Contents

Abstract

This research presents a comprehensive machine learning framework to analyze the rich and complex ICLR 2025 peer review dataset, integrating both detailed textual reviewer feedback and quantitative rating scores. Addressing challenges such as severe class imbalance, diverse text lengths, and missing data, we implement rigorous data preprocessing and feature engineering techniques to enhance model robustness. We systematically evaluate multiple predictive approaches, including regression, classical classifiers, recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and state-of-the-art transformer-based models (DistilBERT). Our extensive hyperparameter optimization and stratified data splitting reveal that transformer architectures, particularly DistilBERT, outperform traditional models in accurately classifying reviewer ratings while effectively managing class imbalance. Additionally, unsupervised topic clustering using TF-IDF embeddings combined with K-means uncovers key emerging research themes, providing valuable insights into the scientific landscape of submissions. This study underscores the critical importance of fusing numeric and textual data modalities for predictive accuracy and demonstrates the computational efficiency and generalizability benefits of transformer models in large-scale peer review analysis. Despite limitations posed by dataset subjectivity and computational demands, our approach significantly advances automated, scalable peer review evaluation and contributes to enhancing transparency and fairness in scientific assessment.

1 Introduction

The International Conference on Learning Representations (ICLR) is one of the premier conferences in machine learning and artificial intelligence. As the volume of submissions continues to grow each year, there is a need for scalable and efficient methods to automate the analysis of submissions and its peer review process.

This project leverages the ICLR 2025 dataset to analyze reviewer comments and ratings, uncovering insights into scientific discourse and peer evaluation. ICLR's open-review system publicly shares submissions, reviews, rebuttals, and decisions, creating a rich dataset of textual and numerical metadata for large-scale analysis. However, traditional manual and rule-based approaches are no longer sufficient to manage the complexity and size of modern scientific review data.

In this report, we aim to build models that combine both text and numerical inputs to predict review scores more accurately. To do this, we experiment with a range of word embedding techniques from basic methods like TF-IDF to more advanced models like DistilBERT tokenizers. We also compare different model architectures, including RNNs, LSTMs, and transformer-based models, DistilBERT, to understand which approaches are the best in capturing meaningful patterns in peer reviews. This project seeks to uncover insights into peer review behaviour, improve score prediction, and contribute to more transparent and scalable review systems in academic publishing.

2 Literature Review

- **Classical Text Representation Approaches:** Classical text representation methods like Bag-of-Words (BoW) and TF-IDF convert text into numerical features, emphasizing term importance in large datasets like ICLR2025. Although efficient and interpretable, they ignore word context and syntax, causing semantically different texts to appear similar (Yan et al., 2020). Despite these drawbacks, their simplicity and speed make them useful for initial analysis before applying more advanced context-aware models.
- **Topic Modeling Techniques:** Latent Dirichlet Allocation (LDA) are key unsupervised methods for discovering themes in large text datasets like the ICLR submissions. LDA produces more descriptive topics but involve computationally intensive processes that challenge scalability (George & Vasudevan, 2020). These trade-offs emphasize the importance of choosing and optimizing the right method for large scientific review datasets.
- **Static Word Embeddings:** Word2Vec, FastText, and GloVe advanced text representation by creating dense embeddings that capture semantic relationships better than traditional sparse methods, improving tasks like clustering and classification. Static embeddings like Word2Vec assign a single vector per word, limiting their ability to handle context and polysemy (Church, 2016). Additionally, training these models demands significant computational resources, challenging scalability for large datasets like ICLR2025. Therefore, this assignment emphasizes models that balance performance with efficiency and interpretability, reflecting the need to trade off semantic depth for practical feasibility in large-scale text analysis.
- **Unsupervised Clustering Algorithms:** Clustering methods like K-Means, DBSCAN, and Gaussian Mixture Models (GMMs) effectively identify thematic groups in textual submissions (Task B). K-Means is scalable but requires a predefined number of clusters; DBSCAN handles noise and arbitrary shapes but struggles with high-dimensional text; GMMs capture cluster overlap probabilistically but are computationally costly. Dimensionality reduction techniques such as t-SNE and UMAP aid interpretation but may not scale well to large datasets like ICLR2025. Balancing accuracy, interpretability, and efficiency is essential for large-scale text clustering (Jain, 2010; Ester et al., 1996; McLachlan & Peel, 2000; van der Maaten & Hinton, 2008).
- **Dimensionality Reduction Methods:** Dimensionality reduction methods like PCA and t-SNE facilitate visualization and interpretation of high-dimensional embeddings. PCA is efficient but limited to linear structures, while t-SNE and UMAP capture local and global patterns more effectively at higher computational cost. These techniques are crucial for interpreting clustering and revealing patterns in text embeddings, aiding Task B analysis (Jolliffe, 2002; van der Maaten & Hinton, 2008; McInnes et al., 2018).
- **Neural Network Architectures for Textual Analysis:** Neural architectures such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Recurrent Neural Networks (RNNs) effectively model sequential and contextual patterns in text. These models are well-suited for predicting reviewer recommendations, as they handle variable-length inputs like summaries and assessments of strengths and weaknesses. By capturing long-range dependencies and hierarchical features, they enable nuanced analysis of peer review content. However, their use demands careful management of training complexity, computational resources, hyperparameter tuning, and rigorous validation to prevent overfitting and ensure generalizability (Kim, 2014; Lipton et al., 2015; Zhou et al., 2015).
- **Contextualized Embedding Approaches (Transformer-based):** Advanced transformer-based embed-

dings like BERT dynamically capture contextual semantics, enabling superior handling of polysemy and word sense disambiguation compared to static embeddings. These models excel in tasks such as predicting reviewer ratings from text due to their nuanced linguistic representation. However, transformers require significant computational resources, including GPUs and extended training time. DistilBERT mitigates this by using knowledge distillation to offer a lighter, faster alternative that retains much of BERT's accuracy, making it suitable for resource-constrained settings. (Patil et al., 2023).

3 Data Cleaning & Pre-Processing

3.0.1 Checking Duplicate Rows

We checked for duplicate rows in the `ICLR2025_train` dataset, including and excluding the `ReviewID_Train` column from consideration. No duplicate entries were found, indicating that no review from a single marker was accidentally recorded more than once. While some papers were reviewed by multiple markers, each review includes distinct scores for `Soundness`, `Presentation`, `Contribution`, and `Confidence`, as well as unique comments in the `Summary`, `Strengths`, and `Weaknesses` fields.

We also checked the submissions file `ICLR2025_sub` for duplicate rows, including and excluding the `PaperID` column from consideration. No duplicate entries were found, indicating that no paper was accidentally submitted or recorded more than once.

3.0.2 Handling Missing Values and Outliers

Upon examining `ICLR2025_train`, we identified some missing text values. There is 1 missing value in the `Summary` column, 26 missing values in the `Strengths` column, and 17 missing values in the `Weaknesses` column. Given that the number of missing values is a very small proportion of the 28,048 total entries, we will impute them with the placeholder *'Not Provided'* to maintain consistency in the dataset.

Next, we examined `ICLR2025_sub`. We identified 5022 missing values in the `tldr` column. This is a substantial proportion of the 11,672 total entries. The `tldr` column is a one-sentence summary provided by the author(s) to summarise their papers. It closely resembles the `keywords` column, which includes a list of author-provided keywords. To reduce noise in our model inputs, we decided to remove the `tldr` column from our analysis.

