ATHENS UNIVERSITY OF ECONOMICS & BUSINESS

M.Sc. in Data Science

Text Analytics

Instructor: Ion Androutsopoulos

Assignment 06 - Exercises on Natural Language Processing with Transformers

Anagnos Theodoros (p3352323) - Michalopoulos Ioannis (p3352314) - Kafantaris Panagiotis (p3352328) - Vigkos Ioannis (p3352326)

16.6.2024

Contents

1 Exercise 01

The implementation details and code can be accessed through the following Colab notebook link: Link here

2 Introduction

This report presents the implementation and results of fine-tuning a pre-trained BERT model for text classification and named entity recognition using HuggingFace Transformers. The objective is to enhance the performance of sentiment classification on a labeled text dataset.

2.1 Dataset

We used a Twitter dataset from Kaggle for our sentiment analysis task. The dataset includes tweets labeled with sentiment categories: positive, negative, and neutral. The dataset statistics are as follows:

- Number of records: *4870*
- Number of classes: 3 (Positive, Negative, Neutral)

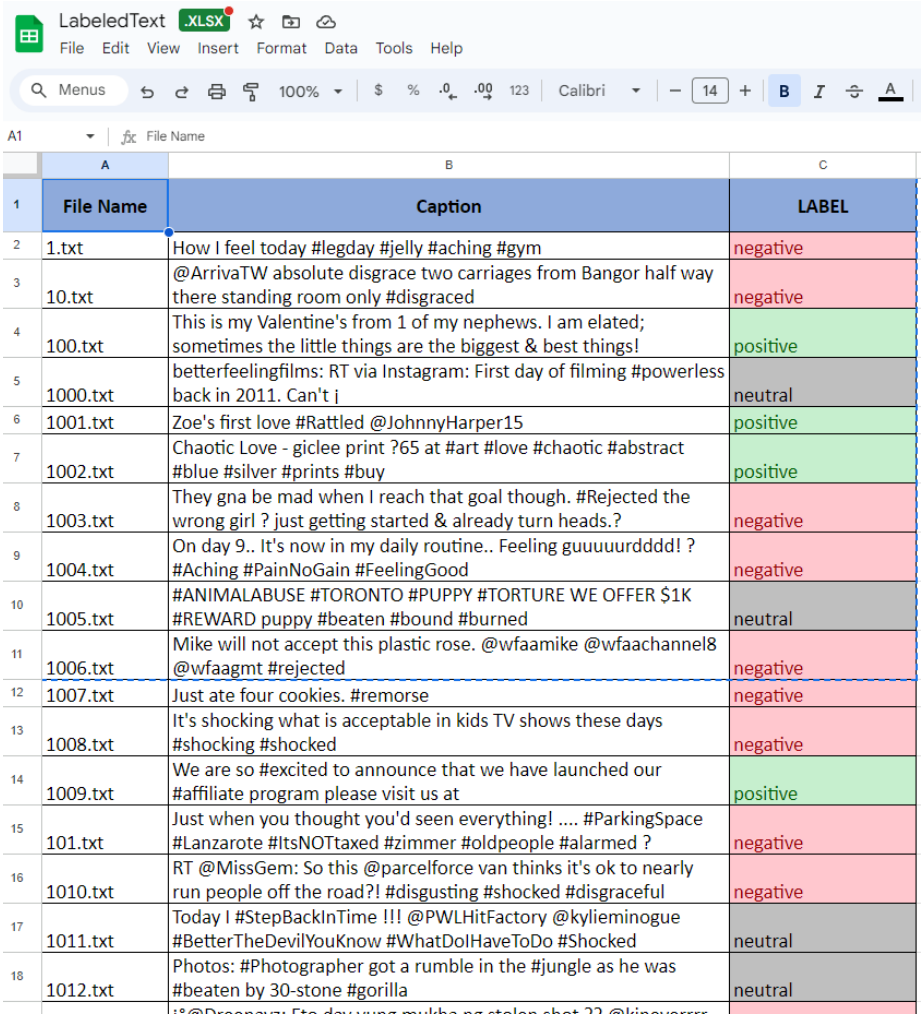
Provenance:

- This dataset enriches a benchmark dataset available at: `https://mcrlab.net/research/mvsa-sentiment-analysis-on-multi-view-social-data/`

Collection Methodology:

- This dataset is enriched by augmenting a dataset of images and captions with sentiment analysis-generated labels. A method was put forth to determine the ideal sentiment using the caption feature to implement six different sentiment analysis techniques. Final classification of each dataset item is based on the percentage of positive, negative, and neutral polarities extracted for each caption.

