Natural language processing and pattern recognition methods

Implementing NLP Solutions Using Transformers and Attention Mechanisms

Bauyrzhan Zhakenov

23MD0314

Date: May 4, 2025

1. Introduction

1.1 Objective of the Project

Attention mechanism in Natural Language Processing (NLP) firstly used on Aligned Machine Translation task. Then in 2017 this mechanism was used in Transformer networks [1].

The main problem solved by Attention and Transformers is that RNN suffers from long-range dependencies in the data and old approaches often loose details for long sentences. Attention by using attention blocks helped to save context between any part os the sequence.

During Text Generation task, before generate new word model looks back to all states of input. During Sentiment Analysis task model learns to weigh each word of the sequence how much it influence positivity or negativity. During Named Entity Recognition task previous techniques failed to extract connection between far apart words, but with attention mechanism there is no difficulty with it. So, Attention helps focus model on the right parts of the input.

The project centers on the Attention mechanism and Transformer networks. It begins by outlining the limitations of RNNs and why we need Transformers, then explains how the self-attention mechanism lets tokens attend to one another. Next, it gives a brief overview of major pretrained models before showing how BERT can be used for Named Entity Recognition and how to fine-tune it the annotated NER dataset. Finally, it demonstrates sentiment analysis with RoBERTa.

2. Limitations of RNNs and the Need for Transformers

2.1 RNNs vs. Transformers

The main problem solved by Attention and Transformers is that RNN suffers from long-range dependencies in the data and old approaches often loose details for long sentences. Attention by using attention blocks helped to save context between any part os the sequence.

RNNs process inputs one step at a time, which prevents parallelization and makes training slow. RNN encodes the sequence to a single fixed-length vector, causing loss of information for long sentences.They struggle to retain context over long text, often focusing only on short-term dependencies. Transformers overcome these issues by doing away with recurrence entirely. They process all tokens using self-attention, enabling parallel computation and much faster training on GPU. With the self-attention mechanism, Transformers learn relationships between all words in a sequence regardless of their positions. The main advantage is the ability to extract meaning of the context through attention, instead of compressing all information into one hidden state

2.2 Transformer Model

The original Transformer is composed of an encoder stack and a decoder stack [1]. The encoder takes an input sequence and produces a high-level representation, and the decoder generates an output sequence using the encoder output and its own self-attention. Input tokens are first converted to vector embeddings. Transformers rely on positional encoding added to these embeddings to inject information about token positions in the sequence. At the core of each transformer layer is the self-attention mechanism. Each layer computes attention by creating Query, Key and Value vectors for each token and finding how each token relates to others. Each transformer block has a position-wise feedforward neural network. This is a simple two-layer fully connected network applied to the representation of each token to further transform the features. Transformers use skip connections around sub-layers and apply layer normalization. These residual connections help gradients flow effectively in deep networks, mitigating vanishing-gradient issues [2].

2.3 Results

We trained both an LSTM and a Transformer model on the IMDB dataset for text classification. After 3 epochs, the Transformer outperformed the LSTM, achieving an accuracy and F1-score of 85, compared to 80 for the LSTM.

2.4 Figures

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.83	0.81	12500
1	0.82	0.77	0.80	12500
accuracy			0.80	25000
macro avg	0.80	0.80	0.80	25000
weighted avg	0.80	0.80	0.80	25000

Figure 1: RNN Classification Report

Classification Report:

	precision	recall	f1-score	support
0	0.85	0.86	0.86	12500
1	0.86	0.84	0.85	12500
accuracy			0.85	25000
macro avg	0.85	0.85	0.85	25000
weighted avg	0.85	0.85	0.85	25000

Figure 2: Transformer Classification Report

3. Self-Attention Mechanism

3.1 Self-Attention

At the core of each transformer layer is the self-attention mechanism. Each layer computes attention by creating Query, Key and Value vectors for each token and finding how each token relates to others. Query - what information token is looking for, Key - content of other tokens, and Value - actual information of those tokens. The self-attention module compares each Query with all Keys to measure similarity. These similarities become attention weights. High weight means the model finds a token highly relevant to another. The output for each token is computed as a weighted sum of all the Value vectors, using the attention weights as coefficients. [3]

3.2 Visualizations

The heatmap shows that repeated tokens (“Captain!”) shows their highest attention to each other, while all other tokens, appearing only once, focus most strongly on themselves.Example confirms that self-attention inherently captures token identity: identical words reinforce each other’s representations, and unique words default to self-reinforcement.