3.0.3 Text Pre-Processing for Natural Language Processing

To prepare raw textual data for model training, we adopted commonly used NLP pre-processing practices as outlined by (Silva, 2023). As machine learning models relies heavily on the quality of data fed into it, data pre-processing is crucial to enhance the performance and efficiency of the model. We standardised the text across the five text columns title, abstract, summary, strengths, and weaknesses.

- **Text Cleaning**
  - Lowercasing all text to ensure consistency. This ensures words like "Reinforcement" and "reinforcement" are treated identically.
  - Removing punctuation, special characters, and digits, as they generally do not contribute meaningfully to text analysis.
  - Trimming excessive whitespace to improve tokenization accuracy, as most tokenizers split based on spaces.
- **Tokenization**: We employed a basic English tokenizer to split text into individual word tokens.
- **Stop Word Removal**: Common words such as "the", "an", and "is" were removed using the Natural Language Toolkit (NLTK), as they often add little semantic value.
- **Lemmatization**: We applied Part-of-Speech (POS) based lemmatization to convert words into their base forms (e.g., "models" to "model"), thereby reducing textual noise and enhancing interpretability.

4 Task A - Exploratory Data Analysis

4.1 Rating Distribution

The rating scores in the ICLR dataset represent reviewer recommendations and are defined as follows: 1 (Strong Reject), 3 (Reject), 5 (Marginally Below Acceptance Threshold), 6 (Marginally Above Acceptance Threshold), 8 (Good Paper), and 10 (Strong Accept). We will treat these scores as categorical variables as they reflect distinct decision thresholds.

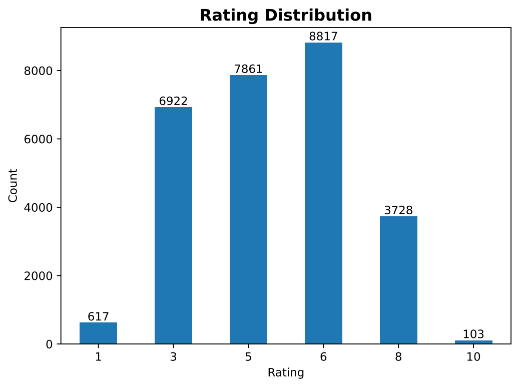


Figure 1: Distribution of Ratings

From Figure 1, the distribution of ratings is notably imbalanced. Most papers are scored within the mid-range ratings 5 and 6, 7,861 reviews and 8,817 reviews respectively. This suggests that the scores given for the majority of papers are close to the marginal acceptance threshold. The extreme scores of 1 and 10 are visibly underrepresented.

This class imbalance can pose as a challenge across different model architectures. This imbalance can introduce a performance bias to our models, where learning algorithms perform well on frequent classes but poorly on minority classes (Chen et al., 2024). If we fail to address class imbalance, we risk our model struggling in predicting extreme ratings like 1 (Strong Reject) and 10 (Strong Accept). Our model would over predict majority classes like 5 and 6.

There are two key strategies we will adopt to account for imbalanced classes in our machine learning/ deep learning algorithms:

- Using evaluation metrics beyond accuracy: We will incorporate the **Macro F-1 Score and Weighted F1-Score** as performance measures to give more insight into the accuracy of the model, particularly on underrepresented classes.
- Introducing a **Class Weight** parameter during model training: This encourages the model to assign higher importance to minority classes during training, reducing bias towards majority classes.

4.2 Text Length Analysis

A detailed analysis of text lengths was done on the submissions dataset, focusing on titles and abstracts. Titles ranged from 1 to 23 words, with an average of about 9 words. Abstracts varied more widely, from 1 to 461 words, averaging around 188 words. No limits were set on text length during preprocessing to avoid losing important information, allowing later models to handle the varying lengths.

For the rating dataset, reviewer comments in the Summary, Strengths, and Weaknesses fields showed wide variation. Minimum lengths were as short as one word, while maximum lengths reached 1,487 words for summaries, 607 for strengths, and 3,573 for weaknesses. Average lengths were lower—about 88 words for summaries, 64 for strengths, and 204 for weaknesses—with medians even lower, showing that a few very long comments skew the averages. This range highlights the need to carefully manage varying text lengths in analysis and modeling.

4.3 Keyword Frequency Analysis

Analysis of the top 20 author keywords in ICLR submissions shows a strong focus on large language models, with "large language models" appearing over 800 times, followed by "reinforcement learning" and related terms. Generative modeling topics like diffusion models, deep learning, and generative models are also prominent (see Appendix Figure 11). Core machine learning themes such as benchmarks, representation learning, and transformers rank highly, along with emerging areas like interpretability, federated learning, and alignment. Graph neural networks remain important, reflecting ongoing interest in structured data. Overall, the submissions emphasize large-scale language and generative modeling, alongside evaluation, interpretability, and collaboration-focused topics.

5 Topic Clustering

5.1 Description and definitions

Research papers submitted for the ICLR 2025 were grouped according to their title, abstract, and keywords (all text variables) to identify thematic similarities and emerging research trends. Clustering techniques were applied to the submission data set. Clustering refers to an unsupervised learning method in which labels are unknown and sometimes do not exist. The goal is to group papers that are similar to each other and dissimilar to papers within other clusters. For our clustering algorithms to process data efficiently, high-dimensional, unstructured data such as text data need to be translated into numerical vector representations. Hence, the need to select a text feature engineering technique.

This report's section experiments with different combinations of embedding methods and clustering algorithms to group papers into similar topics based on combined text of abstracts, titles and keywords. Three clustering algorithms were applied to two different embedded methods. Text columns were combined to add more information on the topics than abstract, title or keywords columns alone. more precise and specific. Indeed, by themselves text columns can be sparse or biased if some information is omitted in the title or keywords for instance. Combining them creates a more complete document and leverages complementary strengths giving more precise and specific clusters than using one column itself.

5.2 Experimental Setup

The same Data Cleaning and Data Pre-Processing steps as Section 3 were used. The whole workflow is coded in Python 3 and uses PyTorch. It was run on a Tesla T4 GPU (GPU T4).

Hyperparameter selection: Both the K-means and the GMMs algorithms have to set the number of clusters **k** before training. Analysis is required to find the optimal **k**.

K-means: K-means objective is to find centroids that minimize the total inertia (within-cluster sum of squared distances).

The elbow method and **the silhouette score** analysis were implemented before training to determine the optimal **k**. The "elbow" point in the curve is where the optimal **k** is. It is the point beyond which adding more clusters yields only marginal reduction in the within-cluster sum of squares WCSS. **The silhouette score** is a measure of how well each sample lies within its cluster compared to other clusters. Silhouette analysis tells you how separate and cohesive your K-means clusters are, and which **k** maximizes that balance.

GMMs: For classic GMMs, we can use the Bayesian Information Criterion (BIC) score and choose the **k** with the lowest score. We applied this analysis to our dataset but the running time was lengthy and we decided to use the same number of clusters found with TF-IDF and K-means combined $\mathbf{k} = \mathbf{8}$.