Sample:



LabeledText .XLSX

File Edit View Insert Format Data Tools Help

Menus 100% $ % .0 .00 123 Calibri 14 B I A

A1 File Name

	A	B	C
1	File Name	Caption	LABEL
2	1.txt	How I feel today #legday #jelly #aching #gym	negative
3	10.txt	@ArrivaTW absolute disgrace two carriages from Bangor half way there standing room only #disgraced	negative
4	100.txt	This is my Valentine's from 1 of my nephews. I am elated; sometimes the little things are the biggest & best things!	positive
5	1000.txt	betterfeelingfilms: RT via Instagram: First day of filming #powerless back in 2011. Can't ¡	neutral
6	1001.txt	Zoe's first love #Rattled @JohnnyHarper15	positive
7	1002.txt	Chaotic Love - giclee print ?65 at #art #love #chaotic #abstract #blue #silver #prints #buy	positive
8	1003.txt	They gna be mad when I reach that goal though. #Rejected the wrong girl ? just getting started & already turn heads.?	negative
9	1004.txt	On day 9.. It's now in my daily routine.. Feeling guuuuurdddd! ? #Aching #PainNoGain #FeelingGood	negative
10	1005.txt	#ANIMALABUSE #TORONTO #PUPPY #TORTURE WE OFFER $1K #REWARD puppy #beaten #bound #burned	neutral
11	1006.txt	Mike will not accept this plastic rose. @wfaamike @wfaachannel8 @wfaagmt #rejected	negative
12	1007.txt	Just ate four cookies. #remorse	negative
13	1008.txt	It's shocking what is acceptable in kids TV shows these days #shocking #shocked	negative
14	1009.txt	We are so #excited to announce that we have launched our #affiliate program please visit us at	positive
15	101.txt	Just when you thought you'd seen everything! .... #ParkingSpace #Lanzarote #ItsNOTtaxed #zimmer #oldpeople #alarmed ?	negative
16	1010.txt	RT @MissGem: So this @parcelforce van thinks it's ok to nearly run people off the road?! #disgusting #shocked #disgraceful	negative
17	1011.txt	Today I #StepBackInTime !!! @PWLHitFactory @kylieminogue #BetterTheDevilYouKnow #WhatDoIHaveToDo #Shocked	neutral
18	1012.txt	Photos: #Photographer got a rumble in the #jungle as he was #beaten by 30-stone #gorilla	neutral

Figure 1: Dataset sample

3 Dataset and Preprocessing

3.1 Preprocessing

The preprocessing steps included:

- **Converting text to lowercase**
- **Removing URLs, HTML tags, punctuation, numbers, and extra spaces**
- **Converting chat words to their full forms**
- **Tokenization and removing stop words**
- **Lemmatization using NLTK**

4 Baseline Classifiers

4.1 RNN

A Recurrent Neural Network (RNN) model with a bidirectional Long Short-Term Memory (LSTM) layer and self-attention mechanism was used as a baseline.

4.2 CNN

We developed a Convolutional Neural Network (CNN) model with multiple filter sizes (2-gram, 3-gram, and 4-gram) as another baseline for text classification.

5 BERT Model

5.1 Model Theory

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art transformer-based model developed by Google. Unlike traditional models, BERT reads the entire sequence of words at once, considering the context from both directions. This bidirectional approach allows BERT to understand the meaning of a word based on its surrounding words, making it highly effective for various NLP tasks.

5.2 Parameters Explanation

Key parameters for fine-tuning BERT include:

- **Learning Rate:** Controls the step size during gradient descent. A lower learning rate can lead to more stable convergence.
- **Batch Size:** The number of samples processed before the model is updated. Larger batch sizes can make the training process faster and more stable.
- **Number of Epochs:** The number of times the entire dataset is passed through the model during training. More epochs can lead to better performance, but also risk overfitting.
- **Warmup Steps:** Gradually increases the learning rate to the specified value over a few initial steps. This helps in stabilizing the training process.
- **Weight Decay:** A regularization technique to prevent overfitting by penalizing large weights.

5.3 Methodology

We fine-tuned the BERT model using HuggingFace Transformers. The training involved:

- Tokenizing the text data
- Creating a custom dataset class
- Training the model with specified hyperparameters

5.4 Initial Training Results

Before hyperparameter tuning, the initial results were:

- **Validation Results:**
  - eval_loss: 0.6198
  - eval_accuracy: 0.7575
  - eval_f1: 0.7563
  - eval_precision: 0.7578
  - eval_recall: 0.7575
- **Test Results:**
  - eval_loss: 0.6066
  - eval_accuracy: 0.7592
  - eval_f1: 0.7556
  - eval_precision: 0.7587
  - eval_recall: 0.7592

