Write me: extracting data from emails

Author: Evgenia Fomina

Supervisor: Professor Alfio Ferrara

Computer Science Department, University of Milan, Italy

Introduction

Email remains one of the most prevalent forms of digital communication, both in personal and professional contexts. Despite its common usage, the automatic extraction of meaningful information from email messages remains a complex task, due to the semi-structured nature of the content and the diverse communicative intents embedded in text.

Research question

The goal of this project is to develop a machine learning-based pipeline capable of extracting and classifying content from email messages into structural and non-structural categories.

The focus is on identifying:

- Structural elements such as pleasantries and email signatures.
- Non-structural elements including arguments and topical content within the body of emails.

Dataset

For my project I used Enron email dataset which contains of approximately 500,000 emails generated by employees of the Enron Corporation. The dataset is structured in the following format:

	file	message
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...
3	allen-p/_sent_mail/1000.	Message-ID: <13505866.1075863688222.JavaMail.e...
4	allen-p/_sent_mail/1001.	Message-ID: <30922949.1075863688243.JavaMail.e...

Project structure

The project is organized into the following key sections: data preparation, topic modeling, and the development of models for extracting pleasantries, email signatures, and argumentative sentences.

1. Data Preparation

During the data preparation phase, the dataset was parsed into key components such as sender, receiver, subject, and email body. Recurrent patterns that do not contribute to the core content—such as email headers like “Forwarded by” or similar boilerplate text—were identified and removed to ensure cleaner and more meaningful input for subsequent analysis.

2. Data preprocessing

Text preprocessing involved cleaning and preparing the email content to ensure consistency for analysis and model training. This included steps such as lowercasing all text, removing punctuation, numbers, URLs, and stopwords, as well as applying lemmatization, stemming, and tokenization. Additionally, each email was segmented by lines—for signature extraction—and by sentences—for

identifying pleasantries and argumentative content. These steps help standardize the textual data and improve the effectiveness of downstream processing and modeling tasks.

For the preprocessing step, I utilized the *spaCy* and *NLTK* libraries.

3. Dataset for the models

To build models for extracting pleasantries, signatures, and argumentative sentences, I applied supervised learning methods, while for topic modeling, I employed unsupervised learning algorithms.

For the supervised tasks, I constructed specific labeled datasets as follows:

Pleasantries Extraction. A binary classification dataset was created by identifying and labeling sentences that contain pleasantries (e.g., greetings and polite expressions) and sentences that do not. Each sentence was annotated with a label indicating the presence or absence of a pleasantry.

	sentence	label
0	please cc the following distribution list with...	1
1	please plan to attend the below meeting\n\n\n ...	1
2	thanks\nrain\nx.	0
3	tim\n\nmike grigsby is having problems with ac...	0
4	can you please make sure he has an active pass...	1

Signature Extraction. Since email signatures typically appear at the end of the message, I extracted the last six lines from each email body to create a candidate pool of signature lines. These lines were then manually labeled to indicate whether they belonged to the signature or not, forming a labeled dataset for supervised training.

body_lines	signature_label
can you send me a schedule of the salary and ...	0
scheduling group. plus your thoughts on any ch...	0
(patti s for example)	0
phillip	1
for delivered gas behind san diego, enron ene...	0
enron entity. i have forwarded your request to...	0
number is 713-853-7107.	0
phillip allen	1

Argumentative Sentence Classification. For argument mining, a multi-label classification dataset was constructed. Each sentence was annotated based on its argumentative function, using the following set of categories:

- Claim: A statement expressing a point of view or an assertion that requires support.
  Example: I believe that remote work improves productivity.
- Premise: A reason or piece of evidence provided to support a claim.
  Example. Because employees have fewer distractions at home, they can focus better.

- Major Claim: The central thesis or main argument of the text.
  Example. It is undeniable that climate change is the greatest challenge of our time."
- Support: A sentence that provides additional justification or reinforcement for another argument component.
  Example. Studies show that employees who work remotely are more productive.
- Contrastive: A sentence that introduces an opposing perspective or highlights a contrast with previously stated ideas.
  Example. The project was successful, but it took longer than expected.
- Causal: A sentence that explains a cause-and-effect relationship between concepts or events.
  Example. The meeting was delayed because the manager was stuck in traffic.