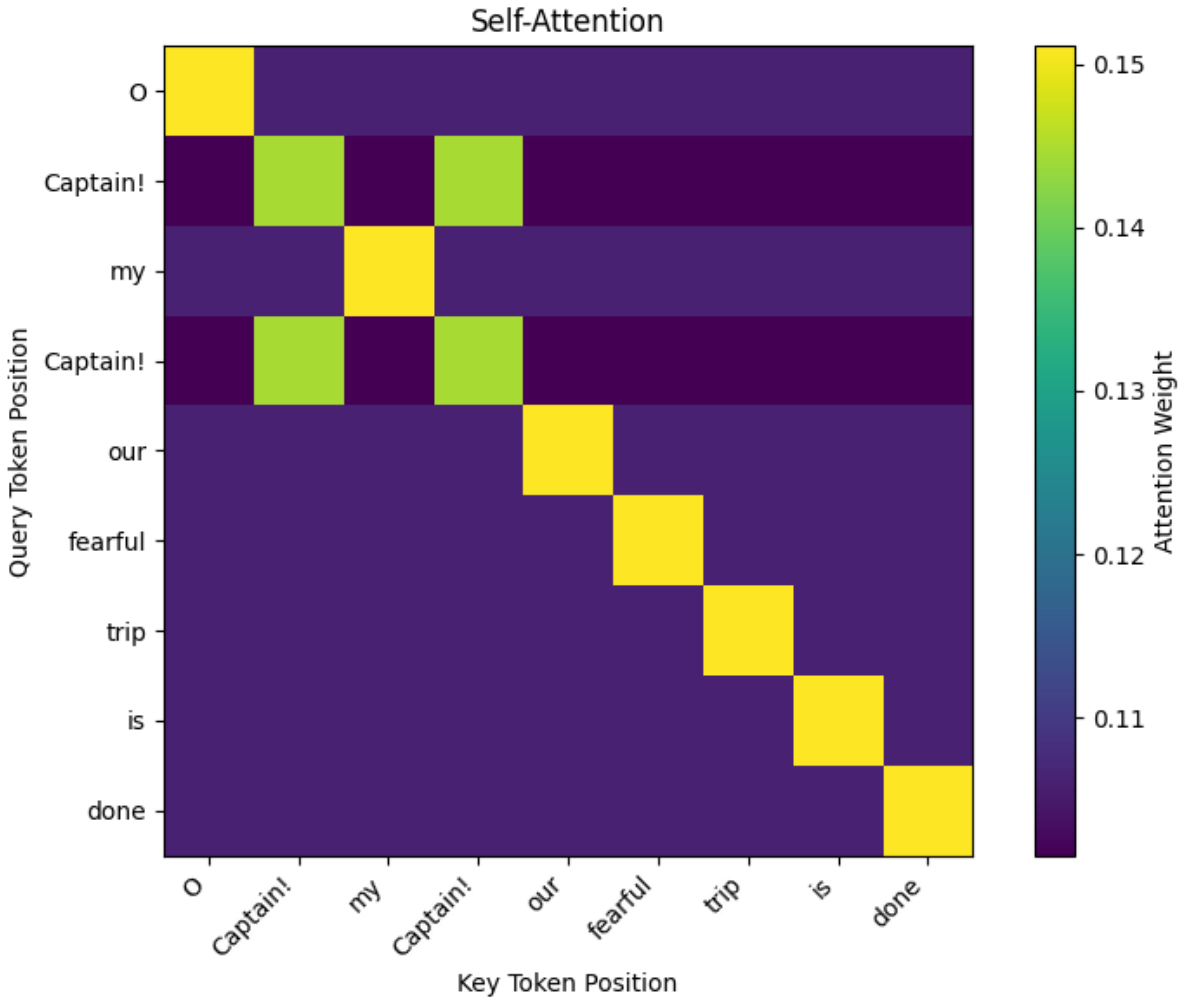

Figure 3: Self Attention

4. Pretrained Language Models (BERT, GPT, T5)

4.1 Fine-tuning and Results

Fine-tuning is a type of transfer learning, when we continue to train a pre-trained model for the new task [4]. Fine-tuning allows the model to adapt to better fit the specifics of the target task for higher performance. Fine-tuning generally requires only a few new parameters, since the bulk of the model is reused. This is why models like BERT are so valuable – they can be fine-tuned multiple times for different tasks with small effort.

Bidirectional Encoder Representations from Transformers(BERT) is is an encoder-only model pretrained on masked language modeling and next-sentence prediction [5]. To fine-tune BERT for a task like NER or text classification, a small layer is added on top. Then whole model is trained on the task data.

Generative Pre-trained Transformer(GPT) models are decoder-only language models. After pretraining on next-word prediction, GPT can be fine-tuned on specific generative tasks. GPT-2

can be fine-tuned to generate domain-specific text by continuing training on a new data. During fine-tuning, we may input prompts or prefixes and train the model to produce desired outputs.