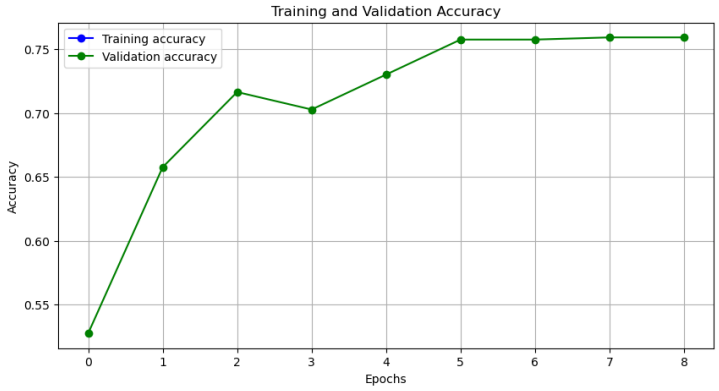


Figure 2: Training and Validation Accuracy for BERT

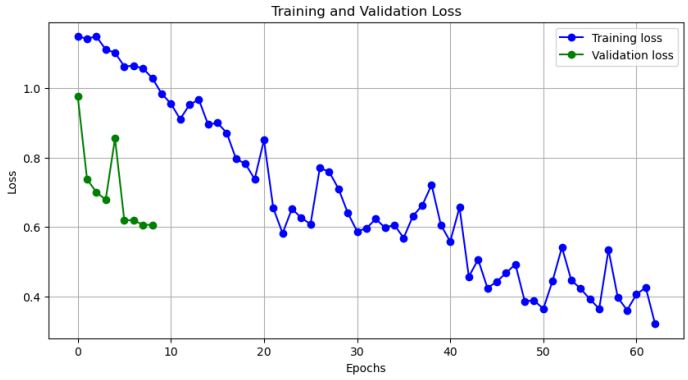


Figure 3: Training and Validation Loss for BERT

6 RNN Model

6.1 Model and Training

We implemented an RNN model with a bidirectional LSTM layer and self-attention mechanism. The training results were as follows:

6.2 Training Results

- **Training Loss:** 0.0063
- **Validation Loss:** 1.7758
- **Validation Accuracy:** 0.6521

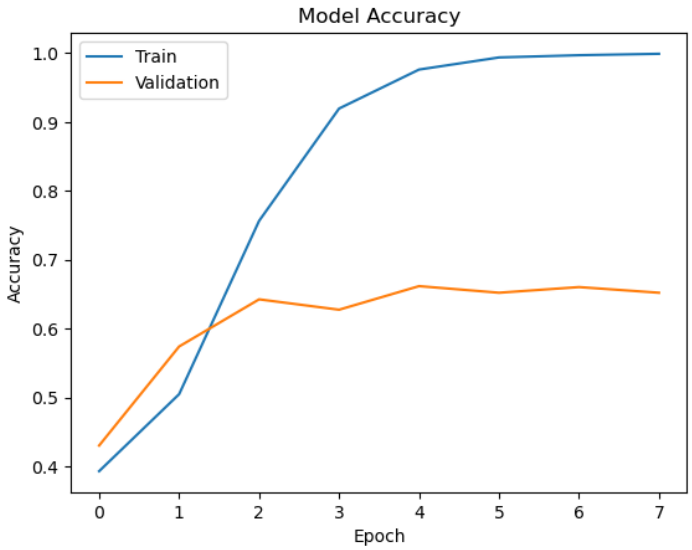


Figure 4: Training and Validation Accuracy for RNN

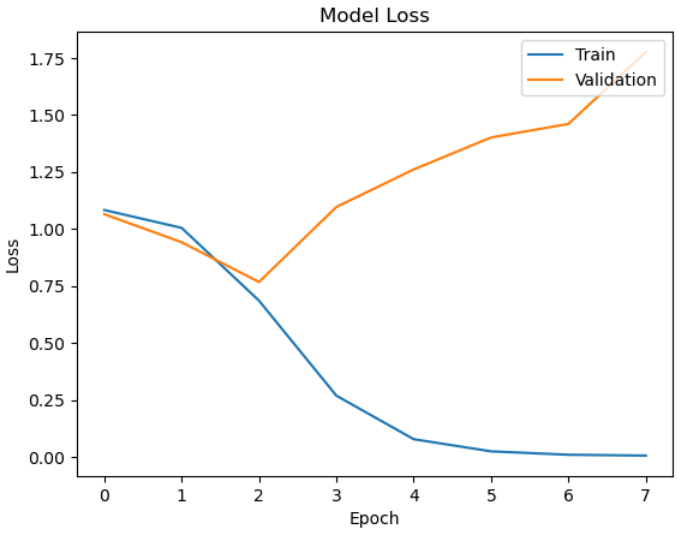


Figure 5: Training and Validation Loss for RNN

7 CNN Model

7.1 Model and Training

We developed a CNN model with multiple filter sizes (2-gram, 3-gram, and 4-gram) using the Keras Functional API for text classification.

