Sequence tagging in NLP

Ahmed Bahaa 49-0354

May 18, 2024

Abstract

This report presents the development and evaluation of a sequence tagging model for Part-of-Speech (POS) tagging. The motivation behind this research is to understand the current state of the field and identify potential areas for improvement. The literature review section includes a summary of two recent works in the field, highlighting their contributions,and the results of their work based on performance evaluation metrics such as ROUGE. The report aims to provide a comprehensive understanding of the current state of sequence tagging models in NLP and suggest potential directions for future research.

1 Introduction

Natural Language Processing (NLP) stands at the forefront of artificial intelligence, revolutionizing how machines comprehend and interact with human language. By bridging the gap between human communication and computational analysis, NLP enables computers to understand, interpret, and generate human language in a way that mimics human intelligence. NLP has witnessed unprecedented growth and innovation in recent years, fueled by advancements in deep learning, neural networks, and transformer models (Khurana et al., 2023). These breakthroughs have propelled NLP applications to new heights, empowering machines to perform tasks such as machine translation, sentiment analysis, text summarization, and question answering with remarkable precision. The ability of NLP systems to extract insights from unstructured text data has revolutionized industries ranging from healthcare and finance to marketing and customer service.

2 Motivation

The motivation for this project stems from the fundamental importance of Part-of-Speech (POS) tagging in Natural Language Processing (NLP). POS tagging, which involves categorizing words in a sentence into grammatical groups, is essential for various advanced NLP applications such as syntactic parsing, named entity recognition, machine translation, and information retrieval. Improving the accuracy of POS tagging enhances linguistic understanding and significantly boosts the performance of these downstream tasks. However, achieving high accuracy, especially in diverse and unseen contexts, remains a challenge. This project aims to address these challenges through a comprehensive approach that includes building a neural network model from scratch and leveraging pre-trained models via fine-tuning and transfer learning. These efforts will help in adapting POS tagging models to various domains and improving their robustness. Additionally, the project offers substantial educational value, providing hands-on experience in data analysis, model development, and fine-tuning techniques, while contributing to ongoing research in NLP. By addressing these motivations, this project not only aims to improve POS tagging performance but also advances the broader field of NLP technology and its practical applications.

3 Literature Review

In the following section, we will delve into recent advancements in the field of summarization models. We will explore a spectrum of approaches, from traditional extractive methods to cutting-edge abstractive techniques, employed in recent research endeavors. Through an in-depth examination of recent works, we aim to elucidate the state-of-the-art methodologies, innovations, and challenges in automatic text summarization.

Recent work in summarization has made significant progress due to introducing large-scale datasets such as the CNN-DailyMail dataset (Nallapati et al., 2016) and the New York Times dataset (Sandhaus, 2008). However, less work has focused on summarizing online conversations. Early approaches to conversation summarization consisted of feature engineering, template selection methods, and statistical machine learning approaches (Oya et al., 2014). More recent modeling approaches for dialogue summarization have attempted to take advantage of conversation structures found within the data through dialogue act classification, discourse labeling, topic segmentation, and key-point analysis (Ganesh and Dingliwal, 2019). However, such approaches focus exclusively on dialogue summarization, and it is not trivial to extend such methods to longer conversations with many more participants. The authors thus introduce a method to model the structure of the discourse over the many-party conversation.(Barker and Gaizauskas., 2016b) identify three key components of conversational dialogue: issues (that individuals discuss), viewpoints (that they hold about these issues), and assertions (that they make to support their viewpoints). they build on this framework and advances in argument mining for end-to-end training for summarization. Work in argument mining has aimed to identify these argumentative units and classify them into claims, premises, and major claims, or claims describing the key concept in a text.