Text-to-Text Transfer Transformer(T5) is an encoder–decoder model that treats every NLP task as a text-to-text problem. Fine-tuning T5 involves supplying inputs with a task-specific prefix and training the model to generate the appropriate output text [6].

4.2 Generated Text

In our script, both LSTM and GRU models are evaluated based on training and validation accuracy and loss. As usually, the training process for both models shows a progressive increase in accuracy and a corresponding decrease in loss over epochs. But in complex tasks the GRU model might display faster convergence due to its simpler architecture. Validation curves often reveal similar performance, with slight differences depending on the dataset and specific hyperparameter settings. While LSTMs may achieve marginally higher accuracy on some tasks due to their more complex gating mechanisms, GRUs offer a compelling balance of efficiency and effectiveness, making them a popular choice when computational resources are limited.

4.3 Evaluation

GPT-2 on Shakespeare dataset shows Perplexity=12 and Accuracy around 4. Text generator shows not so high results and weak—high perplexity with token mismatch show it's not capturing original.

BERT demonstrates robust performance in news-topic classification, achieving high precision and recall across all four categories. Its F1-scores, all exceeding 0.91, reflect accurate predictions.

A T5's translation task's BLEU score below 6 indicates minimal overlap between the model's output and reference translations. Low score suggests that T5, as configured, fails to produce fluent or semantically faithful French translations, and would require substantial retraining or fine-tuning to reach acceptable quality levels.

4.4 Figures

Classification Report:

	precision	recall	f1-score	support
World	0.9619	0.9558	0.9588	1900
Sports	0.9879	0.9879	0.9879	1900
Business	0.9262	0.9111	0.9185	1900
Sci/Tech	0.9156	0.9363	0.9258	1900
accuracy			0.9478	7600
macro avg	0.9479	0.9478	0.9478	7600
weighted avg	0.9479	0.9478	0.9478	7600

Figure 4: BERT Classification Task

```
ROMEO:     I am dead, and a new sun
      Hath dimmed my glory.

APEMANTUS.
What, is there no mourning,
That is more lamenting than weeping?

LEPIDUS.
There's none, it seems;
For mourners love that their sorrow shows.
I am, I am, I am, I am. Thou hast done
Some good to me, if I may call it so.

APEMANTUS.
The gods defend thee, I take it.

LEPIDUS.
'Tis most unlike
That thou hast given me counsel and counsel-contemplation.
This is most unlike. I know what you say, Patroclus,
Thou wert truly dead.

APEMANTUS.
Why, thou believ'st it.

LEPIDUS.
Then I
```

Figure 5: GPT Shakespeare's Text Generation

5. Named Entity Recognition (NER)

5.1 NER with spaCy and Transformers

spaCy is a free open source library for advanced NLP tasks in Python. It can be used to build information extraction or natural language understanding systems [7]. The transformer-based NER approach generally achieves higher accuracy and F1 score than the spaCy default models. Pretrained transformers understand the language context deeply, leading to better recognition of entities.

5.2 Comparison

A transformer-based NER model achieved an F1 score of 0.7944, demonstrating strong precision and recall in identifying named entities.

In contrast, spaCy's NER pipeline yielded an F1 score of just 0.0634, indicating a failure to correctly recognize the majority of entities.

Such big difference in F1 shows the necessity of transformer architectures for high-quality named entity recognition and low-level approach to use spacy's NER pipeline.

5.3 Figures

Transformer NER classification report:

	precision	recall	f1-score	support
LOC	0.7879	0.8525	0.8189	4657
ORG	0.6911	0.7252	0.7077	4745
PER	0.8813	0.8439	0.8622	4556
micro avg	0.7827	0.8064	0.7944	13958
macro avg	0.7867	0.8072	0.7963	13958
weighted avg	0.7855	0.8064	0.7952	13958

Figure 6: Transformer for NER

6. Training Custom NER Models

6.1 NER Dataset

The CoNLL-2003 dataset is a classic benchmark for named entity recognition, introduced in the CoNLL-2003 shared task [8]. It contains annotated text for two languages - English and German.

6.2 Model Evaluation

BERT fine-tuned on CoNLL-2003 achieves uniformly high entity recognition performance, with F1-scores of 0.96 for LOC, 0.88 for MISC, 0.92 for ORG, and 0.96 for PER. Micro-averaged F1 of 0.94 reflects consistently accurate token-level predictions across all 11,136 test entities. Macro-averaged F1 of 0.93 confirms balanced performance even on smaller classes, demonstrating BERT's robust generalization for named-entity recognition.