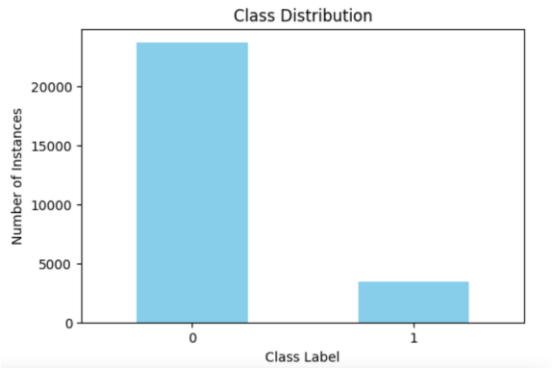
Each sentence in the dataset was labeled with one or more of these categories, enabling a multi-label classification approach for training the argument mining model. Each sentence could belong to one or more of these categories simultaneously, allowing for multi-label annotation.

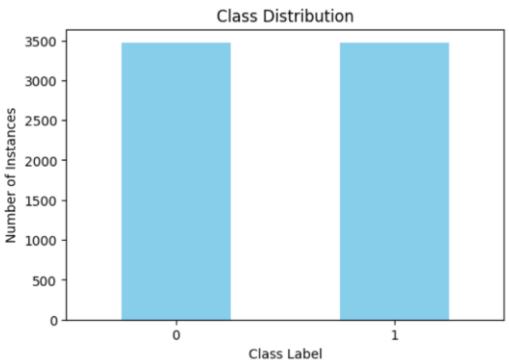
labels	is_premise	is_claim	is_major_claim	is_support	is_contrastive	is_causal
0	1	1	0	1	0	0
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	1	0
4	0	1	0	0	0	0
5	1	1	0	1	1	0
6	1	1	0	0	0	0
7	0	0	1	0	0	0
8	1	1	0	1	1	1
9	0	1	1	0	0	0
10	1	1	0	1	0	1
11	0	0	1	0	1	0
12	1	0	1	1	0	0

4. Train and test datasets

Each dataset used for model training was divided into two subsets: a training set and a test set.

For the task of pleasantry extraction, the dataset exhibited a significant imbalance between the two classes—pleasantries and non-pleasantries. To address this issue, I applied the downsampling of the majority class technique. This method involves reducing the number of samples in the majority class to match the number of samples in the minority class, thus creating a balanced dataset and preventing the model from being biased toward the dominant class.





For the task of argumentative sentence classification, I used the stratified sampling technique during the train-test split. Stratification ensures that the distribution of each class label is preserved in both the training and test sets, which is particularly important in multi-label classification tasks to maintain representative class proportions across splits.

Since the dataset exhibits class imbalance, we applied class weighting during training. This technique assigns higher weights to underrepresented classes and lower weights to majority classes, ensuring that the model does not become biased toward more frequent classes and learns to recognize minority class instances more effectively.

5. Build the models

Pleasantry Extraction: To extract sentences containing pleasantry phrases I use traditional machine learning models such as Naive Bayes and Support Vector Machines (SVM), I applied the TF-IDF (Term Frequency–Inverse Document Frequency) technique for text representation in the form of numerical feature vectors. Since they cannot directly process raw text, TF-IDF transforms each sentence into a fixed-length vector that captures the importance of each term in the context of the sentence and the corpus.

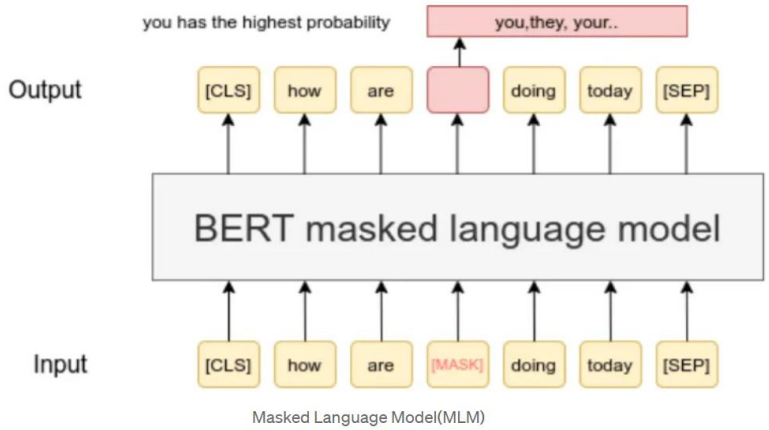
Naïves bayes algorithms is based on the theorem which states the chances that the class will occur to be true by multiplying the probable chances. This way the hypothesis will occur true given certain scenarios.

