper was published and in **which conference**).
- The source will be **checked each 30 secs** for updates.

2. **Embedding Model**

- The model used is **mixedbread-ai/mxbai-embed-large-v1**, a large-scale embedding model with 335M parameters.
- The **SentenceTransformerEmbedder** class from the **pw.xpacks.llm.embedders package** is employed to apply this model to the documents.

3. **Document Parser:**

- The **ParseUnstructured** class is employed to process PDFs into a list of strings.
- The parser uses a **default caching strategy** to improve performance by reusing previously parsed data when possible.

4. **Document Splitter:**

- The **TokenCountSplitter** is used to divide the text into chunks.
- The **max_tokens** parameter is set to 400**.** This ensures that all of the chunks fit in the 512 token limit of our selected embedder.
- It also attempts to break chunks at sensible points such as punctuation marks.

5. **Document Retrieval** : looking for the most similar chunk of text

- The **LshKnnFactory** is used to create a retrieval index. It uses **Locality-Sensitive Hashing (LSH)** for approximate nearest neighbor search faster than **BruteForceKnn.**
- The retrieval is based on the **cosine distance** between the query and document embeddings.

6. Document Store: storing the embeddings efficiently

- The Pathway **DocumentStore** from **pw.xpacks.llm.document_store** is used for storing the embeddings.

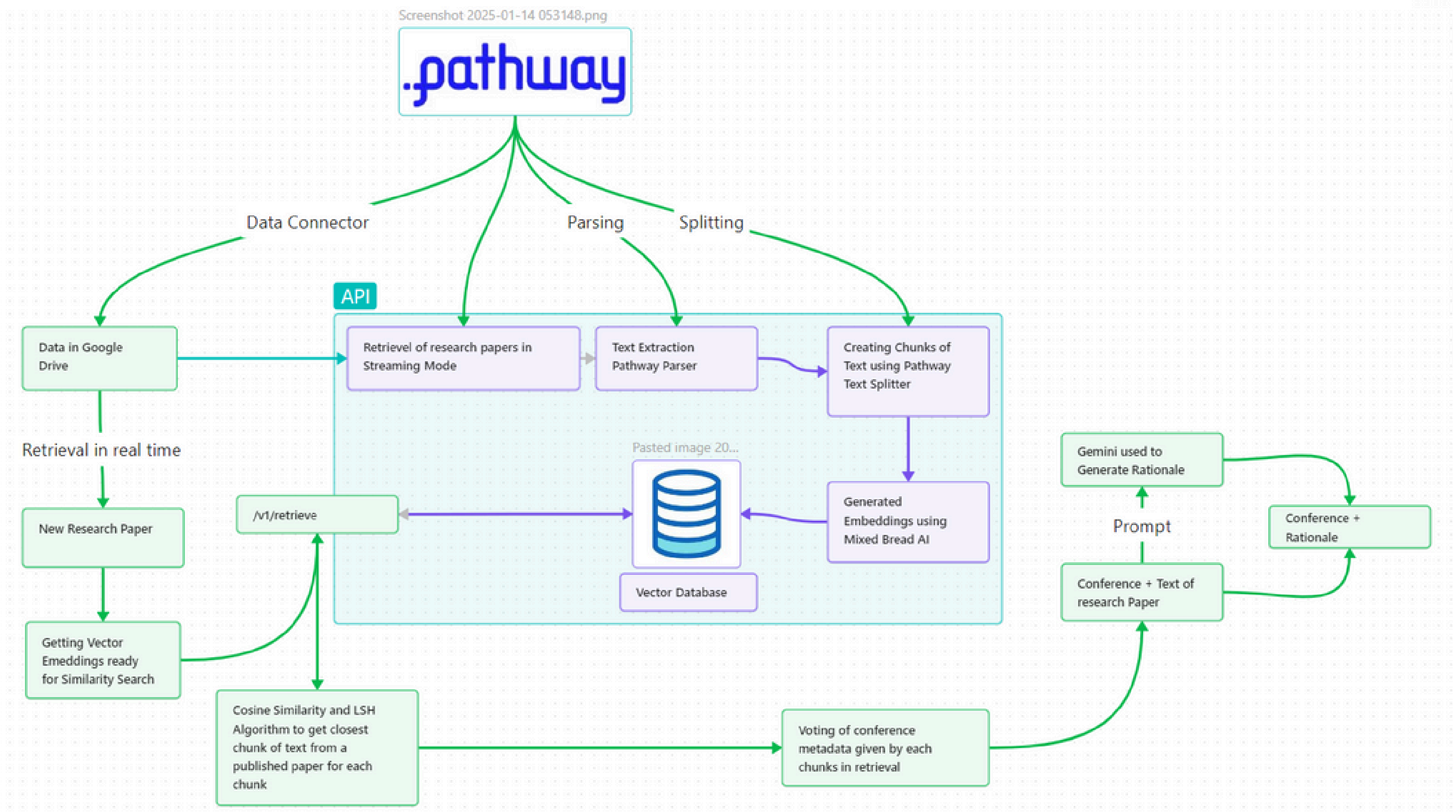
The document store is designed to facilitate the efficient storage, indexing, and retrieval of document chunks.