The authors of the first paper aim to address this research gap by crowd-sourcing a suite of four datasets, which they called ConvoSumm, that can evaluate a model's performance on a broad spectrum of conversation data (Fabbri et al.,2021). For the news comments subdomain, they used the NYT Comments dataset, which consists of 2 million comments made on 9,000 New York Times articles published between 2017 and 2018. For the discussion forums and debate subdomain, they selected Reddit data from CoarseDiscourse which contains annotations about the discourse structure of the thread. For the community question answering subdomain, they used StackExchange (Stack), which provides access to all forums and has been used in modeling for answer relevance and question deduplication. For the email threads subdomain, they used the publicly available W3C corpus. They used BART-large as their base abstractive text summarization model. BART is a transformer encoder-decoder (seq2seq) model with a bidirectional (BERT-like) encoder and an autoregressive (GPT-like) decoder. BART is pre-trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text. BART is particularly effective when fine-tuned for text generation (e.g. summarization, translation) but also works well for comprehension tasks (e.g. text classification, question answering). They finetuned BART using a polynomial decay learning rate scheduler with Adam optimizer. They used a learning rate of 3e-5 and warmup and total updates of 20 and 200 (Fabbri et al.,2021).

Concerning the results, they trained BART on 200 examples from their validation set for abstractive models, using the remaining 50 as validation and test on the final test set of 250 examples. They evaluated the results using the scores of ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 Measures overlap of unigrams (single words) between the generated summary and reference summary. ROUGE-2 Measures overlap of bigrams (pairs of adjacent words) between the generated summary and reference summary. ROUGE-L Measures longest common subsequence between the generated summary and reference summary. For the first dataset (NYT), the ROUGE-1/2/L scores were 35.91/9.22/31.28 respectively. For the second dataset (Reddit), the scores were 35.50/10.64/32.57 respectively. For the third dataset (Stack), the scores were 39.61/10.98/35.35 respectively. Finally, for the last dataset (Email), the scores were 41.46/13.76/37.70. After that, the authors of the paper fine-tuned the Vanilla Bart model by using BART-arg, which is trained on argument-mining input. For the first dataset, The ROUGE-1/2/L scores after fine tuning were 36.60/9.83/32.61 respectively. For the second dataset, the scores were 36.39/11.38/33.57 respectively. For the third dataset, the scores were 39.73/11.17/35.52 respectively. Finally, for the last dataset, the scores were 40.32/12.97/36.90.

In summary, the results suggest that using argument-mining input improves the performance of the BART model in generating summaries. BART-arg achieves higher ROUGE scores for the first 3 datasets. For email data, however, this did not improve upon the BART baseline, likely due to the dataset's characteristics; email data is shorter and more linear, not benefiting from modeling the argument structure or removing non-argumentative units.

In the second work, the authors apply the attentional encoder-decoder for the task of abstractive summarization. They propose several novel models that address critical problems in summarization that are not adequately modeled by the basic architecture, such as modeling key-words, capturing the hierarchy of sentence-to word structure, and emitting words that are rare or unseen at training time (Nallapati et al.,2016). The model that they used is the Encoder-Decoder RNN. The encoder consists of a bidirectional GRU-RNN (Chung et al., 2014), while the decoder consists of a uni-directional GRU-RNN with the same hidden-state size as that of the encoder, and an attention mechanism over the source-hidden states and a soft-max layer over target vocabulary to generate words. In addition to the basic model, They also adapted to the summarization problem, the large vocabulary 'trick' (LVT) described in (Jean et al., 2014). In their approach, the decoder-vocabulary of each mini-batch is restricted to words in the source documents of that batch. In addition, the most frequent words in the target dictionary are added until the vocabulary reaches a fixed size. The aim of this technique is to reduce the size of the soft-max layer of the decoder which is the main computational bottleneck. Moreover, this technique also speeds up convergence by focusing the modeling effort only on the words that are essential to a given example. This technique is particularly well suited to summarization since a large proportion of the words in the summary come from the source document in any case. As discussed earlier, one of the key challenges is to identify the key concepts and key entities in the document, around which the story revolves. In order to accomplish this goal, the authors need to go beyond the word-embeddings-based representation of the input document and capture additional linguistic features such as parts-of-speech tags, named-entity tags, and TF and IDF statistics of the words. Therefore, they create additional look-up based embedding matrices for the vocabulary of each tag-type, similar to the embeddings for words. For continuous features such as TF and IDF, they convert them into categorical values by discretizing them into a fixed number of bins, and use one-hot representations to indicate the bin number they fall into. This allows us to map them into an embeddings matrix like any other tag-type. Finally, for each word in the source document, they simply look-up its embeddings from all of its associated tags and concatenate them into a single long vector.