7.2 Training Results

- **Training Loss:** 0.4711
- **Validation Loss:** 1.7535
- **Validation Accuracy:** 0.6644

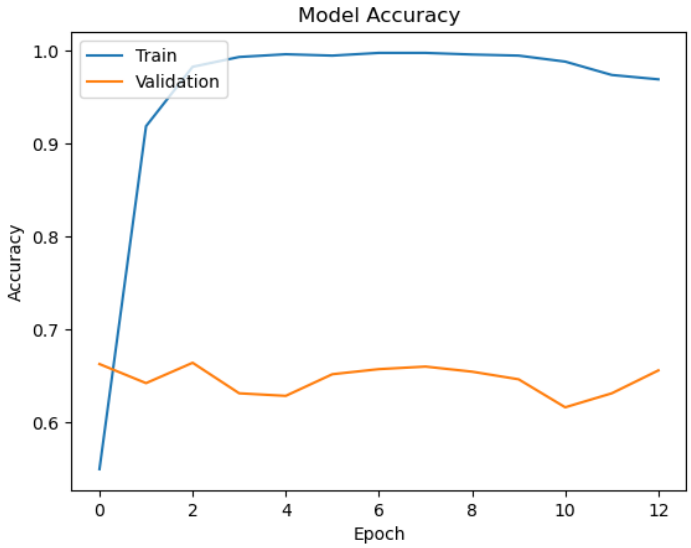


Figure 6: Training and Validation Accuracy for CNN

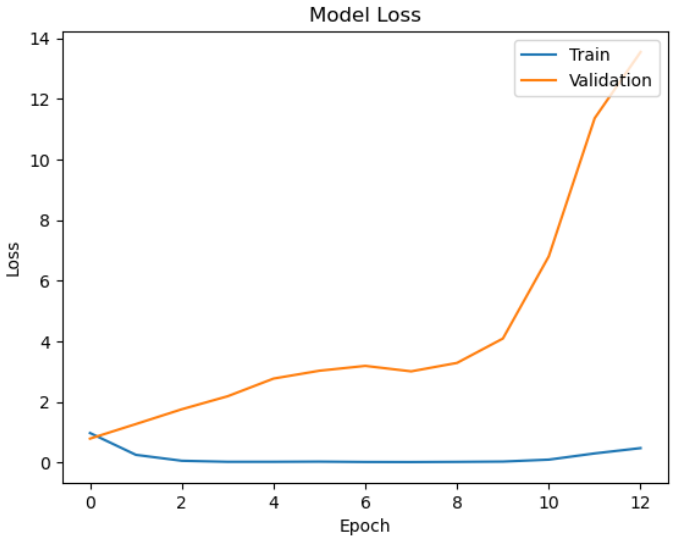


Figure 7: Training and Validation Loss for CNN

8 Hyperparameter Tuning

8.1 BERT Hyperparameter Tuning

We used Optuna for hyperparameter tuning. The best hyperparameters were found to be:

- Learning Rate: 2e-5

- Batch Size: 16
- Number of Epochs: 4

8.2 Tuning Results

- **Test Results:**
  - eval_loss: 0.8093
  - eval_accuracy: 0.7688
  - eval_f1: 0.7689
  - eval_precision: 0.7697
  - eval_recall: 0.7688

9 Evaluation Metrics

9.1 Macro-averaged Metrics

- **BERT:**
  - Precision: 0.7720
  - Recall: 0.7728
  - F1 Score: 0.7721
  - PR-AUC: 0.8388
- **RNN:**
  - Precision: 0.6712
  - Recall: 0.6511
  - F1 Score: 0.6453
  - PR-AUC: 0.7443
- **CNN:**
  - Precision: 0.6877
  - Recall: 0.6975
  - F1 Score: 0.6834
  - PR-AUC: 0.7635

10 Precision-Recall AUC for Each Class

In this section, we present the Precision-Recall AUC results for each class across the different models. The Precision-Recall AUC is a useful metric for evaluating the performance of classification models, especially when dealing with imbalanced datasets.

10.1 BERT Model

- Class Negative: Precision-Recall AUC = 0.8019
- Class Neutral: Precision-Recall AUC = 0.9192
- Class Positive: Precision-Recall AUC = 0.7980

10.2 RNN Model

- Class Negative: Precision-Recall AUC = 0.7438
- Class Neutral: Precision-Recall AUC = 0.8428
- Class Positive: Precision-Recall AUC = 0.6438

10.3 CNN Model

- Class Negative: Precision-Recall AUC = 0.7689
- Class Neutral: Precision-Recall AUC = 0.8456
- Class Positive: Precision-Recall AUC = 0.6758