7. API Endpoint Configuration:

The setup involves a REST API with the **/v1/query** endpoint for querying and returning most similar chunks of texts given some text.

8. Cache:

- Caching is enabled to improve performance by **storing the results of previous queries,** reducing the need for repeated computations.



Screenshot 2025-01-14 053148.png
pathway
Data Connector
Parsing
Splitting
API
Data in Google Drive
Retrievel of research papers in Streaming Mode
Text Extraction Pathway Parser
Creating Chunks of Text using Pathway Text Splitter
Retrieval in real time
Pasted image 20...
Gemini used to Generate Rationale
New Research Paper
/v1/retrieve
Generated Embeddings using Mixed Bread AI
Prompt
Conference + Rationale
Vector Database
Conference + Text of research Paper
Getting Vector Emeddings ready for Similarity Search
Cosine Similarity and LSH Algorithm to get closest chunk of text from a published paper for each chunk
Voting of conference metadata given by each chunks in retrieval

Query and Retrieval using API call:

1. Input Data Handling:

- **Research Paper Ingestion:** The system starts by ingesting research papers from Google Drive in PDF format. This is done using Pathway's Google Drive connector.When ever a new paper is added a callback function **on_change** gets triggered due to using **pw.io.subscribe** function which ensures any changes made in pathway table sends a **callback**.
- A **dictionary mapping** Conference Folder id's to Conference Name was made, as we can **extract parents folder id** from **metadata** of the retreived files ,mapping to conference name would be easier.

2. PDF Text Extraction :

- **PyMuPDF for Text Extraction:** To extract the textual content from the PDFs, **Fitz** of **PyMuPDF** was used.Fitz enables **fast** and efficient extraction of raw text from PDF documents while maintaining the structure and formatting where possible.
- **Data Cleaning:**The data extracted has many "**\n**" characters between words which need to be replaced by **spaces** for **meaningful cleaned** text.

3. Text Chunking :

- **Chunking Process:** Once the text is successfully extracted from the PDFs,it is split into **smaller chunks** using **Pathway's TokenCountSplitter function** ensuring compatibility with the models used in the next steps (e.g. Querying).
- **Ensuring Context Retreival :**The Pathway's TokenTextSplitter ensures that the words are **not being cut** in between while chunking ,the splitter **carefully splits** so that context remains intact.