The authors used the CNN/Daily Mail Corpus as their dataset. the authors used the human generated abstractive summary bullets from new-stories in CNN and Daily Mail websites as questions (with one of the entities hidden), and stories as the corresponding passages from which the system is expected to answer the fill-in-the-blank question. The authors used three main models in the experiments. The first one is words-lvt2k-1: This is the baseline attentional encoder-decoder model with the large vocabulary trick. This model is trained only on the first sentence from the source document. The second model is words-lvt2k-2-ptr. This model is identical to the model above except for the fact that it is trained on the first two sentences from the source. The last model is words-lvt2k-hieratt (Nallapati et al.,2016).

The performance of the three models on the CNN/Daily Mail test set is as follows: For the first model (words-lvt2k-1) the Rouge-1/ Rouge-2 / Rouge-L scores are 32.49 11.84 29.47 respectively. For the second model (words-lvt2k-ptr), the scores are 32.12 11.72 29.16 respectively. Finally, for the last model (words-lvt2k-hieratt), the scores are 31.78 11.56 28.73 respectively. These scores indicate the quality of the generated summaries compared to reference summaries. The words-lvt2k model achieved the highest scores in all three Rouge metrics.

To conclude, in this work the authors applied the off-the-shelf attentional encoder-decoder RNN that was originally developed for machine translation to summarization and show that it already outperforms state of-the-art systems on the CNN/Daily Mail dataset.

4 Data Analysis and Insights over the Dataset

I started my data analysis by creating a dictionary that contains all the vocabulary in my dataset. By doing so, I took a look at all the unique words in my dataset. Firstly, i started with some data analysis to gain some understanding of the nature of the dataset. I started with creating a method called "termFrequency" that returns a dictionary where each 'key' is a word in the dataset and each 'value' is the frequency of that word in the dataset.After that, I sorted the dictionary to get the top 10 most frequent words in my dataset, and then i visualized the results on a bar chart [figure 1]. The top most frequent word was "compare" with a count of 57516. This can provide insights about our data that most of the prompts focus on comparisons between different entities, concepts, ideas, or phenomena. Then, i created a method called "wordDocFreq" that calculates the document frequency of each word, which means how many prompts contain each word. Again, "Compare" has the highest document frequency. This again means that most of the documents contain questions in the form of comparison.