The core idea behind SVM is to find the optimal hyperplane (a line) that best separates the data into two distinct classes while maximizing the distance (the margin) between the nearest data points of each class, known as support vectors.

Signature Extraction. I used the BERT pre-trained model. BERT architecture incorporates a mechanism called self-attention, allowing BERT to weigh the significance of each word based on its context, both preceding and succeeding. Before its using, it needs to be prepared and structured in a way that it can understand, including tokenization, input formatting, and the Masked Language Model (MLM) objective.

```
Input IDs: torch.Size([64])
Attention Mask: torch.Size([64])
Labels: torch.Size([])
Length of dataset: 1000
{'input_ids': tensor([  101,  2064,  2017,  4604,  2033,  1037,  6134,  1997,  1996, 10300,
         1998,  2504,  1997,  3071,  1999,  1996,   102,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0]), 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]), 'label': tensor(0)}
```

During its training, some words are masked (replaced with [MASK]) in sentences, and BERT learns to predict those words from their context. This helps BERT grasp how words relate to each other, both before and after.



Masked Language Model(MLM)

The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words.

```
Epoch 1, Loss: 0.3564252107590437
Epoch 2, Loss: 0.15370155446231365
Epoch 3, Loss: 0.10244566595181823
```

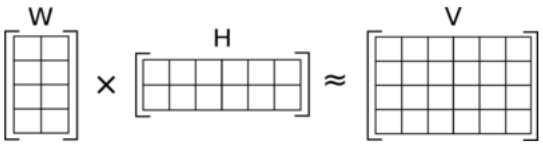
The results of loss function in my model are decreasing from 0.3564 to 0.1024 over 3 epochs that is a good indicator that the model is learning effectively.

Argumentative Sentence Classification. Since each sentence can exhibit multiple argumentative roles (e.g., claim, premise, support), we formulated the task as a multi-label classification problem and used a BERT-based model to capture the contextual semantics. The model was trained using a binary cross-entropy loss function suited for multi-label scenarios. The training loss showed a consistent decrease across epochs, indicating effective learning.

```
Epoch 1, Loss: 0.8268726985643976
Epoch 2, Loss: 0.13064806519781288
Epoch 3, Loss: 0.04757325501614221
```

Topic modelling. For topic modelling i used two models: LDA and NMF. NMF (Non-Negative Matrix Factorization) reduces the dimension of the input corpora. It uses factor analysis method to provide comparatively less weightage to the words with less coherence.

For a general case, consider we have an input matrix V of shape m x n. This method factorizes V into two matrices W and H, such that the dimension of W is m x k and that of H is n x k. For our situation, V represent the term document matrix, each row of matrix H is a word embedding and each column of the matrix W represent the weightage of each word get in each sentences (semantic relation of words with each sentence).



LDA assumes that documents are made up of a mixture of topics, and each topic is a distribution over words. It aims to reverse-engineer this process to discover the latent topics underlying the corpus. In the context of LDA, documents are represented as a bag-of-words model, where the order of words is disregarded, and only the frequency of words matters. Topics are latent (hidden) thematic structures that represent sets of words that frequently occur together in documents.

The model assumes that each word in a document is generated by randomly choosing a topic from the document's topic distribution and then randomly choosing a word from the chosen topic's word distribution.