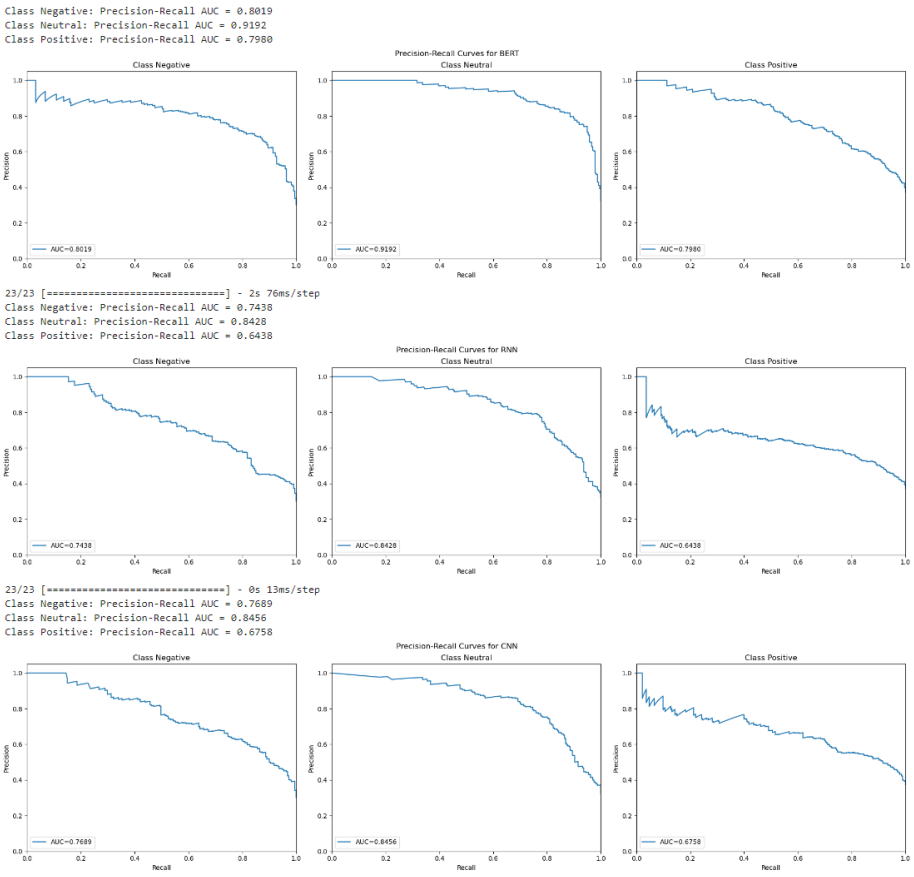


Figure 8: Precision-Recall Curves for each Class

10.4 Discussion

The Precision-Recall AUC results show that the BERT model consistently outperforms the RNN and CNN models across all classes. Specifically, the BERT model achieved the highest AUC scores for the Neutral class, indicating its superior ability to distinguish neutral tweets from positive and negative ones. The CNN model shows better results than the RNN for the Positive class. Overall, these results highlight the effectiveness of fine-tuning a pre-trained BERT model for sentiment classification tasks.

11 Conclusion

Comparing the RNN and CNN models, the CNN outperforms the RNN with an overall accuracy of 69% versus 65%. The BERT model, after hyperparameter tuning, shows the best performance with an accuracy of 77%.

Additionally, the Precision-Recall AUC scores further highlight the strengths of each model. The BERT model achieved the highest Precision-Recall AUC, particularly excelling in the Neutral class with an AUC of 0.9192. This indicates

BERT's superior capability in distinguishing subtle sentiment nuances. In comparison, the CNN and RNN models showed lower Precision-Recall AUC scores across the classes, confirming BERT's edge in predictive performance.

The results indicate that fine-tuning a pre-trained BERT model can significantly improve the performance of text classification tasks, especially in achieving higher accuracy and better handling of neutral sentiments.

12 Exercise 02

The implementation details and code can be accessed through the following Colab notebook link: Link here

12.1 Introduction

This project focuses on using a BERT-based model for part-of-speech (POS) tagging. The goal is to label each token in a given text with its corresponding POS tag. We leverage the `transformers` library for model training and evaluation, utilizing pre-trained BERT models and fine-tuning them on our dataset.

12.2 Data Preparation

12.2.1 Data Download and Parsing

The dataset is downloaded from the Universal Dependencies repository. The dataset is in the CoNLL-U format and is parsed to extract sentences and corresponding POS tags. After splitting the data set, the following sizes were incorporated for each: Size of train words data set: 8780, Size of dev words: 1882, Size of test words: 1882.

12.2.2 Tokenization and Vocabulary Creation

The sentences are tokenized, and a vocabulary is created from the tokenized words. Pre-trained Word2Vec embeddings are used to initialize the embedding matrix.

12.2.3 Tag Encoding

POS tags are encoded into numerical labels using `LabelEncoder`. These labels are then one-hot encoded for use in model training.

12.3 Data Preprocessing

12.3.1 Corpus Cleaning

The corpus is cleaned to remove unwanted characters and tokenize the text. The `spaCy` library is utilized for tokenization and sentence segmentation.