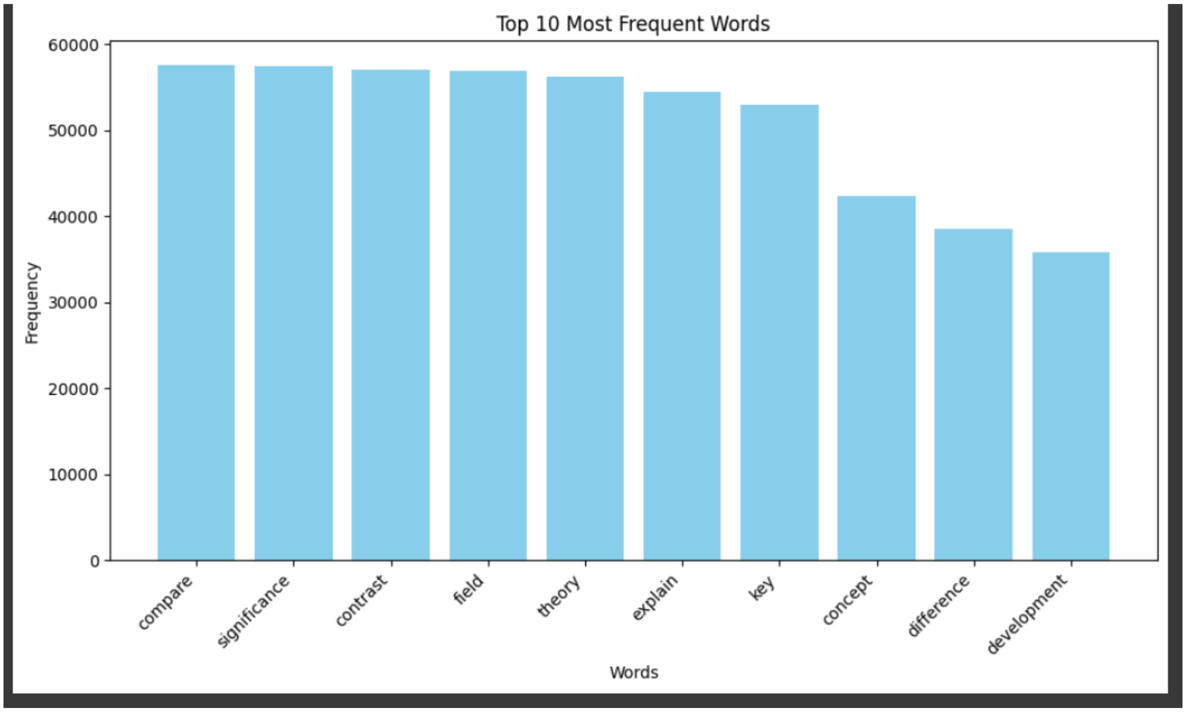


Figure 1: Most frequent words in the dataset.

Then i calculated the IDF score for each word in my dataset. The IDF score indicates how unique or important each word is relative to the entire corpus of documents. I created a dictionary where each key is the word and each value is the IDF score of that word. The results indicated that the word "connectionism" had the highest IDF score with a score of 11.73. The IDF score reflects how much information the word provides.The higher the IDF score, the more rare or unique the word is, and therefore, it's considered more important or significant in distinguishing between documents. Words with high IDF scores are often indicative of terms that are crucial for understanding specific topics or concepts in the dataset. "connectionism" may be a key concept within psychology or cognitive science. Figure 2 shows the top 10 words with the highest IDF scores.

Then i calculated the TF-IDF score for each word in my dataset. The TF-IDF score is a statistical measure used to evaluate the importance of a word in a document relative to a collection of document.

A high TF-IDF score for a word in a document indicates that the word is both frequent within that document and rare across the entire corpus, making it more likely to be relevant to the content of that document. I sorted the results and found that the word "theory" has the highest TF-IDF score of 85802. A high TF-IDF score suggests that the word "theory" is significant and representative of the content in the dataset. It indicates that "theory" appears frequently in some documents (high TF) but is relatively rare across the entire dataset (high IDF), making it a distinguishing term.

Moreover, the presence of "theory" as the word with the highest TF-IDF score may indicate that the dataset contains documents with a strong focus on theoretical concepts or discussions. It suggests that the topic or theme revolving around "theory" is central to the dataset.

Some of the limitations encountered was that many prompts had the same word repeated many