4. Conference Prediction:

- **Similarity-based Retrieval :**Each chunk of text is then sent to a REST API for **similarity-based retrieval.**The API **queries Pathway's document store** to find relevant research paper's chunks or conference details that match the content of the chunk.
- **Voting Mechanism for Aggregation:** Since each paper is chunked into multiple pieces, a **voting mechanism** is implemented to aggregate the conference predictions **from each individual chunk.** This ensures that the final conference recommendation for the paper is derived from the **consensus across all chunks.**
- **Top-K Strategy:**The **top_k** parameter is set to *"**1**"*, meaning that the API will return the **most similar result for each chunk**. This helps identify the **best possible** conference for the paper based on its content.

5. Rationale Generation :

- **Detailed Rationales :**Once the conference predictions are made,the system **generates detailed rationales** explaining why a particular conference was chosen. This is done by using a **structured prompt sent to the Gemini-1.5-Flash model**,which provides contextual explanations based on the paper's content and its **relevance to the predicted conference**.

4 Integration of Task 1 in Task 2

In **Task 2**, the **classification system** from **Task 1** was **seamlessly integrated** to enhance the **end-to-end pipeline.** The process is as follows:

1. Real-Time PDF Retrieval:

Using **Pathway's Google Drive connector in streaming mode**, **research papers** in PDF format are automatically **retrieved** whenever added to the designated folder.

2. Text Extraction:

The textual content of the retrieved PDFs is extracted using **PyMuPDF**, ensuring the data is clean and formatted for processing.

3. Scoring and Classification:

- **Function Import from Task 1**: The functions for **generating scores** using the **Gemini model** were imported from Task 1. These ensure consistency and reusability in scoring across tasks.
- **Trained Random Forest Model:** The **Random Forest model** from **Task 1**, saved using **joblib**, is loaded to classify the papers.
- **Score Generation**: **Gemini** is used to **generate scores** for the extracted text based on evaluation criteria (**Originality, Methodology, Clarity, Impact, Novelty**).
- **Classification:** The generated scores are passed to the Random Forest model to classify the paper as either **Publishable or Non-Publishable**.

4. RAG-Based Conference Recommendation System:

- For papers **classified as Publishable**, the **RAG-based system** is executed to get the conference it belongs .
- A **130–150** word rationale is generated using **Gemini**, explaining the **conference recommendation** based on the paper's content.

5. Output Generation:

- The final results, including the **publishability classification, conference recommendation,** and **rationale,** are compiled into a **CSV file** for easy tracking and reporting.

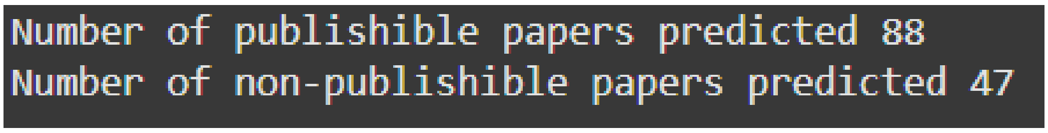
5 Challenges and Solutions

CHALLENGES	SOLUTIONS
Fine-tuning an LLM could lead to **overfitting**, which might restrict its ability to capture the full **context of the problem.** This would limit the model's adaptability to a variety of scientific papers and criteria.	Instead of fine-tuning,we opted for a **scoring** approach where the LLM is used to evaluate a paper based on predefined **scientific criteria**. This allows us to leverage the model's **capabilities without overfitting** and ensures the evaluation process remains contextually aware.
Due to the **restrictions** of the **Free Gemini API,** there was a risk of incomplete or **failed responses** when generating scores, leading to inconsistencies in the training and evaluation process.	To address this, **a retry mechanism** with **delays** was implemented to ensure the scores were properly generated. This approach allows for **automatic retries** and manages temporary API failures, ensuring reliable output without **manual intervention**.
When we were trying to build the **entire pipeline** of Task 2 as the **same program**, the resultant code seemed to end up being very **complicated** and **hard to manage**. It was especially hard to manage **real-time updates on data sources** (e.g. GDrive).	Being heavily influenced by the **LLM App** repository by Pathway, we decided to build the document index as a REST API, in which we only needed to carefully configure the **app.yaml**. We ended up implementing the rest of the solution as a different program the utilizes the **retrieve** endpoint of this API.
Increased latency due to **BruteForceKnn** and making sure there's no absence of a **caching mechanism** in querying of the Index	We used **pw.stdlib.indexing.LshKnnFactory** instead of BruteForceKnn and used **cosine** similarity as the distance metric, instead of **euclidean**. We set **with_cache: true** in our configuration YAML file.
Entire research papers could **not be embedded** or queried due to **token limit constraints**.	Papers were divided into **smaller, manageable chunks** using a text chunking mechanism, ensuring each chunk stayed within token limits while **maintaining context.**
Dividing papers into multiple chunks required **querying each chunk separately**. Consequently, setting top_k=1 for all queries meant identifying the best match for each chunk, but there was **no holistic decision-making** for the paper.	Introduced a **voting system** where each chunk contributed a vote to a conference. The conference receiving the **highest votes** across chunks was chosen as the **most suitable**, balancing individual chunk evaluations with **overall relevance.**

