# Mini Project 3: Classification of Textual Data

Jonathan Halimi, Thien Pham, Rafid Saif

November 17, 2023

## 1 Abstract

The significance of machine learning algorithms in improving the accuracy of trend predictions and classifications is garnering increasing attention. This project focuses on a widely used method for textual data classification, specifically, emotion detection. The technique that helps machines and computers detect, express, and understand emotions is known as emotional intelligence. We will implement Naive Bayes from scratch and Bidirectional Encoder Representations from Transformers (BERT) model on the Emotion datasets with pertained weights through package and fine-tuning, and compare their performance on accuracy.

# Contents

# 2 Introduction

The field of machine learning has witnessed remarkable advancements, particularly in the domain of textual classification, one of the most common tasks in Natural Language Processing (NLP). Its applications not only range from tagging customer feedback into categories, but to routing support tickets according to their language. In this mini-project, we implement the Naive Bayes model from scratch and the Bidirectional Encoder Representations from Transformers (BERT) model with pertained weights through package and fine-tuning. The two algorithms are computed for multip-classification experiments to compare their performance accuracy on the Emotion dataset. This dataset includes English Twitter messages with six basic emotions: anger, fear, joy, love, sadness, and surprise. Our goal is to apply existing machine learning libraries and gain a deeper understanding of the underlying mechanisms by building our solutions from scratch when necessary.

# 3 System Models

In our project, we aim to develop models capable of automatically identifying emotional states, such as anger or joy, expressed in text. To achieve this, we focused on experimenting with two distinct types of models: the state-of-the-art deep learning model BERT (Bidirectional Encoder Representations from Transformers) and the traditional Naive Bayes classifier.

BERT represents the cutting edge in natural language processing. Leveraging the Transformer architecture, it utilizes attention mechanisms to contextualize each word in a sentence, understanding its meaning based on surrounding words. This approach is particularly well-suited for complex tasks like emotion detection where context is key. We experimented with various initialization states and hidden layer configurations of BERT, fine-tuning our model to optimize its performance for the task at hand. Our experiments were greatly facilitated by the Hugging Face ecosystem and PyTorch, the latter enabling us to utilize GPU acceleration to enhance performance [2].

In contrast, Naive Bayes is a probabilistic model that applies Bayes' theorem under the assumption of independence between features. We implemented this model from scratch, experimenting with different alpha values to fine-tune its performance. Despite its computational efficiency, a notable advantage of Naive Bayes, our findings indicated that it was less accurate compared to the BERT model for emotion detection tasks. This difference highlights the trade-off between computational simplicity and the sophisticated, context-aware analysis offered by models like BERT.

# 4 Experiments and Conclusion

## 4.1 Datasets

The Emotion dataset from HuggingFace includes Twitter messages with six different basic emotions: anger, fear, joy, love, sadness, and surprise. Hence, we'll be treating this problem as a multiclass classification problem [1].

## 4.2 Preprocessing method

In our project, we tailored the preprocessing of text data differently for the Naive Bayes and BERT models to suit their specific computational needs.

For the Naive Bayes method, our approach centred on converting unstructured text into a numerical form that the algorithm could interpret effectively. This transformation was achieved using the bag-of-words model. This model treats text as a collection of words, disregarding their order but focusing on their frequency. Each document in our dataset is thus transformed into a vector, with each dimension representing the count of a particular word from the vocabulary in that document. This frequency-based representation aligns well with the probabilistic nature of Naive Bayes, facilitating efficient processing and classification.

For the BERT model, a more sophisticated preprocessing technique was necessary to align with its advanced contextual understanding capabilities. We utilized the transformer package for tokenizing the input text, a crucial step given BERT's reliance on its specific tokenization method. This process not only converts the text into token IDs but also creates additional features like attention masks and token type IDs, which are vital for BERT to comprehend the text's structure and meaning accurately. The inclusion of special tokens to denote the start and end of sentences, as well as for padding and separation, was also an integral part of this preprocessing stage. Leveraging the Huggingface interface within PyTorch, we efficiently executed the tokenization and conversion of

text into numerical features, preparing the data in a format that is optimally suited for the intricate workings of the BERT model [3],[4], [5].