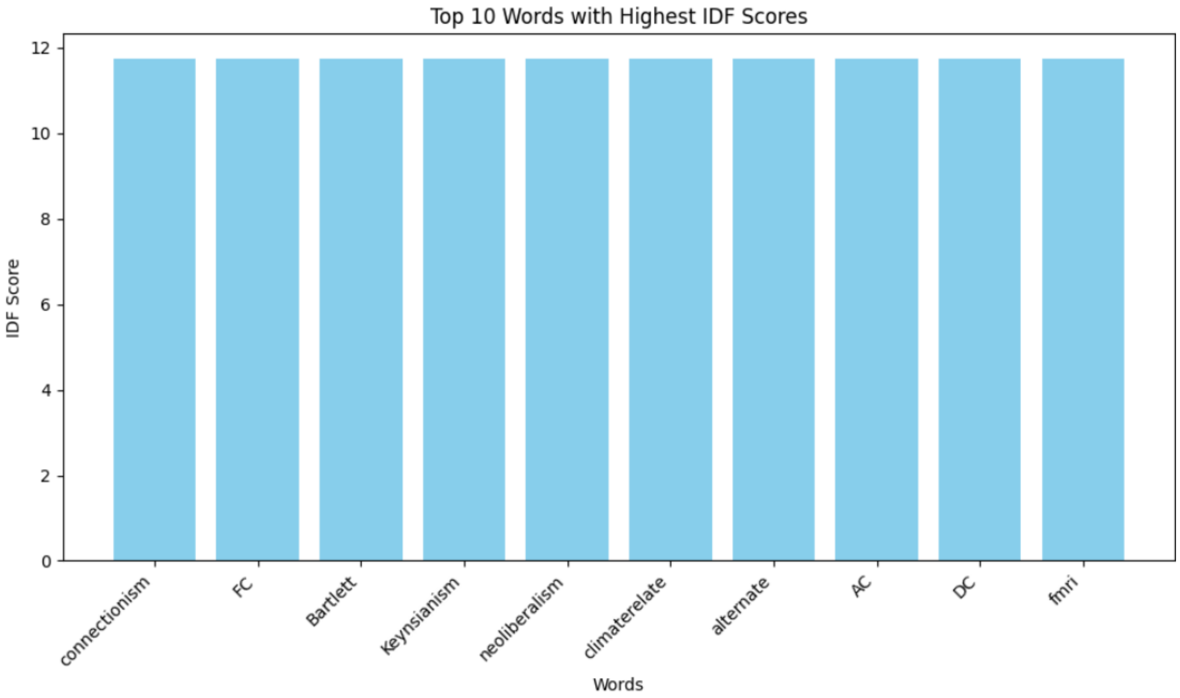


Figure 2: Words with the highest IDF score.

times which resulted in high and un-realistic prompt lengths. To tackle this issue, I dropped any row that has the same word repeated more than 4 times.

5 Methodology

5.1 Data preparation

To make sure that the data is ready for use, I checked if there were null values present in the data. Then, by inspecting the dataset, I discovered some rows that have the same word repeated many times. This needs to be cleaned by dropping these rows. I dropped any row that has the same word repeated more than 4 times.

Moreover, i did some preprocessing steps such as: tokenization, lemmatization, removing stop words, removing punctuation. I also created a column called ”pos results,” which contains the pos tag for each token in my dataset.

5.2 Neural Network architecture

The neural network model that i built consists of three layers. The first layer is the embedding layer. It is used to represent the input text data in a continuous vector space. The embedding layer takes 3 parameters. The first parameter is input dimensionality, which represents the length of my vocabulary. The second parameter is the output dimensionality, which represents the dimension of the dense embedding vectors. This defines the size of the vector space in which words will be embedded. The last one is the input length, which represents the length of input sequences (the number of words in each sequence). This parameter is needed to ensure consistent input size. i used 59 as the input length.

The second layer is the Bidirectional LSTM Layer. This layer is a Bidirectional Long Short-Term Memory (LSTM) network. It processes the sequence data in both forward and backward directions, which helps in capturing information from both past and future states for a given time step in the sequence. The Bi-LSTM layer takes as a parameter the number of LSTM units which determines the output dimensionality of the LSTM. A Bidirectional LSTM runs two LSTMs on the input sequence: one from the start to the end (forward LSTM) and another from the end to the start (backward

LSTM). The outputs of these two LSTMs are concatenated at each time step, providing the model with information from both past and future contexts for each point in the sequence.