6. Model evaluation

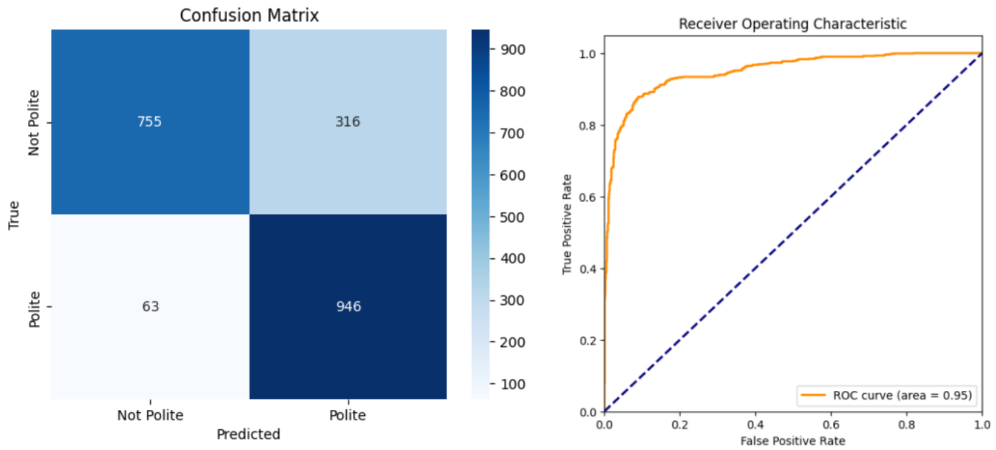
Pleasantries Extraction. I evaluated Naïve Bayes Classifier. The model achieved an overall accuracy of 82%, indicating that it correctly classified the majority of the 2080 test instances.

The model is very precise (0.92) when it predicts class 0 and slightly lower (0,75) when it predicts 1, meaning some false positives occur. Also it misses some true class 0 instances (higher false negatives).

Balanced at 0.80 for class 0 and 0.83 for class 1, suggesting good harmonic balance between precision and recall.

ROC AUC Score: 0.95 – an excellent indicator that the model discriminates well between the two classes overall.

	precision	recall	f1-score	support
0	0.92	0.70	0.80	1071
1	0.75	0.94	0.83	1009
accuracy			0.82	2080
macro avg	0.84	0.82	0.82	2080
weighted avg	0.84	0.82	0.82	2080



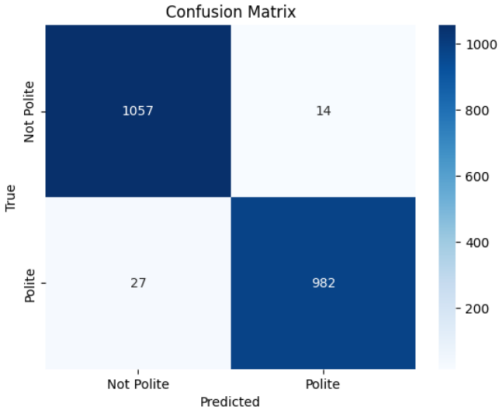
The evaluation of the SVM model shows the better results. The overall accuracy reached 98%, indicating that the model correctly classified the vast majority of instances in the test set.

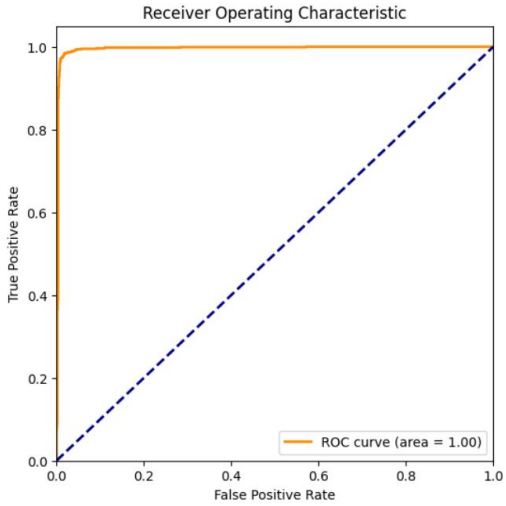
The precision, recall, and F1-score for both classes were also consistently high (all around 0.98), reflecting a strong balance between sensitivity and specificity. Specifically:

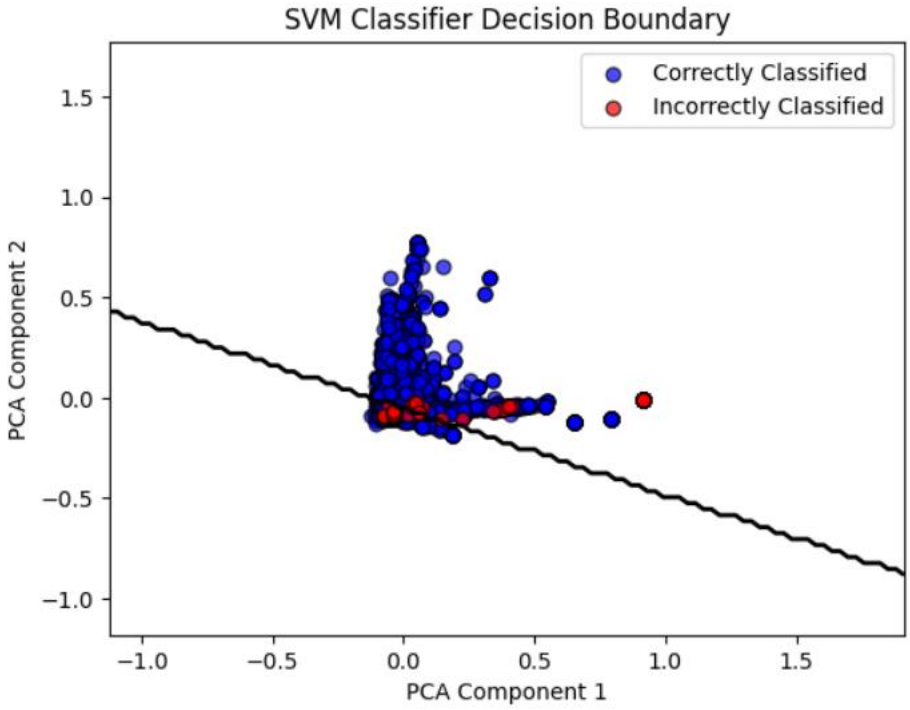
The confusion matrix confirms this performance, showing very few misclassifications (14 false positives and 27 false negatives out of 2080 samples).

Finally, the ROC curve achieved an AUC of 1.0, suggesting the model has excellent discriminative ability and performs optimally in distinguishing between the two classes.

	precision	recall	f1-score	support
0	0.98	0.99	0.98	1071
1	0.99	0.97	0.98	1009
accuracy			0.98	2080
macro avg	0.98	0.98	0.98	2080
weighted avg	0.98	0.98	0.98	2080