5.3 Results and Comparison of Combinaisons

Table 1: Comparison of clustering algorithms across feature representations

Algorithm	Representation	Observation
K-means	BoW	With k=2 based on the Elbow method and Silouhette score (See appendix A.3, Figure 4) the t-SNE visualization is poor (See appendix A.3, Figure 5), documents have been randomly assigned to perfect clusters.
	TF–IDF	With k=8, We can clearly see the different topic clusters and which topics are more popular (Figure 2). The k was selected in a similar manner (See appendix A.3, Figure 6).
DBSCAN	BoW	With k=15, No clear separation Too many clusters have been determined by the algorithm. It is hard to separate them into main topics (See appendix A.3, Figure 7).
	TF–IDF	With k=26, No clear separation. Too many clusters have been determined by the algorithm. It is hard to separate them into main topics (See appendix A.3, Figure 7).
GMMs	BoW	With $k = 8$, more computationally expensive than K-means yet produces comparable cluster quality(See appendix A.3, Figure 8).
	TF–IDF	With $k = 8$, more computationally expensive than K-means yet produces comparable cluster quality (See appendix A.3, Figure 8).

5.4 Best model

K-means algorithm is one of the unsupervised learning algorithms that, due to its efficiency and simplicity, is often chosen for grouping text documents (Barbara Probierz et al, 2022). Hence, we select TF-IDF and K-means as our best model. Indeed, K-means is less computationally intensive compared to GMMs and is faster to run. Also, by comparing every algorithm t-SNEs with Bag of Words or TF-IDF, we observe that TF-IDF enhances the performance of clustering.

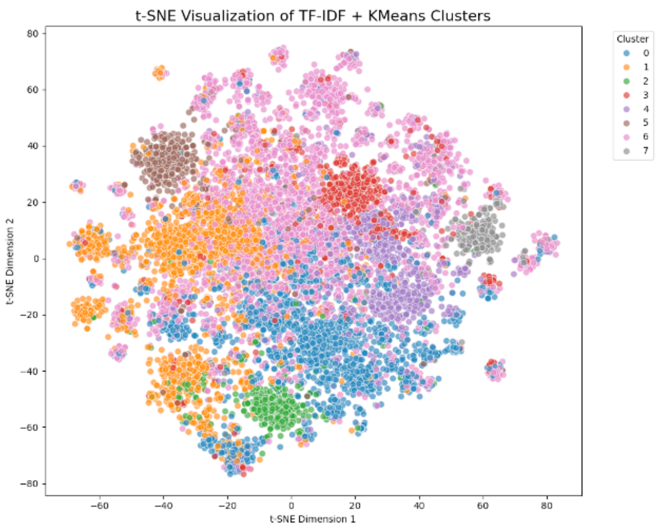


Figure 2: t-SNE Visualization for TF–IDF + K-means

5.5 Topic Modeling and Insights

Using our chosen embedding and clustering method, we then applied topic modeling to assign meaningful labels to each group of similar documents. Topic modeling distills each cluster into its most representative keywords. A common approach is Latent Dirichlet Allocation (LDA), which you initialize with the number of topics KK. LDA then determines how strongly each document reflects each topic and which words define those topics. See Appendix A.3, Figure 3 for LDA bar charts highlighting the top keywords in each cluster.

Domain knowledge and Chat GPT 4.0 were used to make sense of the different keywords per cluster and link them to potential main data science research topics. Table 2 shows results from a Keywords per cluster analysis, where keywords were added to a Chat GPT prompt to match them with research topics.

Table 2: Cluster topics and document counts

Cluster	Main ICLR Research Topics in 2025	Number of Documents
0	Time-Series Forecasting Analysis	258
1	Large Language Models (NLP & LLMs)	2648
2	Generative Image & Diffusion Models	1424
3	Graph Neural Networks	579
4	Reinforcement Learning	493
5	Deep Learning Training & Optimization	5511
6	Adversarial Machine Learning & Robustness	386
7	Video Generation & Temporal Modeling	373

Table 2 highlights popular research areas in 2025. The top 3 topics for ICLR 2025 submissions were respectively: Deep Learning Training & Optimization with 5511 submissions. Large Language Models (NLP & LLMs) with 2648 related papers and Generative Image & Diffusion Models with 1424 documents.By tracking how popular each research topic is year-to-year, you can match your reviewer pool to demand. Recruiting the right number of specialists in each area so that submissions on the most in-demand topics always high quality reviewing processes.

6 Feature Engineering

6.1 Min-Max Scaling

The dataset's numeric features vary widely in scale, which can hinder model training. To address this, Min-Max scaling was applied to all continuous predictors except the target variable, *Rating*. This rescales features to the [0, 1] range, ensuring balanced contributions during learning and preventing dominance by large-valued variables. Without scaling, models may converge slowly, become unstable, or produce biased results.

The *Rating* was excluded from scaling to maintain its fixed ordinal meaning and preserve evaluation integrity.

7 Train-Vali Split

The original dataset ICLR Training contains a total of 28, 048 observations. We have decided to split ICLR2025 train into three sets: 70% for training, 20% for validation, and 10% for testing. We will use a stratified split using the 'Rating' column to account for class imbalances. This chosen split maximises the impact of each subset by ensuring the training set is large enough to learn meaningful patterns from both text and numeric data, the validation set provides a reliable measure for tuning hyperparameters, and the test set can offer an unbiased estimate of final model performance.

8 Model Building

8.1 Defining Machine Learning Task

This analysis evaluated regression, classification, and clustering approaches for all the variables in the ICLR 2025 review dataset, combining numeric scores and text data. For more details, see Appendix A.4

- **Regression** using numeric features achieved a moderate RMSE of **1.30**, reflecting rating subjectivity and missing textual context.
- **Classification** grouped ratings into low, medium, and high classes, with a Random Forest achieving **73%** accuracy. Medium ratings had strong recall (86%), while high ratings were harder to predict (36% recall), highlighting class imbalance challenges.
- **Clustering** of abstracts via TF-IDF and K-Means revealed meaningful research themes, aiding exploratory analysis.

Overall, **classification** best balances interpretability and performance for rating prediction, with regression useful for fine-grained scores and clustering providing complementary insights. Future work should incorporate textual features to improve predictive accuracy. It is also important to note that, the values of the 'Rating' variable, our target output, are categorical, thus reinforcing the suitability of classification-based approaches.

8.2 Model Comparison RNN & LSTM

8.2.1 Model Description and Discussion

A Recurrent Neural Network (RNN) is designed to process sequential data by maintaining a hidden state that captures temporal dependencies (GeeksforGeeks, 2025). In this report, the RNN uses the tanh activation for hidden state updates, ReLU in the dense layer, and softmax at the output. The hidden state at time t is computed as:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{1}$$

While effective for short-term dependencies, RNNs suffer from vanishing gradients over long sequences. Long Short-Term Memory (LSTM) networks address this with memory cells and gated mechanisms controlling information flow (Mishra, 2020). An LSTM unit maintains a cell state C_t and gates: input i_t, forget f_t, and output o_t. Key updates include:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \quad h_t = o_t \odot \tanh(C_t) \tag{2}$$

The LSTM applies sigmoid activations to gates, tanh for cell state updates, ReLU for dense layers, and softmax for outputs. The hidden state is concatenated with numeric features and passed through a fully connected layer for final prediction. Full LSTM equations are detailed in Appendix **??**.

Random search was used for hyperparameter tuning, evaluated on a validation set with a fixed seed for reproducibility. The Adam optimizer was selected for its adaptive learning rates and stable convergence, suitable for these models. Sparse categorical cross-entropy served as the loss function, while the weighted average F1 score was used to address class imbalance and evaluate classification performance.