### 4.3 Experiments settings

In this section, we detail the experiment settings for both the Naive Bayes and BERT models. Our objective is to provide comprehensive information so that our experiments can be accurately replicated.

For the Naive Bayes classifier, we implemented the model from scratch in Python. We used the bag-of-words model for feature extraction. Regarding the Naive Bayes model itself, we applied Laplace smoothing with a smoothing parameter ("alpha") of 1. This choice was made to mitigate the problem of zero probabilities in our dataset, a common issue with categorical data.

In contrast, for the BERT model, we utilized the pre-trained "bert-base-uncased" model from Hugging Face's transformers library. The model was fine-tuned on our dataset with the following hyperparameters: the learning rate was set to 5e-5, and the number of training epochs was set to 3. These choices were informed by common practices in the field and preliminary experiments. For the tokenization process, we adhered to BERT's tokenizer with a maximum sequence length of 128 tokens. This length was chosen to encompass the majority of our text data while keeping computational requirements in check. We also employed a batch size of 8 for training, balancing the trade-off between computational demand and model convergence. See Figure 2.

We used a single NVIDIA 3060ti as well as Google Collab for training our models. It's important to note that the performance and training time can vary significantly with different hardware configurations. Therefore, replicating our experiments on different hardware may lead to slightly different results or require adjustments in the hyperparameters, especially the batch size and learning rate for the BERT model. To improve our findings, we also used a model with Kaiming initialized weights, and the default model, with only 6 hidden layers and 8 attention heads, compared to 12 hidden layers and 12 attention heads previously. This is because we assume that given the simplicity of our task, we would like to avoid overfitting.

### 4.4 Experiments results

The data presented in Table 1 and 2 below, the performance of the Naive Bayes and BERT-based models on accuracy.

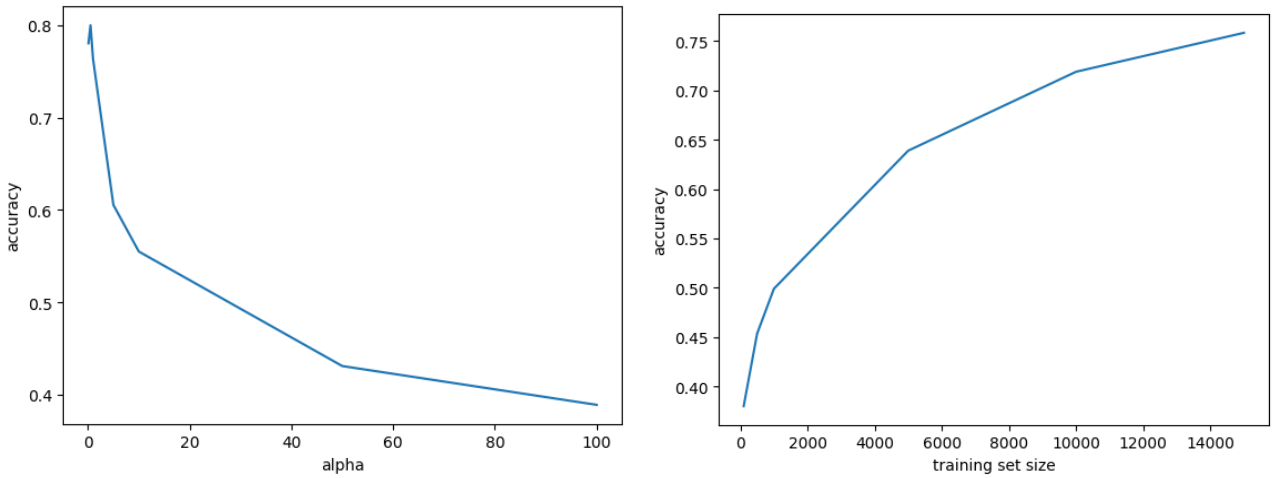
Table 1: Naive Bayes model performance result