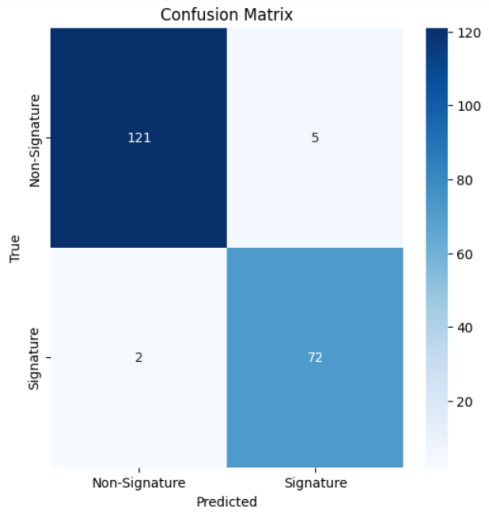


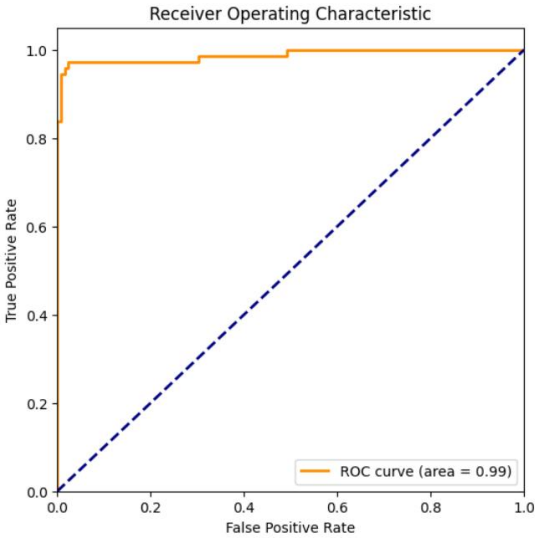


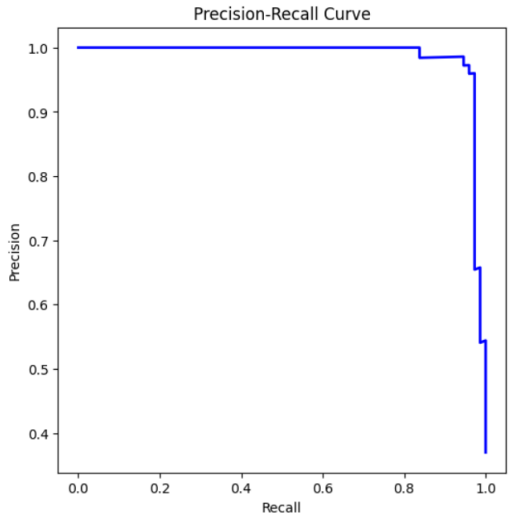
Signature Extraction. I evaluated the BERT model for extracting signatures. The evaluation of the model for signature extraction demonstrates strong overall performance. The model achieved an accuracy of 96.5%, indicating that the majority of email segments were correctly classified.

The confusion matrix confirms this performance, with only 5 false positives and 2 false negatives out of 200 total samples, demonstrating the model's reliability.

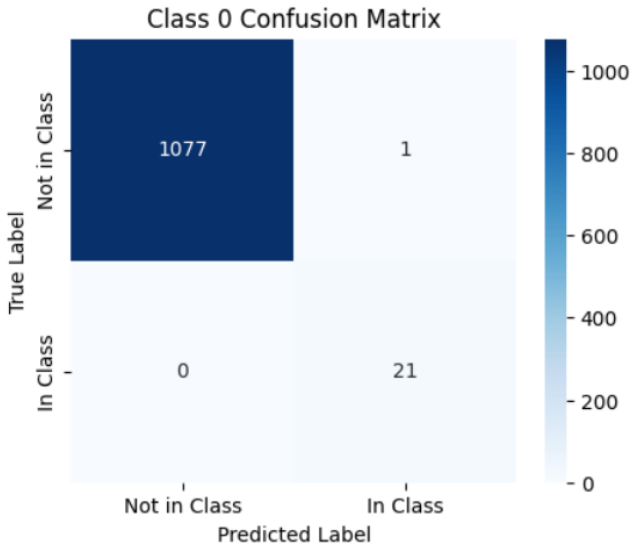
Moreover, the ROC curve AUC score of 0.99 reflects an excellent ability to distinguish between signature and non-signature segments.