6 Results

Task 1 Results :

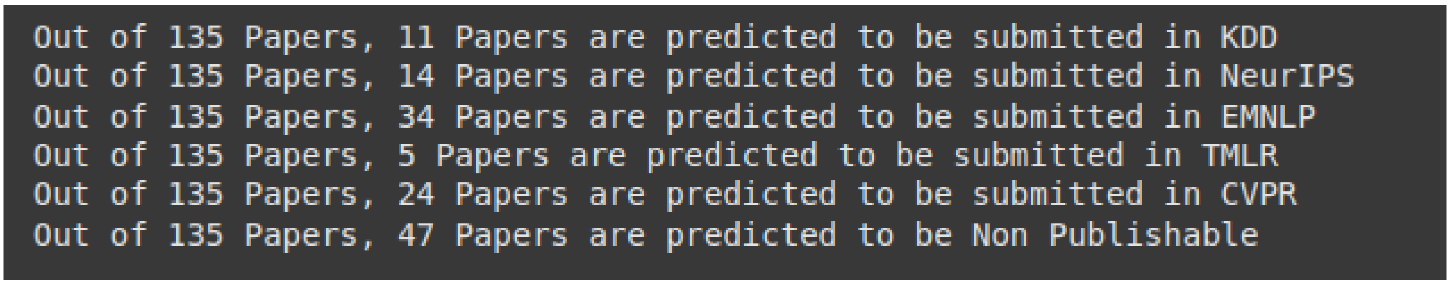
Out of the **135 Unlabebled Research Papers** Our Model Classified **88 Research Papers** as **Publishible** and **47 Papers as Non-Publishible**



```
Number of publishible papers predicted 88
Number of non-publishible papers predicted 47
```

Task 2 Results :

- The system successfully classified research papers as publishable or non-publishable.
- For publishable papers, it recommended conferences such as **CVPR, NeurIPS, EMNLP, TMLR,** and **KDD.**Rationales provided meaningful insights, focusing on methodology, novelty, relevance, and impact.
- Results were stored in a **CSV** file, making them easy to interpret and share.
- Out of 135 papers ,
  - **11** Papers are predicted to be presented in **KDD**
  - **14** papers are predicted to be presented in **NeurIPS**
  - **34** Papers are predicted to be presented in **EMNLP**
  - **5** Papers are predicted to be presented in **TMLR**
  - **24** Papers are predicted to be presented in **CVPR**



```
Out of 135 Papers, 11 Papers are predicted to be submitted in KDD
Out of 135 Papers, 14 Papers are predicted to be submitted in NeurIPS
Out of 135 Papers, 34 Papers are predicted to be submitted in EMNLP
Out of 135 Papers, 5 Papers are predicted to be submitted in TMLR
Out of 135 Papers, 24 Papers are predicted to be submitted in CVPR
Out of 135 Papers, 47 Papers are predicted to be Non Publishable
```