| $\alpha$ | 0.1 | 0.5 | 1 | 5 | 10 | 50 | 100 |
|---|---|---|---|---|---|---|---|
| Accuracy | 0.7805 | 0.8000 | 0.7635 | 0.6055 | 0.5550 | 0.4310 | 0.3890 |
| Avg. F1 Score | 0.7049 | 0.6645 | 0.5590 | 0.2948 | 0.2394 | 0.1641 | 0.1299 |

Table 2: Default BERT performance per epoch

| Epoch | 0 | 1 | 2 |
|---|---|---|---|
| Accuracy | 0.9285 | 0.9055 | 0.9225 |
| Avg. F1 Score | 0.8742 | 0.8542 | 0.8786 |
| Modified Model Accuracy | 0.848 | 0.8835 | 0.8835 |
| Modified avg. F1 Score | 0.7896 | 0.8323 | 0.8158 |

In our comparative analysis, the fine-tuned BERT-based model demonstrated superior performance in terms of accuracy and F1 scores attributed to several factors. Firstly, the BERT architecture's deep bidirectional nature allows it to understand the context of words in text more effectively than traditional models. Secondly, our fine-tuning process, which included adjusting the learning rate and training epochs specifically for our dataset, allowed the model to adapt more precisely to the nuances of the Emotion classification task. More results plots can be seen in Figure 5 and Figure 6 in the Appendix. In contrast, while the Naive Bayes model was computationally less intensive, its simpler probabilistic approach and reliance on bag-of-words feature representation limited its ability to grasp complex linguistic patterns, leading to lower performance metrics. The standard BERT-based model, without fine-tuning, performed better than Naive Bayes but could not outperform the customized adjustments made in our BERT-based model. These results underscore the importance of model architecture complexity and task-specific fine-tuning in achieving high performance in NLP tasks. While the structurally modified BERT model achieved better accuracy than the default BERT model, it is outperformed in terms of other metrics such as F1 score.




(a) (b)

Figure 1: a) Naive Bayes with different $\alpha$ and b) different training set sizes