8.2.2 Hyperparameter Tuning

To fairly evaluate the performance upper bounds of both RNN and LSTM architectures, this report conducted separate hyperparameter tuning for each model using the same input variables and consistent hyperparameter ranges. The goal was to identify the best-performing architecture under optimal conditions.

Hyperparameter	**Values**
Embedding dimension (`embedding_dim`)	448, 512, 576, 640
LSTM hidden units (`lstm_hidden_units`)	32, 64, 96, 128
Dense units (`dense_units`)	32, 64, 96, 128
Dropout rate (`dropout_rate`)	0.3, 0.4, 0.5, 0.6, 0.7
Learning rate (`learning_rate`)	1×10^{-3}, 1×10^{-4}

Table 3: Summary of key hyperparameters tuned for RNN and LSTM models

8.2.3 Results and Comparision

After conducting hyperparameter tuning using `RandomSearch`, the optimized parameter set for the LSTM model includes: embedding_dim = 640, lstm_hidden = 32, dense_units = 96, dropout = 0.3, and learning_rate = 0.001.

For the RNN model, the optimized parameters are: embedding_dim = 640, rnn_hidden = 32, dense_units = 128, dropout = 0.5, and learning_rate = 0.001.

The results of the LSTM and RNN models are summarized in Table 4.

Table 4: Model performance comparison between RNN and LSTM

Model	Accuracy	Macro-F1 Score	Weighted-F1 Score
RNN	0.45	0.31	0.45
LSTM	0.50	0.34	0.49

From Table 4, it is evident that the LSTM model consistently outperforms the RNN across all evaluation metrics, achieving higher accuracy, macro-F1 score, and weighted-F1 score. This improvement highlights the LSTM's superior ability to capture long-term dependencies in sequential data. Consequently, the LSTM architecture was selected for further refinement.

8.3 Variable Set Comparison

To evaluate the effect of input features on model performance, three variable sets were constructed based on the optimized hyperparameters obtained in the previous section. These are referred to as Model A, Model B, and Model C, each containing different subsets of information fields, as shown in Table 5.

Table 5: Input feature sets for Models A, B, and C

Model	**Input Features**
Model A	title, abstract, summary, soundness, presentation, contribution, strengths, weaknesses, confidence
Model B	title, abstract, summary, strengths, weaknesses, confidence
Model C	summary, soundness, presentation, contribution, strengths, weaknesses, confidence

8.3.1 Model Results and Comparision

To select the variable set that yields the best performance, we compare the weighted average F1 score across three configurations.

Table 6: Performance comparison across different variable sets

Model	Accuracy	Macro-F1 Score	Weighted-F1 Score
Model A	0.539	0.351	0.532
Model B	0.354	0.191	0.322
Model C	0.522	0.353	0.513

From Table 6, it can be observed that Model B underperforms significantly across all metrics, while Models A and C achieve comparable results. This suggests that the inclusion of title and abstract in Model A does not lead to noticeable performance improvement. Therefore, the set of variables used in Model C is selected for the final model.

8.4 Final LSTM Model

After selecting LSTM as the optimal architecture, a second round of hyperparameter tuning was performed using the variable set in Model C to improve performance.

8.4.1 Hyperparameter Tunning

In this second round of hyperparameter tuning, the search space was refined based on the results of the initial tuning. Batch size is also tuned. The number of trials was increased to 20 to allow for more comprehensive exploration of the parameter space. To improve efficiency and prevent overfitting, early stopping with a patience of 10 was applied within a maximum of 50 training epochs.

Table 7: Hyperparameter Ranges for Second Round of LSTM Tuning

Hyperparameter	**Range / Values**
Embedding Dimension	576, 640
LSTM Hidden Units	32, 48, 64
Dense Units	64, 96, 112
Dropout Rate	0.2, 0.3, 0.4
Learning Rate	1×10^{-3}, 5×10^{-4}, 1×10^{-4}
Batch Size	16, 32, 64

8.4.2 Performance Evaluation

As shown in Table 8, the model performs well on classes 3 and 6, achieving relatively high F1-scores of 0.65 and 0.62 respectively. In contrast, performance on classes 1 and 10 is extremely poor, with all metrics at zero. This may result from significant class imbalanced where 1 and 10 have very few samples. This making it difficult for the model to learn meaningful patterns for these categories.

Class	Precision	Recall	F1-Score
1	0.00	0.00	0.00
3	0.56	0.79	0.65
5	0.47	0.31	0.37
6	0.55	0.72	0.62
8	0.72	0.30	0.42
10	0.00	0.00	0.00

Table 8: Classification performance for each class

Overall, the model achieved an accuracy of approximately 0.55, a macro-average F1-score of 0.35, and a weighted-average F1-score of 0.52. While the weighted F1-score reflects better performance due to the higher frequency of certain classes, the macro-average highlights that performance is inconsistent across categories. The train and validation plot can be seen in Appendix 10

8.5 Transformer Models - DistilBERT

8.5.1 Model Comparison

One big problem with LSTMs is the demand for a large amount of memory and processing power. According to (Greff et al., 2017), LSTMs are very complex in their architecture, resulting in slower training periods as the number of layers and hidden states increase. LSTMs are also prone to overfitting due to their high parameter count, especially in tasks involving natural language processing that frequently calls for large, high-dimensional datasets (Kandadi & Shankarlingam, 2025). This results in a model that performs well on the training data but generalizes poorly to unseen reviews.

Transformer models like BERT calculate an 'attention score' to evaluate the importance of words in a sentence in parallel to other words rather than on its own (Dogra et al., 2021). This makes transformer models particularly ideal to pre-train massive text datasets, resulting in significant improvements to accuracy in downstream tasks like text classification.

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{3}$$

where Q, K, and V represent the matrices of queries, keys, and values respectively, d_k is the dimension of the keys, and the softmax function ensures that attention weights sum to 1.

In this task, we will be using DistilBERT, a distilled version of the BERT transformer model which is smaller, faster, and less computationally expensive while still being able to retain 97% of its original performance. The DistilBERT model focuses on reducing the number of layers as modern investigations showed that variations on the last dimension of the tensor (hidden size dimension) have a smaller impact on computational efficiency (Sanh et al., 2019). Lastly, pre-trained models like BERT or DistilBERT is able to prevent overfitting for NLP tasks due to transfer learning. It prevents the need for extensive training on a specific dataset by transferring learned parameters and can provide useful features and representations of natural language that enhances model performance and accuracy.

For our implementation, we use the `DistilBertTokenizer` from Huggingface's Transformers library to tokenize inputs and enforce uniform sequence lengths efficiently. These tokenized inputs are passed into the `DistilBertT` transformer model, which processes tokens and attention masks to produce contextual embeddings. A custom `TensorFlow Keras` classification head then performs the prediction. We use sparse categorical cross-entropy as the loss function to support multi-class classification, and the Adam optimizer (Adaptive Moment Estimation) for efficient convergence.