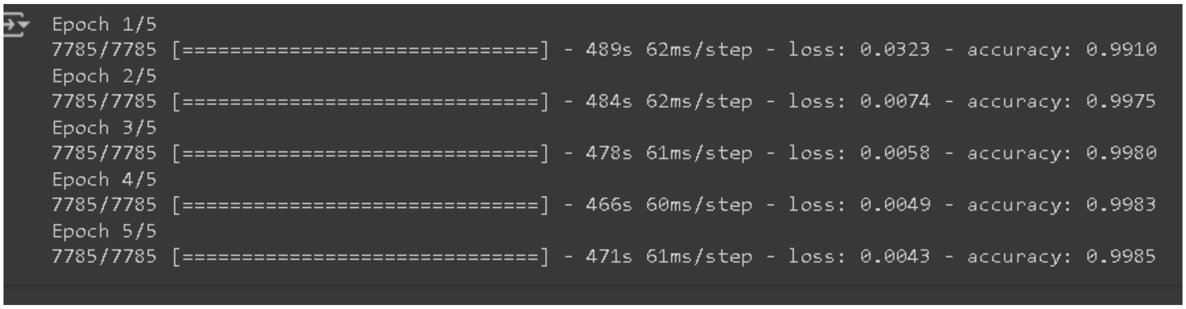
The next layer is the dense layer. This layer is a fully connected (dense) layer that performs classification. It takes the output from the LSTM layer and produces a probability distribution over the target classes (tags) for each time step in the sequence. The first parameter that this layer takes is the number of tags which is 17 in my dataset. The next parameter is the activation function which is softmax. The softmax activation function converts the raw output scores (logits) from the Dense layer into probabilities. It ensures that the sum of the probabilities across all classes is 1. This makes it easier to interpret the output as a probability distribution over the classes.

Finally, i compiled the model using "Adam" optimizer, which is an adaptive learning rate optimization algorithm that's been proven effective in practice. For the loss function, i used The "sparse categorical crossentropy" loss function. It is used for multi-class classification tasks where the target labels are integers.

This architecture is particularly effective for POS tagging due to its ability to capture and utilize contextual information from both past and future words in a sequence, which is crucial for accurately determining the part of speech of each word in a sentence. For example, the architecture provides contextual understanding. POS tagging requires understanding the context in which a word appears, as the same word can have different parts of speech depending on its usage. The Bidirectional LSTM layer allows the model to consider both the preceding and succeeding words in a sequence, enabling a more comprehensive understanding of context. This is essential because the meaning and role of a word can depend on its neighbors in the sentence. Furthermore, the architecture provides sequential data handling. The use of LSTM units helps in managing sequential data effectively. LSTMs are designed to remember long-term dependencies, which means they can keep track of the entire context of a sentence, even if the relevant information is spread out over many words. This capability is important in POS tagging, where a word's part of speech can be influenced by distant words. Moreoever, the architecture provides smooth integration from Word to Tag as it transitions smoothly from raw text to the final POS tags. The Embedding layer converts words into dense vector representations that capture semantic information. This representation is then processed by the Bidirectional LSTM to incorporate contextual dependencies. Finally, the Dense layer with softmax activation classifies each word into its respective part of speech

5.3 Training and evaluation results

Before training the model, I identified my X-Train and y-train. My x-train is the integer representation of tokens (words) of my dataset, while the y-train is the tags. I created a dictionary in which i mapped each tag to an index. I made sure that x-train and y-train have the same length by padding both of them to the max-length which is 59 in my case. I trained the model for 5 epochs using a batch size of 32. The loss and accuracy for each epoch are shown in figure 3.



```
Epoch 1/5
7785/7785 [==============================] - 489s 62ms/step - loss: 0.0323 - accuracy: 0.9910
Epoch 2/5
7785/7785 [==============================] - 484s 62ms/step - loss: 0.0074 - accuracy: 0.9975
Epoch 3/5
7785/7785 [==============================] - 478s 61ms/step - loss: 0.0058 - accuracy: 0.9980
Epoch 4/5
7785/7785 [==============================] - 466s 60ms/step - loss: 0.0049 - accuracy: 0.9983
Epoch 5/5
7785/7785 [==============================] - 471s 61ms/step - loss: 0.0043 - accuracy: 0.9985
```

Figure 3: Training results.

Then, i evaluated the model on my test data and the results yielded an accuracy of 0.9986. Finally, i compared the results of 10 predicted tags with the actual tags, and the model successfully predicted all the tags correct.