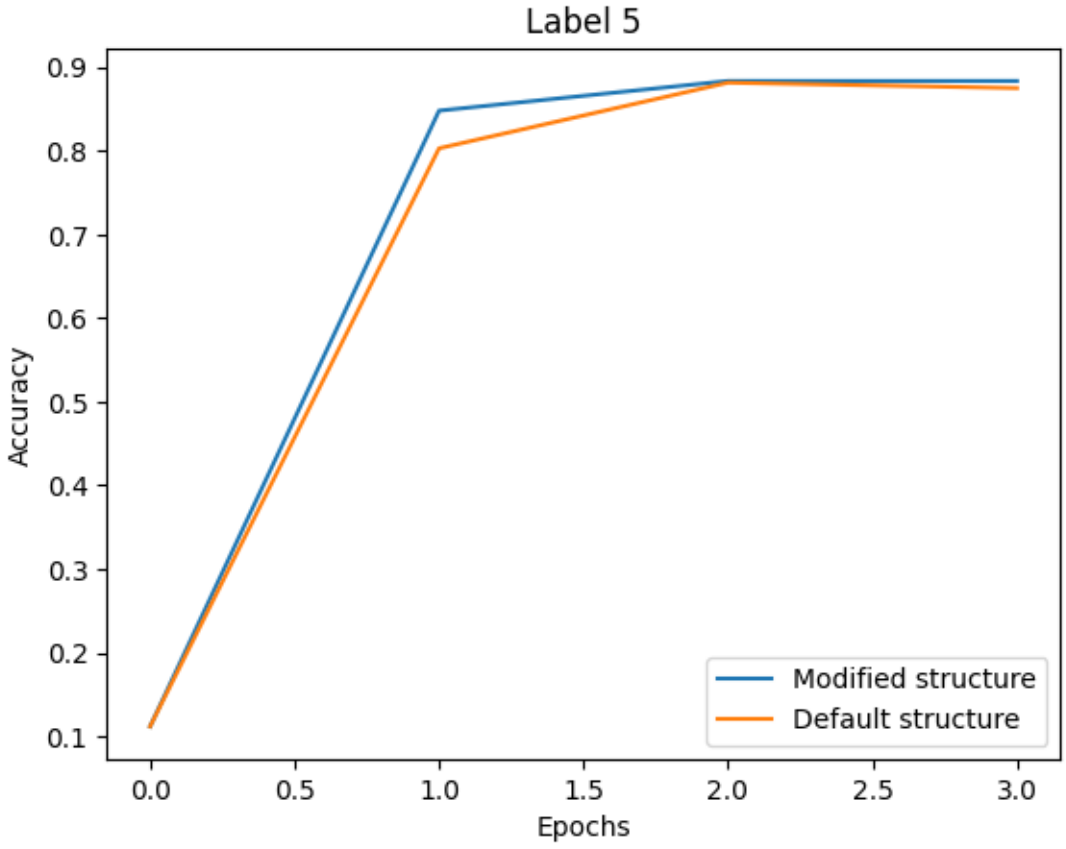



Figure 2: Performance of BERT with default trained model and trained model with 6 layers

### 4.5 Analysis of Attention Matrices

In our analysis, we focused on examining the attention matrices within the BERT model to understand how the model processes and interprets input text for emotion classification. We chose a specific transformer block and attention head from the multi-layer, multi-headed transformer architecture of BERT. This choice was informed by preliminary experiments aimed at identifying the layers and heads that exhibited the most informative attention

patterns for our task.

We visually inspected the attention patterns for correct and incorrect. This analysis involved mapping the attention weights between different words in the input text and the class tokens. By doing so, we aimed to uncover how different words or phrases contributed to the model's final decision, and how these contributions differed between successful and unsuccessful predictions. Figure 3 shows the visualization of a document that was correctly labelled after training. We can see more complex relationships captured after training for both the original and the modified BERT models. For the model initialized with Kaiming weights, we do not see this and instead see much more noisy results.