8.5.2 Hyperparameter Tuning

DistilBERT Model Configuration

- 6 layers, 768 hidden units, 12 attention heads
- Adam optimizer with an initial learning rate of 1×10^{-4}
- Dropout rate of 0.4
- Batch size of 16 (to accommodate GPU memory)
- Maximum sequence length set to 384* (even though tokens may reach 6352), with truncation applied to optimize training efficiency

We implemented four enhancements in our custom Keras model:

- Model Checkpointing: Saves weights after each epoch to allow resuming from the best checkpoint in case of a training crash or failure.
- Early Stopping *: We stop training if validation loss does not improve for 3 consecutive epochs. This helps avoid unnecessary computation and prevents overfitting on the training set.
- Reduce Learning Rate on Plateau *: We reduce the learning rate when progress slows down or a metric has stopped improving. This allows us to make smaller, more precise updates to weights to avoid overshooting the minimum and avoid unnecessary epoch updates
- Apply class weights: We calculate class weights to account for imbalanced classes in Task D. In LSTMs, class weights did not improve Macro and Weighted F-1 scores. However, class weights show significant improvement in the F-1 scores of rare classes like 1 and 10 in DistilBERT.

** These settings have a direct impact on training time per epoch. For quantitative results, see Appendix* **??**.

8.5.3 Results and Comparison

Now we compare the performance of Model A and Model B using the best-performing architecture from Task C, LSTM, against the transformer-based DistilBERT model. Due to GPU constraints, each model was trained for 20 epochs. The feature sets for each model are as follows:

- **Model A:** summary, soundness, presentation, contribution, strengths, weaknesses, confidence
- **Model B:** summary, strengths, weaknesses, confidence

The results are summarised in the table below:

Table 9: Model Performance Comparison for Model A and Model B

Model	Architecture	Accuracy	Macro-F1 Score	Weighted-F1 Score
Model A	LSTM	0.522	0.353	0.513
Model A	DistilBERT	0.537	0.372	0.519
Model B	LSTM	0.354	0.191	0.322
Model B	DistilBERT	0.348	0.187	0.317

For Model A, both LSTM and DistilBERT achieved similar accuracy and weighted f-1 scores. However, there was a slight improvement in the macro f-1 score using the transformer model. This means that DistilBERT is able to handle imbalanced classes better. In Model B, the performance of the LSTM and DistilBERT are the same. The absence of numeric review features such as soundness, presentation, and contribution seems to lower the model's predictive ability.

Overall, Model A clearly outperforms Model B, showing that a combination of variables `soundness + presentation + contribution` and reviewer comments leads to better performance. These numerical features carry valuable information that complement textual data. DistilBERT's capacity in handling large-scale datasets allows it to handle more features more effectively, thus improving their ability to differentiate different classes. The full classification report for Model A is in Appendix 15, and training and validation plots is in Appendix 11.

The comparison between the two models LSTM and DistilBERT highlights that the feature set is a more significant driver of model performance than the choice of model architecture. Including numeric variables alongside text inputs leads to better predictive performance and more balanced performance across classes. The performance of both models is very similar, but transformer-based models like DistilBERT offer additional benefits on computational efficiency.

9 Limitations

There were several limitations identified across the datasets given, methods to collect data for the ICLR, and models used for prediction:

- The dataset contains a significant class imbalance, especially for extreme ratings like 1 (Strong Reject) and 10 (Strong Accept). This makes it difficult for models to learn meaningful patterns from underrepresented classes, reducing predictive accuracy for those categories.
- Scores for criteria like `Soundness`, `Presentation`, and `Contribution` are inherently subjective. Similar submissions may receive different evaluations depending on the reviewer's interpretation or bias. Although steps like marking the same paper by three different markers have been introduced, it still doesn't guarantee submissions are free from subjectivity.
- Both LSTMs and DistilBERT suffer from computational constraints:
  - LSTMs are computationally intensive due to their sequential processing and struggle with long input sequences. Even with bi-directionality, they may miss broader context compared to transformer-based models.
  - Although DistilBERT is lighter than full BERT, it remains resource-intensive. Input truncation to 384 tokens can lead to the loss of important information from longer reviews.
- Although text preprocessing steps like lowercasing, punctuation removal, and lemmatization are inherently important for models to learn, they can unintentionally delete neccessary information. They can distort meaning or remove emotions in marker reviews. This is one of the limitations of extracting information on textual data.

10 Conclusion

This study explored a range of machine learning models to predict reviewer ratings in the ICLR 2025 peer review dataset, integrating both textual feedback and numeric evaluation metrics. The findings demonstrate that combining structured scores such as soundness, presentation, and contribution with reviewer comments significantly improves predictive performance.

Among the evaluated architectures, both LSTM and DistilBERT achieved comparable results in terms of accuracy and weighted F1 scores when using the full feature set (Model A). However, DistilBERT achieved a higher macro F1 score, reflecting a stronger ability to handle class imbalance and rare rating categories. In contrast, models trained on limited feature sets (Model B) showed noticeably weaker performance, regardless of the architecture, underscoring the importance of including numeric review scores.

While LSTMs are effective for sequence modeling, they are computationally intensive and sensitive to long input sequences. DistilBERT, as a lighter transformer-based model, offers greater computational efficiency, scalability, and robustness across imbalanced classes making it more suitable for ICLR 2025 involving large-scale peer review data. Overall, the DistilBERT-based Model A is recommended for reviewer rating prediction tasks. It balances high predictive accuracy with the ability to manage class imbalance and efficiently process both text and numeric data. This project contributes a scalable and interpretable approach to peer review analysis, with potential to support fairer, more consistent, and more transparent academic evaluation processes.

References

Chen, W., Yang, K., Yu, Z., Shi, Y., & Chen, P. (2024, 05). A survey on imbalanced learning: latest research, applications and future directions. *Artificial intelligence review*, *57*. doi: 10.1007/s10462-024-10759-6

Church, K. W. (2016). Word2vec. *Natural Language Engineering*, *23*(1), 155–162. Retrieved from https://doi.org/10.1017/S1351324916000334 doi: 10.1017/S1351324916000334

Dogra, V., Singh, A., Verma, S., Kavita, Jhanjhi, N., & Talib, M. (2021, 01). Analyzing distilbert for sentiment classification of banking financial news. *Analyzing DistilBERT for Sentiment Classification of Banking Financial News*, 501-510. doi: 10.1007/978-981-16-3153-5_53

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the second international conference on knowledge discovery and data mining (kdd)* (pp. 226–231).

GeeksforGeeks. (2025). *Introduction to recurrent neural network.* Retrieved from https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (Accessed: 2025-05-26)

George, S., & Vasudevan, V. (2020). Comparison of lda and nmf topic modeling techniques for restaurant reviews. *Indian Journal of Natural Sciences*, *10*(62), 28210–28216. Retrieved from https://doi.org/10.1515/jisys-2018-0299 doi: 10.1515/jisys-2018-0299

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017, 10). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*, 2222-2232. doi: 10.1109/tnnls.2016.2582924

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, *31*(8), 651–666. Retrieved from https://doi.org/10.1016/j.patrec.2009.09.011 doi: 10.1016/j.patrec.2009.09.011

Jolliffe, I. T. (2002). *Principal component analysis* (2nd ed.). Springer.

Kandadi, T., & Shankarlingam, G. (2025, 01). Drawbacks of lstm algorithm: A case study. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5080605 doi: 10.2139/ssrn.5080605

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1746–1751). Retrieved from https://doi.org/10.3115/v1/D14-1181 doi: 10.3115/v1/D14-1181

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*. Retrieved from https://arxiv.org/abs/1506.00019

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*. Retrieved from https://arxiv.org/abs/1802.03426

McLachlan, G., & Peel, D. (2000). *Finite mixture models.* Wiley.