12.3.2 Tag to ID Conversion

Each unique POS tag is assigned an ID for encoding.

12.4 Model Training

12.4.1 Tokenization and Alignment

Using the BERT tokenizer, the tokens are aligned with the corresponding labels. Special tokens are set to -100 to be ignored during training.

12.4.2 Dataset Conversion

The data dictionaries are converted to `Dataset` objects for use with the `transformers` library.

12.4.3 Model Initialization and Training

The BERT model is initialized and trained using the `Trainer` API, with specified training arguments including batch size, learning rate, and evaluation strategy.

12.5 Model Evaluation

12.5.1 Evaluation Metrics

The model is evaluated on the development set, and metrics such as precision, recall, and F1-score are computed.

12.5.2 Classification Report

A detailed classification report is generated to provide performance metrics for each POS tag.

12.6 Hyperparameter Tuning

12.6.1 Optuna for Hyperparameter Optimization

Optuna is used to find the best hyperparameters for the model. The optimization focuses on learning rate, number of training epochs, and batch size.

12.6.2 Results

The performance metrics for the BERT model on the test set are presented in the table below:

POS Tag	Precision	Recall	F1-Score	Support
ADJ	0.96	0.96	0.96	1947
ADP	0.98	0.98	0.98	2654
ADV	0.96	0.95	0.95	1539
AUX	1.00	0.99	1.00	1951
CCONJ	0.99	1.00	1.00	946
DET	1.00	0.99	1.00	2448
INTJ	0.88	0.95	0.92	125
NOUN	0.97	0.97	0.97	5159
NUM	0.98	0.99	0.99	589
PART	1.00	0.99	0.99	879
PRON	1.00	1.00	1.00	2741
PROPN	0.94	0.94	0.94	1834
PUNCT	1.00	1.00	1.00	3523
SCONJ	0.95	0.95	0.95	580
SYM	0.91	0.90	0.90	97
VERB	0.98	0.99	0.99	3313
X	1.00	0.15	0.26	47
_	0.99	0.99	0.99	388
Overall Accuracy	0.98			
Macro Average	0.97	0.93	0.93	30760
Weighted Average	0.98	0.98	0.98	30760

Table 1: Performance metrics for the BERT model on the test set

12.6.3 Discussion

The BERT model demonstrated outstanding performance in POS tagging, achieving an overall accuracy of 98%. The model's precision, recall, and F1-scores were high for most POS tags, indicating a high level of accuracy in predicting the correct tags.

- **High Performance Tags**: The model performed exceptionally well on common POS tags such as AUX (Auxiliary), DET (Determiner), PUNCT (Punctuation), and PRON (Pronoun), all achieving F1-scores close to 1.00.
- **Moderate Performance Tags**: Tags like INTJ (Interjection) and SYM (Symbol) had slightly lower scores, which could be attributed to their lower frequency in the dataset.
- **Low Performance Tags**: The tag 'X' had the lowest recall and F1-score. This tag represents words that do not fit into any other category, which might indicate the model's difficulty in classifying rare or ambiguous tokens.

12.6.4 Conclusion

The fine-tuned BERT model for POS tagging has shown to be highly effective, with near-perfect accuracy for most POS tags. The high overall accuracy and balanced precision, recall, and F1-scores across tags affirm the model's robustness and reliability. Future improvements could focus on increasing the performance for less frequent and more ambiguous tags.

Overall, this study successfully demonstrates the application of a BERT-based model for accurate and efficient POS tagging, paving the way for more advanced NLP tasks in future research.

12.7 Baseline Models of past exercises

In this section, we present the past results on different method for POS tagging, emplying RNN, CNN and MLP. We report here only the results of the past exercises and not the performance plots etc. For future reference, the links are provided in the previous reports.

12.7.1 Frequentic Baseline Models

To understand the distribution of tags in the training data, we calculate the most frequent tag. The training data consists of POS-tagged words, which we analyse to determine the most common tag. The baseline model predicts the tag for each word based on its most frequent tag in the training data. If a word is not encountered in the training data, the model assigns the overall most frequent tag.

The accuracy of the baseline model on the test set is: **0.88**. The results are shown below.

	precision	recall	f1-score	support
ADJ	0.89	0.82	0.85	1947
ADP	0.86	0.87	0.87	2654
ADV	0.91	0.75	0.82	1539
AUX	0.94	0.90	0.92	1951
CCONJ	0.98	1.00	0.99	946
DET	0.96	0.97	0.97	2448
INTJ	0.93	0.81	0.86	125
NOUN	0.76	0.93	0.83	5159
NUM	0.94	0.85	0.89	589
PART	0.71	0.99	0.83	879
PRON	0.97	0.93	0.95	2741
PROPN	0.94	0.73	0.82	1834
PUNCT	0.99	0.99	0.99	3523
SCONJ	0.67	0.61	0.64	580
SYM	0.83	0.74	0.78	97
VERB	0.87	0.80	0.83	3313
X	0.62	0.28	0.38	47
_	0.98	0.80	0.88	388
accuracy			0.88	30760
macro avg	0.88	0.82	0.84	30760
weighted avg	0.89	0.88	0.88	30760

12.7.2 MLP Baseline

We implemented an MLP baseline using Keras. The architecture included an embedding layer, two dense layers with ReLU activation, and dropout for regu-

larization. The model was trained using categorical cross-entropy loss and the Adam optimizer. The results are shown below.