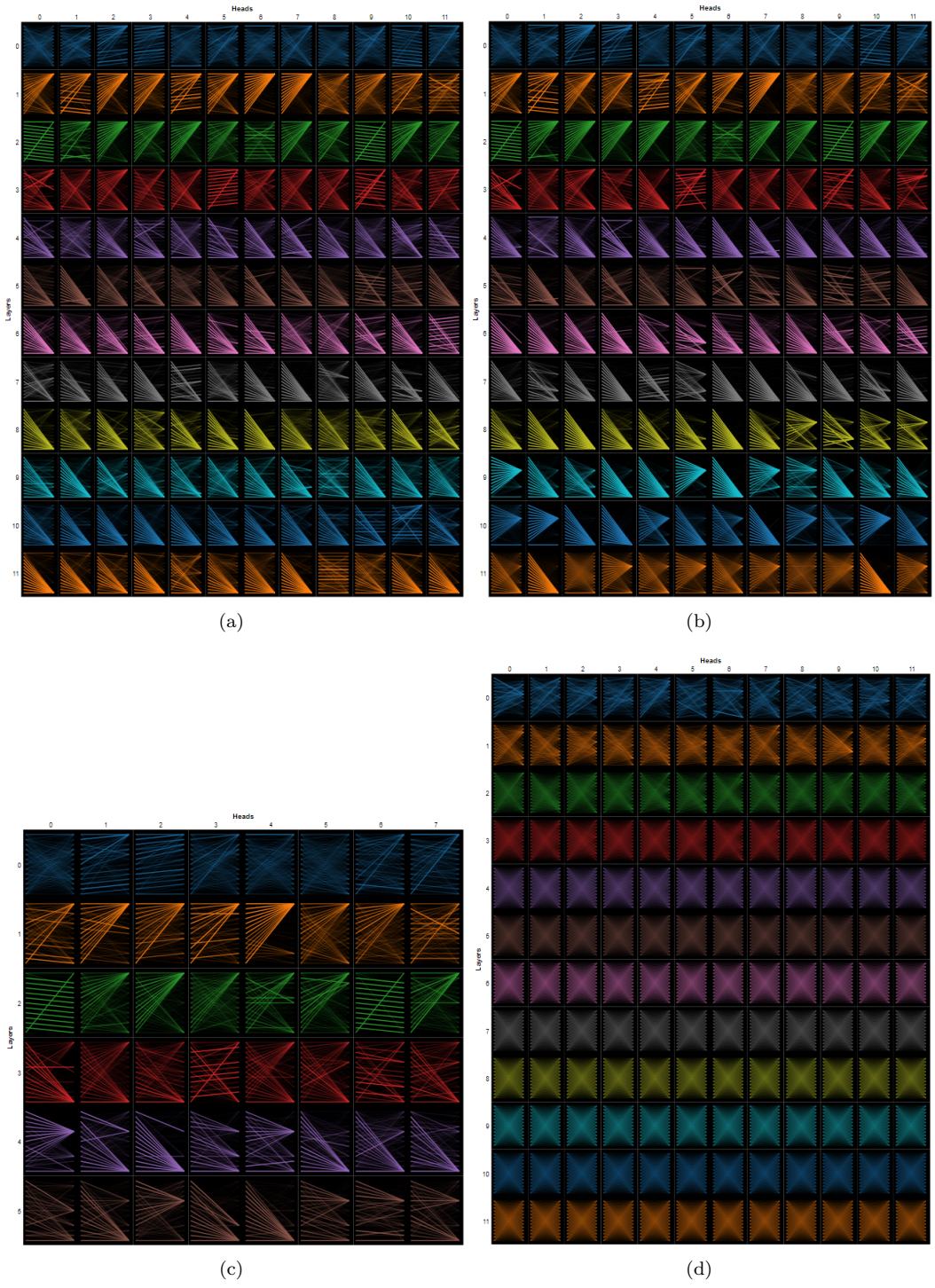



Figure 3: BERT model visualization (a) Default Untrained, (b) Default Trained, (c) 6 layers, (d) Kaiming Trained

# 5 Discussion and Conclusion

## 5.1 Impact of Pretraining on Emotion Prediction

Pretraining on an external corpus, as is the case with BERT, appears to be highly beneficial for the Emotion prediction task. This pretraining allows the model to develop a rich understanding of language and context before it is fine-tuned on the specific task of emotion detection. This is because the model embeddings are trained for contextual and positional coherence. We observed that this pretraining enables the model to capture subtle nuances in language that are crucial for accurately identifying emotions, a capability less pronounced in models without such extensive pretraining. In contrast, we initialized the model with random weights and embeddings using Kaiming initialization, which was one of the best-performing models in assignment 1. We see that after the same training, the accuracy is significantly lower than training the pre-trained model, with a final accuracy of 0.3475. In short, the pre-trained model is already trained in the language, only needing fine-tuning for the specific emotion classification task.

## 5.2 Deep Learning vs. Traditional Machine Learning Methods

The performance difference between deep learning (as exemplified by our BERT model) and traditional machine learning methods (like our Naive Bayes model) was significant. Deep learning models, particularly those utilizing transformer architectures like BERT, have a distinct advantage in their ability to process and interpret the context and complexities of natural language. Their multiple layers and attention mechanisms allow for a more nuanced understanding of text, leading to higher accuracy in tasks like emotion prediction. On the other hand, while traditional methods like Naive Bayes are simpler and more computationally efficient, they lack the depth required to fully grasp the intricacies of human language, resulting in lower performance on this task.