Mishra, D. (2020). *Lstm and its equations.* Retrieved from https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af (Accessed: 2025-05-26)

Patil, R., Boit, S., Gudivada, V. N., & Nandigam, J. (2023). A survey of text representation and embedding techniques in nlp. *IEEE Access*, *11*, 36120–36135. Retrieved from https://doi.org/10.1109/ACCESS.2023.3266377 doi: 10.1109/ACCESS.2023.3266377

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.* Retrieved from https://arxiv.org/abs/1910.01108

Silva, M. D. (2023, 04). *Preprocessing steps for natural language processing (nlp): A beginner's guide.* Retrieved from https://medium.com/@maleeshadesilva21/preprocessing-steps-for-natural-language-processing-nlp-a-beginners-guide-d6d9bf7689c9

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*, 2579–2605. Retrieved from http://jmlr.org/papers/v9/vandermaaten08a.html

Yan, D., Li, K., Gu, S., & Yang, L. (2020). Network-based bag-of-words model for text classification. *IEEE Access*, *8*, 82641–82652. Retrieved from https://doi.org/10.1109/ACCESS.2020.2991074 doi: 10.1109/ACCESS.2020.2991074

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*. Retrieved from https://arxiv.org/abs/1511.08630

A Appendix

ChatGPT Declaration

We acknowledge the use of ChatGPT [`https://chat.openai.com/`] to help with coding portion for this assessment and LaTeX formatting.

Table 10: Team Member Contributions

Team Member	**Tasks**
530825650	Data Cleaning & Pre-Processing (Reporting) Task A – Rating Distribution Task D – DistilBERT (Code + Reporting) Limitations (Reporting)
530598846	Abstract + Introduction + Literature Review Task A (Reporting) Task A (Model Building) Feature Engineering Conclusion
500680807	Task B (Code + Reporting) Task C
530672928	Task C (Code + Reporting)