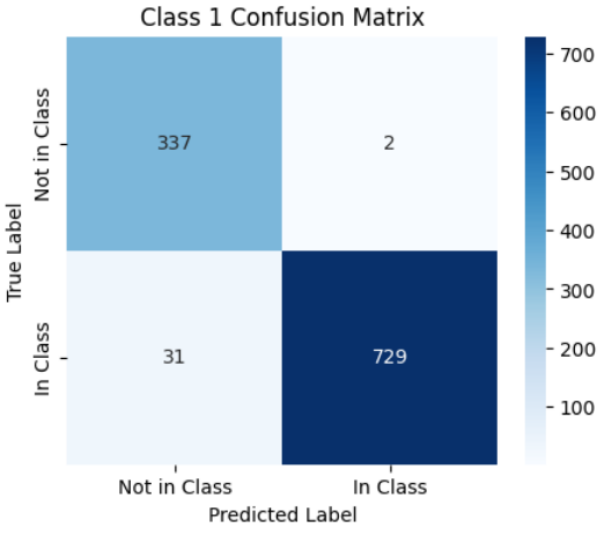


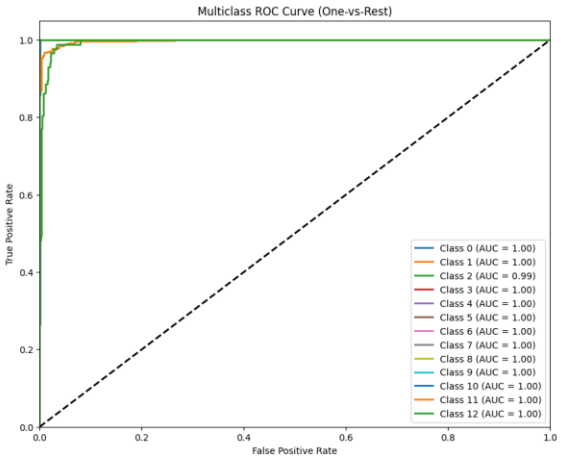


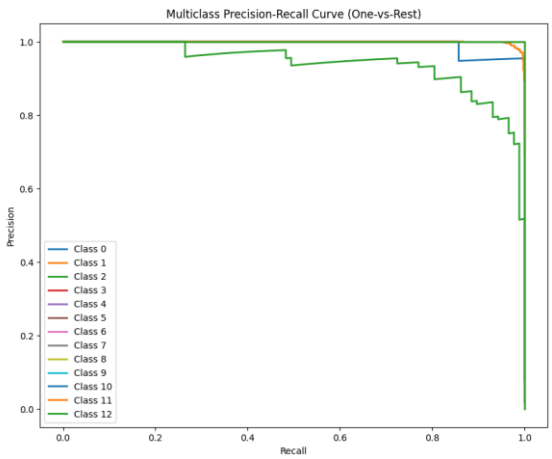


Argumentative Sentences Classifier.







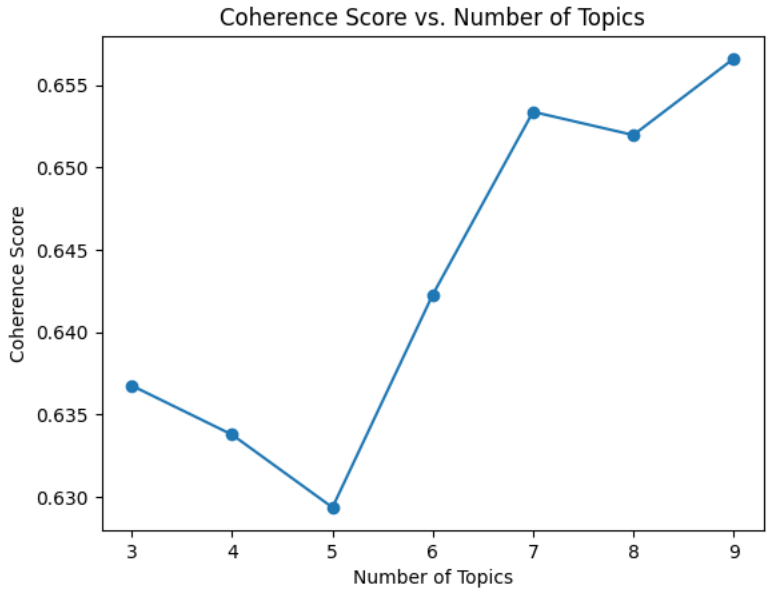


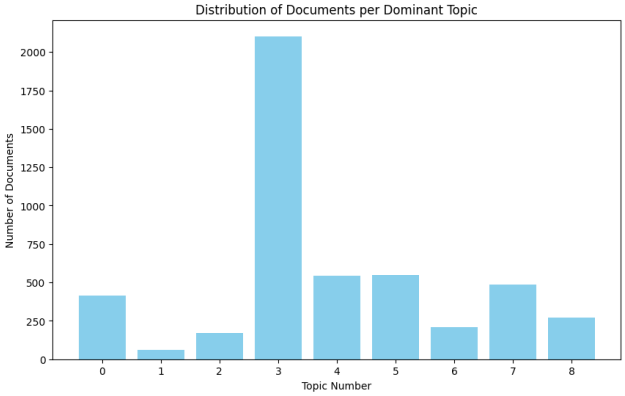
Topic Modelling. To determine the optimal number of topics for both the LDA and NMF models, I evaluated the coherence scores across different topic numbers. The coherence score measures how similar these words are to each other.

Additionally, I analyzed the topic distribution to understand how topics are spread across the documents. This evaluation is important because it allows us to assess the relevance and uniqueness of each topic. A balanced topic distribution suggests that each topic is well-represented, while an imbalanced distribution may indicate that some topics are more dominant than others. By observing how topics are assigned to documents, we can gain insights into the granularity of the topics and their real-world applicability.

LDA

```
Optimal number of topics: 9 with coherence score 0.656589925289154
```





NMF

```
Optimal number of topics: 8 with coherence score 0.5837067392372496
```