## 5.3 Conclusions

In this mini-project, we embarked on a journey to explore the world of textual classification using Naive Bayes and BERT models on the Emotion datasets. Through a series of experiments and analyses, we gained valuable insights into the impact of various factors on model performance.

We found that BERT classifier is a better choice for sentiment prediction tasks like on the Emotions dataset compared to Naive Bayes. BERT can understand the context of words in the text more effectively than traditional models including the entire context of the input text, relationships between words, and their positions in the sentence. This is due to its architecture's deep bidirectional nature that allows it to understand the context of words in text more effectively.

In contrast, while the Naive Bayes model was computationally less intensive, its simpler probabilistic approach and reliance on bag-of-words feature representation limited its ability to grasp complex linguistic patterns, leading to lower performance metrics.

In conclusion, this project highlights the significant influence of design decisions on textual data classification, specifically, emotion detection. Our observations emphasize the importance of making deliberate choices in neural network design and training. The project allowed us to gain a deeper understanding of the underlying mechanisms by building our solutions from scratch when necessary.

# 6 Statement of Contributions

- Jonathan Halimi - Implemented BERT, visualization and wrote this report.
- Thien Pham - Computed data tables, wrote this report.
- Rafid Saif - Implemented naive Bayes, ran experiments of BERT.

# 7 Bibliography

## References


[1] *Emotion Dataset*. URL: https://huggingface.co/datasets/dair-ai/emotion.

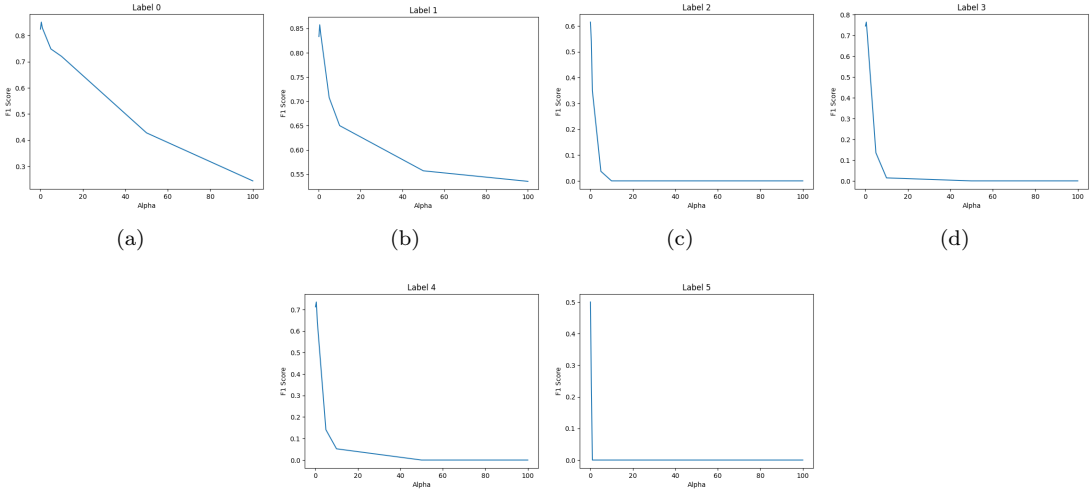
[2] *Pre-trained Bert-models*. URL: https://huggingface.co/bhadresh-savani/bert-base-uncased-emotion.

[3] *Scikit-learn function CountVectorizer1*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[4] *Transformers Package*. URL: https://pytorch.org/hub/huggingface%20pytorch-transformers/.

[5] *Transformers Training*. URL: https://huggingface.co/docs/transformers/training.