A.1 Keyword Frequency Analysis

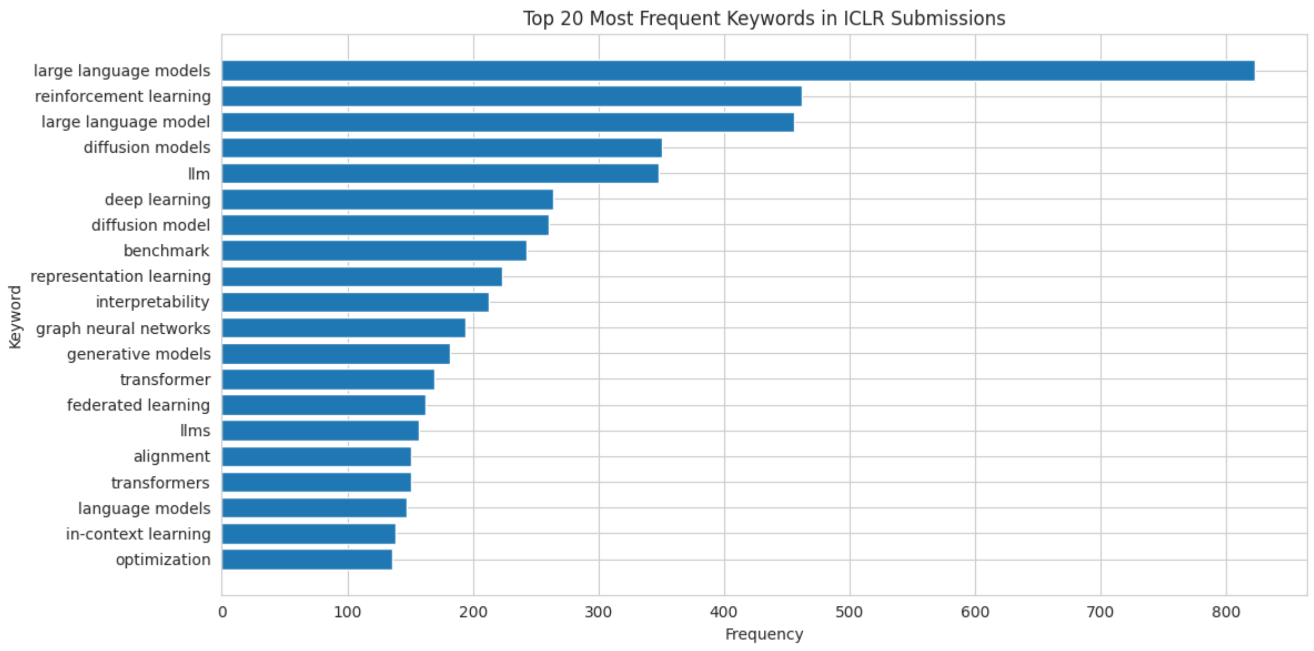


Figure 3: Top 20 Most Frequent Keywords in ICLR Submissions

A.2 LDA Keywords

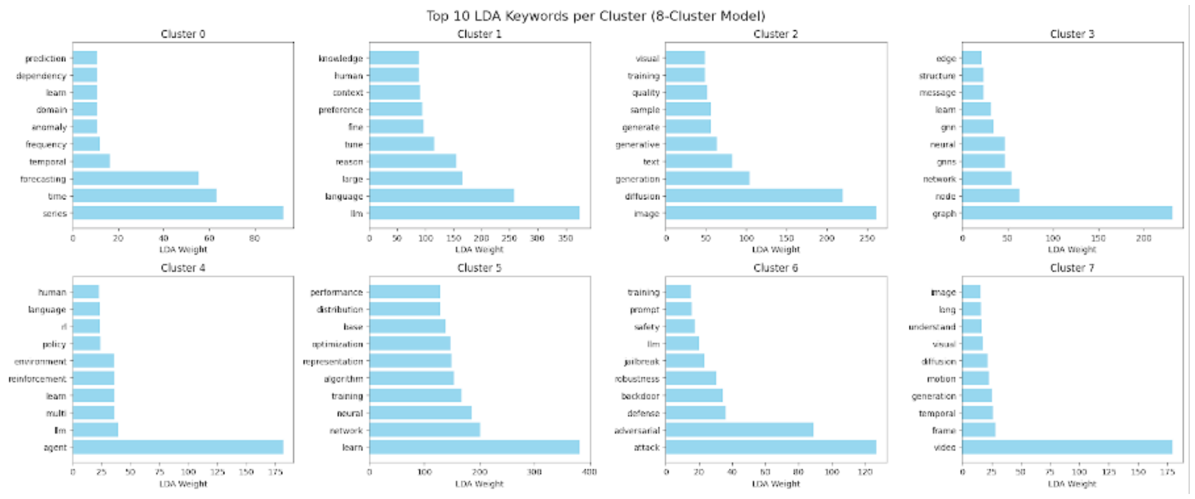


Figure 4: LDA keyword importance per cluster

A.3 Clustering Plots

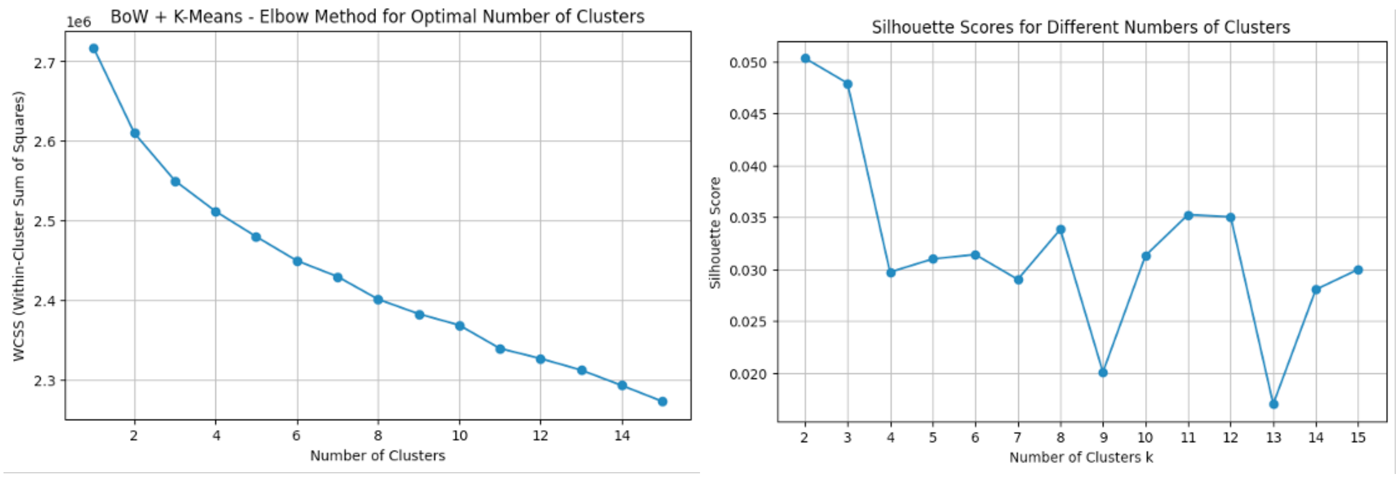


Figure 5: Side-by-side comparison of BoW + K-means diagnostics

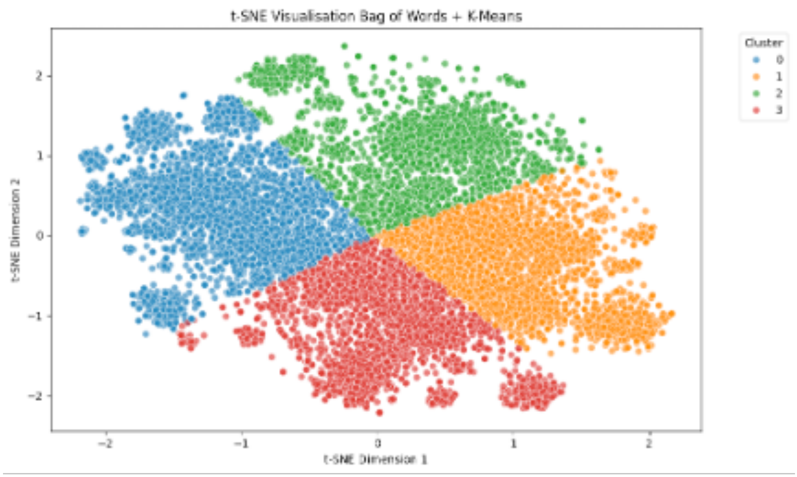


Figure 6: t-SNE visualization for BoW + K-means

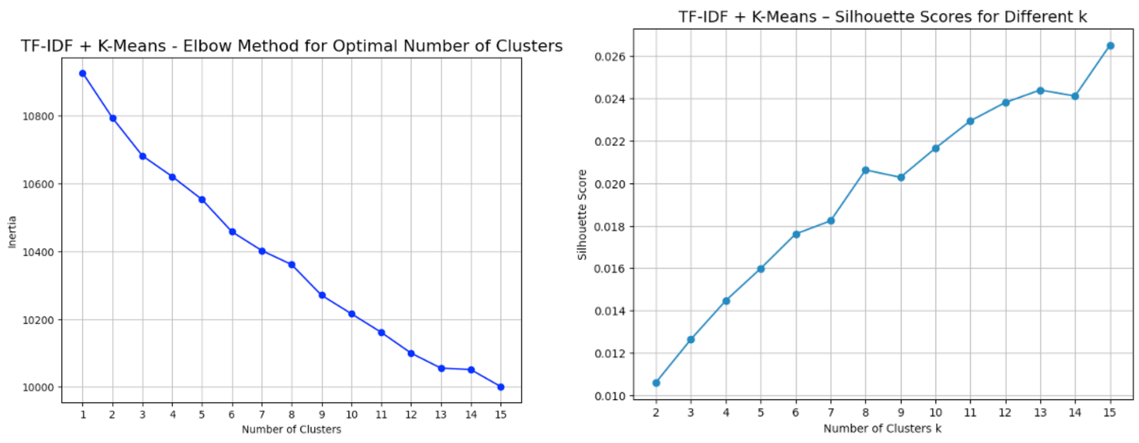


Figure 7: Side-by-side comparison of TF–IDF + K-means diagnostics: (left) Elbow method, (right) Silhouette analysis