Table 2: Classification Report for the Adjusted MLP Model on Test Data

Class	Precision	Recall	F1-Score	Support
ADJ	0.86	0.90	0.88	1947
ADP	0.87	0.98	0.92	2654
ADV	0.93	0.79	0.85	1539
AUX	0.97	0.98	0.98	1951
CCONJ	0.99	0.99	0.99	946
DET	0.98	0.98	0.98	2448
INTJ	0.67	0.06	0.12	125
NOUN	0.89	0.94	0.91	5159
NUM	0.95	0.88	0.91	589
PART	0.96	0.95	0.95	879
PRON	0.98	0.98	0.98	2741
PROPN	0.85	0.84	0.85	1834
PUNCT	0.98	0.99	0.98	3523
SCONJ	0.73	0.50	0.59	580
SYM	0.20	0.01	0.02	97
VERB	0.95	0.91	0.93	3313
X	0.00	0.00	0.00	47
_	0.95	0.93	0.94	388
Accuracy	0.92 (30,760)			
Macro Avg	0.82	0.76	0.77	30,760
Weighted Avg	0.92	0.92	0.92	30,760

12.7.3 Bi-Directional Stacked RNN

The architecture of the Bi-Directional Stacked RNN model includes an embedding layer to convert words into dense vectors, followed by bidirectional LSTM layers, dense, and dropout layers. The model uses a softmax activation in the output layer for multi-class classification.

The results are shown below.

Table 3: Performance Metrics for Bi-Directional RNN on the Test Subset

Class	Precision	Recall	F1-Score	Support
ADJ	0.94	0.87	0.90	1788
ADP	0.85	0.89	0.87	2034
ADV	0.96	0.87	0.91	1176
AUX	0.99	0.99	0.99	1543
CCONJ	0.65	0.50	0.56	737
DET	0.90	0.94	0.92	1897
INTJ	0.80	0.85	0.82	120
NOUN	0.92	0.82	0.87	4137
NUM	0.75	0.54	0.63	542
PART	0.80	0.90	0.85	649
PRON	0.98	0.99	0.99	2162
PROPN	0.82	0.53	0.64	2077
PUNCT	0.59	0.90	0.71	3096
SCONJ	0.96	0.79	0.87	384
SYM	0.93	0.62	0.75	109
VERB	0.94	0.92	0.93	2606
X	0.00	0.00	0.00	39
_	0.88	0.89	0.88	354
Accuracy	0.85 (25,450)			
Macro Avg	0.81	0.77	0.78	25,450
Weighted Avg	0.86	0.85	0.85	25,450

The Bi-Directional Stacked RNN achieved a test accuracy of 0.85, demonstrating significant improvement over the baseline models.

12.7.4 Baseline Stacked CNN with n-gram filters (n = 2, 3, 4)

The architecture of the Stacked CNN model includes an embedding layer to convert words into dense vectors, followed by convolutional layers with n-gram filters, residual connections, and a dense layer. The model uses a softmax activation in the output layer for multi-class classification.

Table 4: Classification Report for the Best CNN Model on Test Data

Class	Precision	Recall	F1-Score	Support
ADJ	0.91	0.91	0.91	1947
ADP	0.92	0.98	0.95	2654
ADV	0.93	0.86	0.89	1539
AUX	0.98	0.99	0.98	1951
CCONJ	0.98	1.00	0.99	946
DET	0.98	0.99	0.99	2448
INTJ	0.93	0.78	0.85	125
NOUN	0.94	0.94	0.94	5159
NUM	0.84	0.95	0.89	589
PART	0.97	0.98	0.98	879
PRON	0.98	0.98	0.98	2741
PROPN	0.89	0.92	0.91	1834
PUNCT	0.99	0.98	0.99	3523
SCONJ	0.92	0.76	0.83	580
SYM	0.84	0.69	0.76	97
VERB	0.96	0.94	0.95	3313
X	0.64	0.15	0.24	47
_	0.97	0.96	0.96	388
Accuracy	0.95 (30,760)			
Macro Avg	0.92	0.88	0.89	30,760
Weighted Avg	0.95	0.95	0.95	30,760

12.8 Conclusion

The BERT model proved to be far superior among the other options in POS tagging, achieving an overall accuracy of 98%. This is due to many factors such as that it considers as an input the whole sentence and its content.