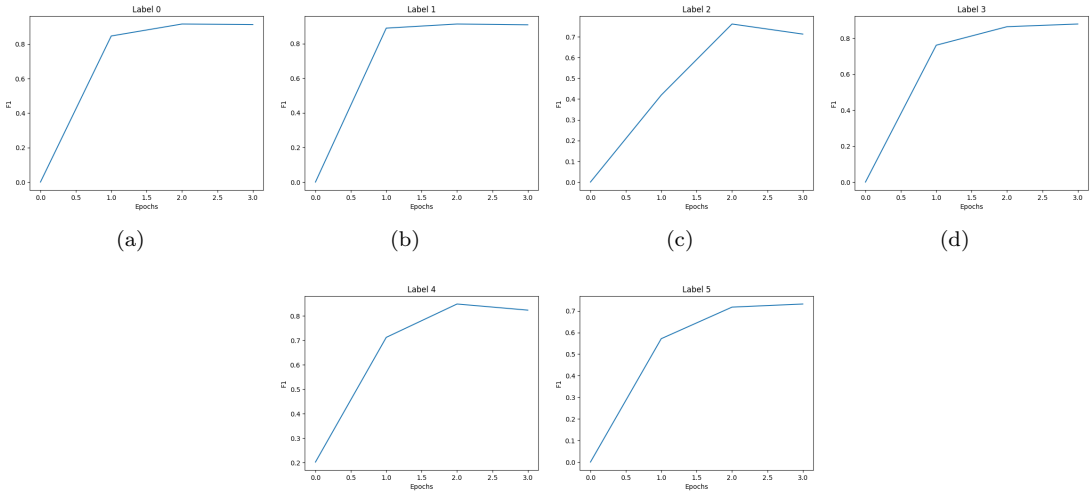
# 8 Appendix A - List of Figures




(a) (b) (c) (d)

(e) (f)

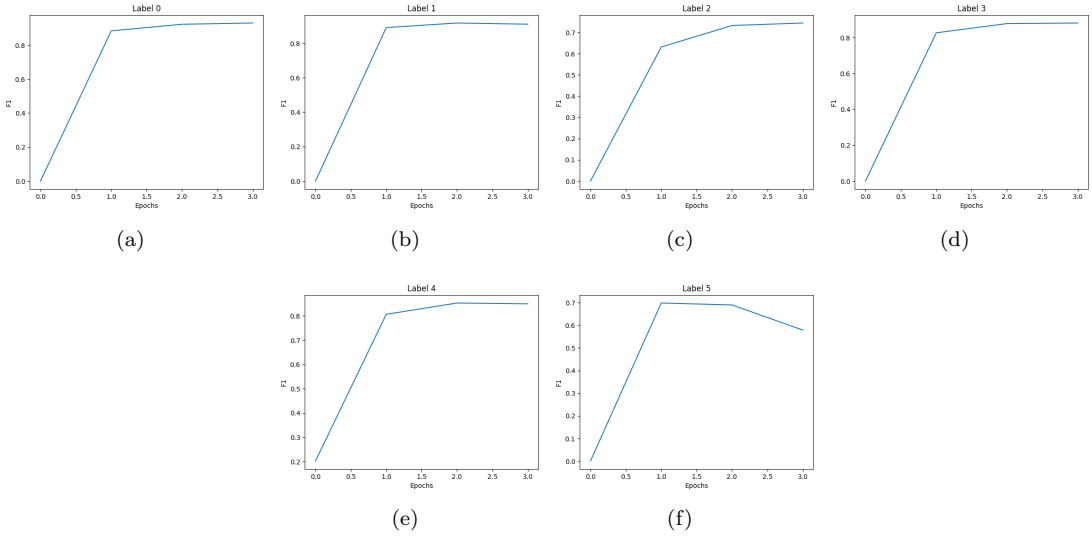
Figure 4: Effect of smoothing on F1 score for naive Bayes




(a) (b) (c) (d)

(e) (f)

Figure 5: Default BERT model F1 score over training epochs




(a) (b) (c) (d)

(e) (f)

Figure 6: Modified BERT model F1 score over training epochs