5.4 BERT

After building the previous neural network model without using any pretrained model, I used BERT as a pre-trained model and applied fine-tuning. The model that i used is BERT- Tiny as it effective and achieves more than 96.8% of the performance of its teacher BERTBASE (Jiao et al.,2019) . I started with defining the model name and loading the tokenizer that will be used.

5.4.1 Architecture

I started with defining the hyperparameters that were used in the model. I chose a batch size of 128, 3 epochs, and a learning rate of 7e-5. I chose 128 because large batch sizes can take better advantage of parallel processing on GPUs, potentially speeding up training per epoch. For the learning rate, i trained the model first on 5e-5, then 3e-5, and finally 7e-5. I setteled on 7e-5 as it provided the lowest loss per epoch while training and the highest accuracy (86%)when evaluating the model on the test set. After that, i converted tokenized input sequences and their corresponding POS tags into PyTorch tensors. Then, i created a tensor dataset by combining the input IDs and labels. The next step is initializing the data loader object which provides an efficient way to iterate over the dataset in batches, with shuffling for randomness. After that, i initialized a pre-trained TinyBERT model for token classification, specifying the number of POS tags. I used Adam optimizer to update model parameters during training.

The training loop started with batch processing. Within each epoch, the dataset is processed in batches to manage memory efficiently and to enable faster convergence. Before performing backpropagation, the gradients of all model parameters are reset to zero. The next step is forward pass. The input batch is passed through the model to compute the output and loss. Next comes backward pass. Gradients are computed through backpropagation based on the loss. Then the optimizer updates the model parameters based on the computed gradients. The training loop is essential for fine-tuning the TinyBERT model on the POS tagging task. It involves multiple epochs, each consisting of batch-wise processing of the dataset. Key steps include zeroing gradients, performing forward and backward passes, updating parameters, and monitoring progress. This iterative process adjusts the model parameters to minimize the loss function, improving the model's accuracy in predicting POS tags.

5.4.2 BERT Evaluation results

During evaluation, the model is tested on unseen data to measure its ability to generalize and predict POS tags correctly. The model has a loss of 0.0162 and an accuracy of 0.8588. I tested the model on unseen text data and the model correctly predicted the tags for each text. I used the text ”Natural Language Processing” and each word got assigned the right POS tag.

5.5 Discussion and findings

The neural network model that i built without using any pretrained model was more effective than the BERT Tiny model. It has an accuracy of 0.998 compared to BERT which has an accuracy of 0.855. Moreover, the first model was tested on 10 prompts from the test data and the predictions were compared with the actual POS tags. The model predicted all the tags correct.

References

[1] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, C¸ aglar Gulc¸ehre, and Bing Xiang. 2016. ˘ Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 280–290, Berlin, Germany. Association for Computational Linguistics

[2] Evan Sandhaus. 2008. The new york times annotated corpus. Linguistic Data Consortium, Philadelphia, 6(12):e26752

[3] Fabbri, A. R., Rahman, F., Rizvi, I., Wang, B., Li, H., Mehdad, Y., & Radev, D. (2021). ConvoSumm: Conversation summarization benchmark and improved abstractive summarization with argument mining. arXiv preprint arXiv:2106.00829.

[4] Nallapati, R., Zhou, B., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.

[5] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. Multimedia tools and applications, 82(3), 3713-3744.

[6] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. Journal of Management Analytics, 7(2), 139-172.

[7] Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In Proceedings of the 8th International Natural Language Generation Conference (INLG), pages 45–53, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

[8] Prakhar Ganesh and Saket Dingliwal. 2019. Abstractive summarization of spoken and written conversation. CoRR, abs/1902.01615.

[9] Emma Barker and Robert Gaizauskas. 2016b. Summarizing multi-party argumentative conversations in reader comment on news. In Proceedings of the Third Workshop on Argument Mining (ArgMining2016), pages 12–20, Berlin, Germany. Association for Computational Linguistics

[10] Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., ... & Liu, Q. (2019). Tinybert: Distilling bert for natural language understanding. arXiv preprint arXiv:1909.10351.