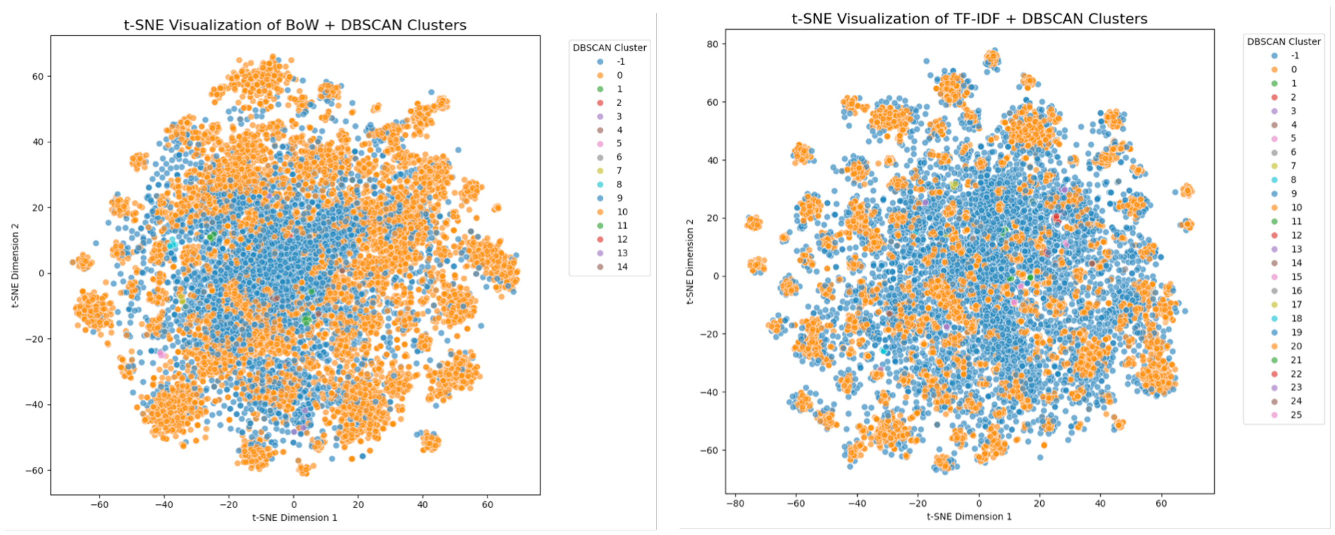


Figure 8: Side-by-side comparison of DBSCAN's t-SNE with BoW & TF–IDF

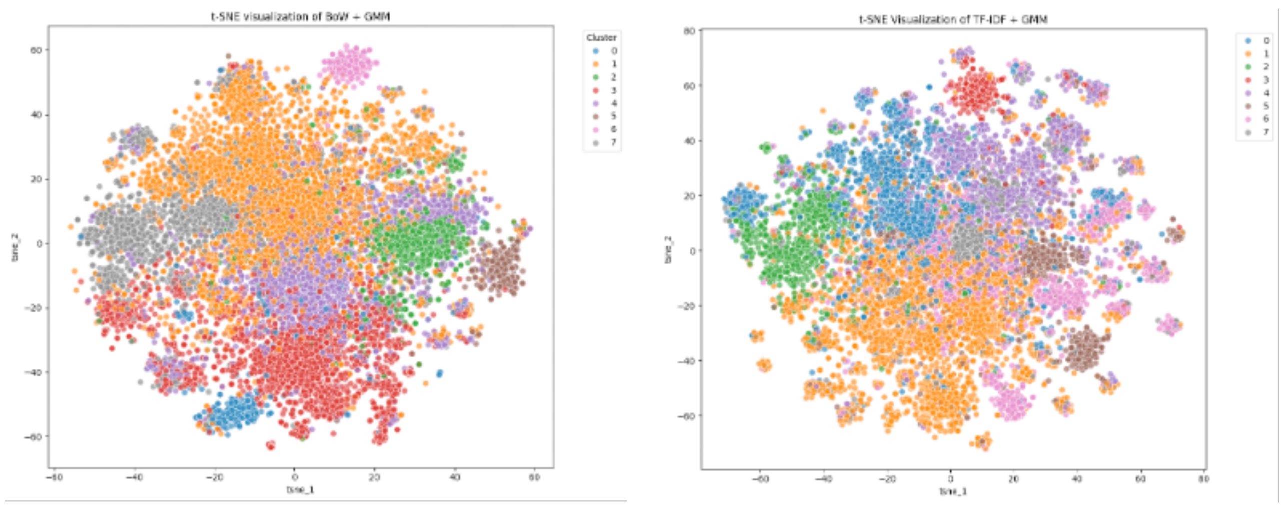


Figure 9: Side-by-side comparison of GMMs's t-SNE with BoW & TF–IDF

A.4 ML Task

Metric	**Value**
Root Mean Squared Error (RMSE)	1.3042

Table 11: Regression performance using RMSE.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.73	0.65	0.69	1529
1	0.73	0.86	0.79	3301
2	0.70	0.36	0.48	780
Accuracy		0.73		
Macro Avg	0.72	0.62	0.65	5610
Weighted Avg	0.73	0.73	0.72	5610

Table 12: Classification performance across three classes.

Abstract (truncated)	Cluster
Visual hallucination (VH) occurs when a multim...	2
Predicting simple function classes has been wi...	3
Adversarial training often suffers from a robu...	1
Neural networks that map between low dimension...	1
The covariance for clean data given a noisy ob...	0

Table 13: Sample abstract texts and their assigned clusters.

A.5 Training and Validation Loss Final LSTM Model

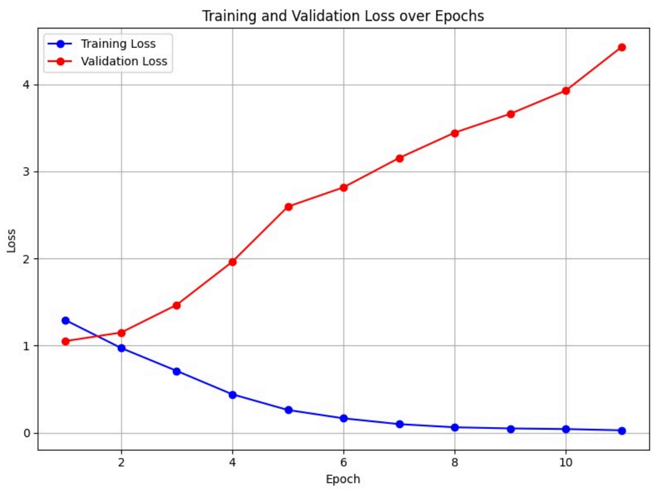


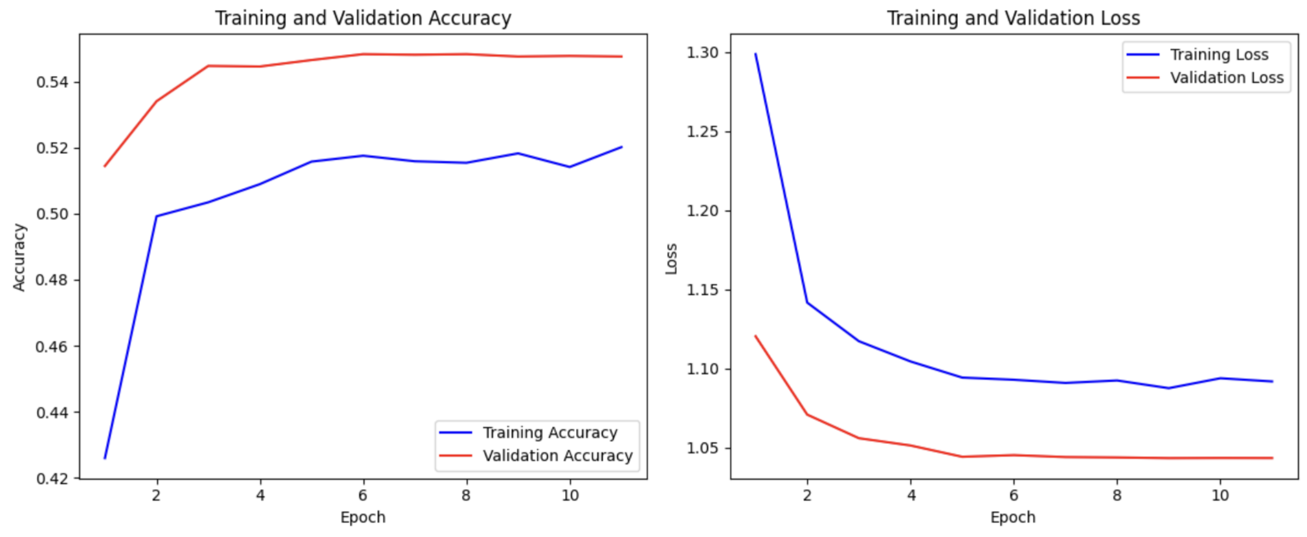
Figure 10: Training and Validation Loss Final LSTM Model

A.6 Epoch Training Time

Table 14: Epoch 1 Training Time (Model A) Under Different Settings

Setting	Epoch 1 Time
Without any fine-tuning	366 ms/step
Reduce batch size	Limited by GPU memory
Introduce early stopping	366 ms/step
Lower `max_length` to 384	263 ms/step

A.7 Train-Vali Plot of DistilBERT Model A



```
Lowest Validation Loss: epoch 9
Highest Validation Accuracy: epoch 6
```

Figure 11: Train and Validation Plot of Model A (DistilBERT)

A.8 Classification Report of DistilBERT Model A

Label	Precision	Recall	F1-score	Support
1	1.00	0.08	0.15	62
3	0.57	0.74	0.65	692
5	0.46	0.35	0.40	786
6	0.52	0.66	0.58	882
8	0.65	0.36	0.46	373
10	0.00	0.00	0.00	10

Table 15: Classification Report For DistilBERT