6.3 Figures

```
— Full Classification Report —
              precision    recall  f1-score   support

         LOC       0.97      0.96      0.96      3635
        MISC       0.87      0.89      0.88      1480
         ORG       0.91      0.93      0.92      2702
         PER       0.96      0.97      0.96      3319

   micro avg       0.94      0.95      0.94     11136
   macro avg       0.93      0.94      0.93     11136
weighted avg       0.94      0.95      0.94     11136
```

Figure 7: BERT on CoNLL-2003

7. Sentiment Analysis

7.1 Sentiment Analysis with BERT

The LSTM completes training in just 14 seconds but only achieves a Test Loss of 0.3640 and 86.80 accuracy, reflecting its limits of context understanding. Fine-tuning BERT required 816 seconds but delivered lower validation loss of 0.2730 and high accuracy of 92.32. For IMDb sentiment analysis, BERT's needs more in time and compute is clearly justified by its better performance.

7.2 Comparison with LSTM

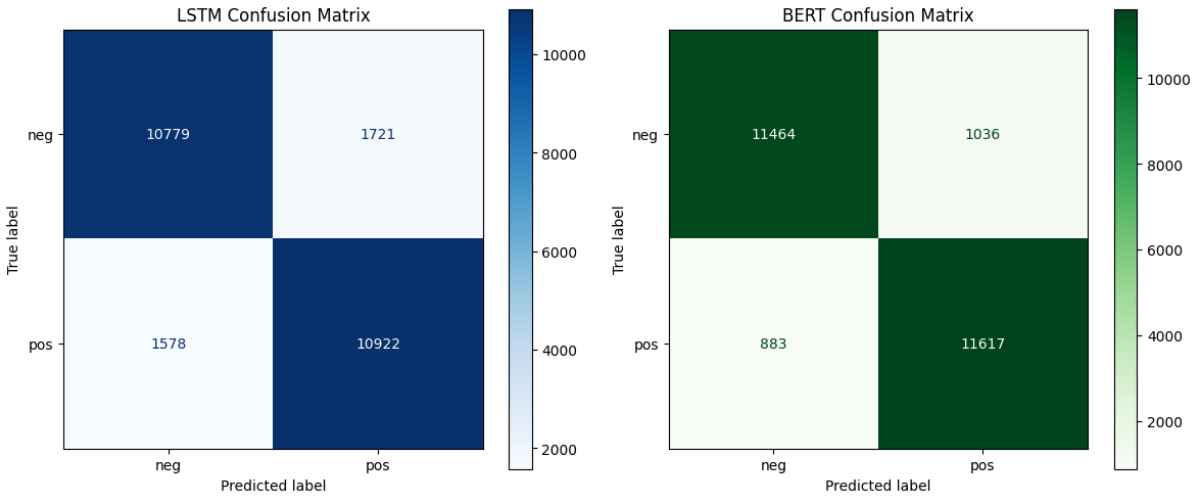

Figure 8: Confusion Matrix: BERT vs LSTM

8. Conclusion

8.1 Summary

Across all the experiments transformer models bring the accuracy and language finesse that simpler methods cannot match. RNN is lightweight approach which fast and easy to use but often fall short when you need high precision. Fine-tuned transformers often deliver reliable results across tasks, showing that adapting to the specific data is on high level.At the same time this capability requires significant computing resources and large amounts of data. In real applications, companies must find balance between model strength and resource limits.

8.2 Future Work

Building on GPT-2's high perplexity and low token accuracy, future work should include Shakespeare-specific fine-tuning on larger dataset and prompt engineering to improve generation quality. To address BERT computational intensity, model distillation and pruning techniques should be applied to save or maybe improve classification accuracy while reducing resource demands. For T5 translation targeted English–French sentence pairs and employing back-translation loops will help raise its BLEU score. Enhancing spaCy's NER with lightweight adapter modules can help for better accuracy.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available:

https://arxiv.org/abs/1706.03762

[2] Voita, "Nlp course by lena voita." [Online]. Available: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

[3] V. Jain and A. Chadha, "Distilled notes for stanford cs224n: Natural language processing with deep learning," https://www.aman.ai, 2020, accessed: 2020-07-01. [Online]. Available: www.aman.ai

[4] F. Mubarak, "Transfer learning and fine tuning." [Online]. Available: https://medium.com/munchy-bytes/transfer-learning-and-fine-tuning-363b3f33655d

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[6] T. Overview, "Transfer learning and fine tuning." [Online]. Available: https://medium.com/@sharathhebbar24/t5-overview-05327690fa93

[7] "spacy documentation." [Online]. Available: https://spacy.io/

[8] E. F. T. K. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," 2003. [Online]. Available: https://arxiv.org/abs/cs/0306050