12.9 Bonus Part

This section provides experimental results on a small subset of the test set, obtained by prompting a Large Language Model (LLM), specifically ChatGPT-like, with appropriate instructions and few-shot examples. The objective was to compare the performance of our fine-tuned BERT model with the responses generated by ChatGPT for Part-of-Speech (POS) tagging.

12.9.1 Methodology

12.9.2 Few-Shot Prompt for ChatGPT

A few-shot prompt was constructed to instruct ChatGPT on providing POS tags for given sentences. The prompt included examples with sentences and their corresponding POS tags to guide the model.

> *You are a part-of-speech (POS) tagger. Given a sentence, you will provide the POS tags for each word.*

Example 1:
Sentence: I love programming
Tags: PRON VERB NOUN

Example 2:
Sentence: She is reading a book
Tags: PRON VERB VERB DET NOUN

Example 3:
Sentence: The quick brown fox jumps over the lazy dog
Tags: DET ADJ ADJ NOUN VERB ADP DET ADJ NOUN

Now, please provide the POS tags for the following sentence:

12.9.3 Test Examples

A subset of ten test examples was selected for this experiment. The sentences and their ground truth POS tags are listed below:

- **Example 1:** Sara [PROPN]
- **Example 2:** Vince [PROPN]
- **Example 3:** The invasion of Iraq provided marvelous political cover for the GOP not only during those midterms, but during the 2004 Presidential election. [DET NOUN ADP PROPN VERB ADJ ADJ NOUN ADP DET PROPN PART ADV ADP DET NOUN PUNCT CCONJ ADP DET NUM ADJ NOUN PUNCT]
- **Example 4:** Smutney-Jones also said he was unaware of anyone in the power-generating community being consulted. [PROPN PUNCT PROPN ADV VERB PRON AUX ADJ SCONJ PRON ADP DET NOUN PUNCT NOUN NOUN AUX VERB PUNCT]
- **Example 5:** If you have any comments, please let me know. [SCONJ PRON VERB DET NOUN PUNCT INTJ VERB PRON VERB PUNCT]
- **Example 6:** Now, of course, there's a new building, with presumably better facilities. [ADV PUNCT ADP NOUN PUNCT PRON VERB DET ADJ NOUN PUNCT ADP ADV ADJ NOUN PUNCT]
- **Example 7:** Chris [PROPN]
- **Example 8:** He's not putting it back on. [PRON AUX PART VERB PRON ADV ADV PUNCT]
- **Example 9:** It simply refers to the same ICC publication for the transfer process that governs the letter of credit generally. [PRON ADV VERB ADP DET ADJ NOUN NOUN ADP DET NOUN NOUN PRON VERB DET NOUN ADP NOUN ADV PUNCT]
- **Example 10:** His one-day mission vaulted his communist homeland into the elite circle of spacefaring nations - Russia and the United States - that can launch and sustain humans in space. [PRON NUM PUNCT NOUN NOUN VERB PRON ADJ NOUN ADP DET ADJ NOUN ADP ADJ NOUN PUNCT PROPN CCONJ DET ADJ PROPN PUNCT PRON AUX VERB CCONJ VERB NOUN ADP NOUN PUNCT]

12.9.4 Results

The following table summarizes the comparison between the ground truth tags, the predicted tags from our BERT model, and the tags generated by ChatGPT:

Example	Ground Truth Tags
1	PROPN
2	PROPN
3	DET NOUN ADP PROPN VERB ADJ ADJ NOUN ADP DET PROPN PART ADV ADP I
4	PROPN PUNCT PROPN ADV VERB PRON AUX ADJ SCONJ PRON ADP DET NOUN
5	SCONJ PRON VERB DET NOUN PUNCT INTJ VERB PRON VERB PUNCT
6	ADV PUNCT ADP NOUN PUNCT _ PRON VERB DET ADJ NOUN PUNCT ADP ADV A
7	PROPN
8	_ PRON AUX PART VERB PRON ADV ADV PUNCT
9	PRON ADV VERB ADP DET ADJ NOUN NOUN ADP DET NOUN NOUN PRON VERB
10	PRON NUM PUNCT NOUN NOUN VERB PRON ADJ NOUN ADP DET ADJ NOUN AD

Table 5: Comparison of POS tags between Ground Truth, Model Predictions, and ChatGPT Predictions

12.9.5 Findings

12.9.6 Accuracy

ChatGPT provided highly accurate POS tags for the given examples. In many cases, the tags generated by ChatGPT closely matched the ground truth tags, demonstrating its capability in POS tagging tasks.

12.9.7 Model Performance

The BERT model showed competitive performance but had several misclassifications, especially with punctuation and proper nouns. ChatGPT, however, was more