Transformer-based Architecture Neural Network Approach to Email Message Autocomplete

Mateus Fernando Felismino da Silva Patriota
Advisor: Prof. Dr. Tiago Figueiredo Vieira
Co-advisor: Prof. Dr. Baldoino Fonseca dos Santos Neto

Institute of Computing
Universidade Federal de Alagoas

Undergraduate Final Project, November 2023

Table of Contents

Table of Contents

Motivation

Studying the area of neural network is of great importance, especially for applications in everyday life and for learning how to improve nowadays algorithms.

- **Recurrent Neural Networks** (RNNs), including Long Short-Term Memory (LSTM) networks have firmly established themselves as state-of-the-art [1];
- In Recent years, the **Transformer** architecture has emerged as a groundbreaking alternative [2];
- Self **attention mechanism** [2];
- Investigate is the **fine-tuning process** of the Transformer-based architecture [3].

Objectives

General Objectives: This project involves the creation of a neural network architecture based on Transformers, **built from the ground up**, with a particular emphasis on improving **email message autocomplete** functionality.

Specific Objectives:

- To design a transformer-based neural network architecture that is able to learn long-range **dependencies in email messages**;
- To pre train and evaluate the proposed architecture on a large dataset of open web texts;
- To fine-tune the proposed architecture on a smaller dataset of user-specific email messages;
- To investigate the effects of different **hyperparameters** on the performance of the fine tuned model.

Table of Contents

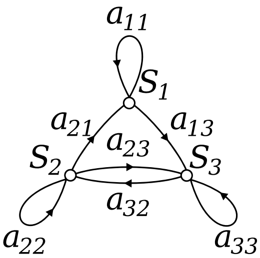
NLP and Machine Learning Approach

- Email auto-completion utilizes AI algorithms, particularly NLP and ML, to predict and suggest sentence completions [4];
- NLP's ability to **understand** and **generate** human language is crucial for efficient auto-complete technology;
- ML algorithms can be trained on extensive text datasets to learn writing styles, common phrases, and patterns;
- The adaptability of AI and its algorithms to user **context, language, and patterns** holds significant potential for the evolution of auto-complete email technology.

Markov Chains

- Markov Chains are probabilistic models that capture the **likelihood** of transitioning from one state to another in a sequence [5].
- May not exhibit the same semantic understanding as NLP-based approaches [6].

Figure: Simple Markov chain graph representation.



Source: Ching (2016) [7].

Generative Pre-trained Transformer 2

- Was introduced in **2019** from OpenAI [8], is a substantial advancement in natural language processing;
- The core idea of **GPT-2** is to predict the next word in a given text based on all the preceding words;
- It also comes with its set of limitations [9].

Bidirectional Encoder Representations from Transformers (BERT)

- BERT employs a **bidirectional transformer architecture**. It is pre-trained using a massive corpus of text data [10];
- It revolutionized natural language processing, it is not without its limitations;
- Computational intensity and large memory requirements, which can make it challenging to deploy in resource-constrained environments [11].

Table of Contents

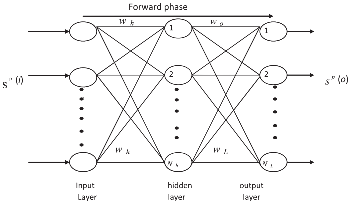
Natural Language Processing (NLP)

- Specialized domain within computer science, artificial intelligence, and linguistics dedicated to investigating the intricacies of automatically generating and comprehending natural human languages.
- In our context, NLP leverages its capabilities to **understand, interpret, and generate** human language to enhance the email composition process [4].
- It can consider the context of the email, the recipient, and the user's writing style to make contextually relevant suggestions.

Neural Networks and Deep Learning

- Inspired by the structure and function of the human brain, which was originated from the simplified mathematical model of **biological neurons** [12].
- The learning process in neural networks involves two main phases: the **forward** pass and the **backward** pass [13].

Figure: Feed-forward neural network.



Source: Hemeida (2020) [14].

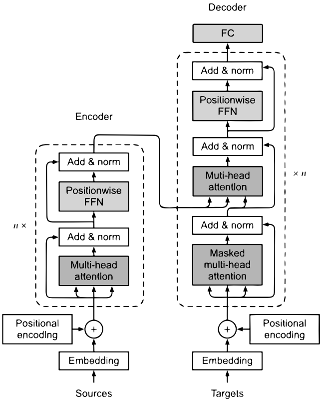
Large Language Model (LLM)

- Advancements in recent years, models like **BERT** [10] and **GPT** [8];
- Can handle NLP tasks, such as **sentiment analysis**, **text classification**, **and machine translation**;
- Considered **unsupervised multitask learners**, as they can learn from vast amounts of text data without the need for task-specific annotations.

Transformer Architecture

- Groundbreaking model architecture introduced in the paper **"Attention is All You Need"** by Vaswani [2];
- Transformer is based on the attention mechanism (self attention);
- Consists of an encoder-decoder structure with **multi-head attention**, enabling the model to weigh the importance of different words or tokens in the input sequence;

Figure: The Transformer - model architecture.



Source: Vaswani 2017 [2], modified.

Attention Mechanism

- Each encoder block consists of two sublayers: **Multi-head** attention and Feedforward network;
- First our model computes the input representation word by word.
- While computing the representation of each word, it relates each word to all other words in the sentence;

Figure: Embeddings for each word.

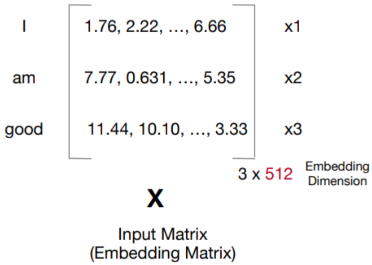
I	x1 = [1.76, 2.22, ..., 6.66]
am	x2 = [7.77, 0.631, ..., 5.35]
good	x3 = [11.44, 10.10, ..., 3.33]

Source: Author.

Attention Mechanism

- We can represent our input sentence using the **input matrix**;
- The dimension of the input matrix will be: **[sentence length x embedding dimension]**;
- Our input matrix dimension will be [3 x 512].

Figure: Embeddings Matrix.



Source: Author.

Attention Mechanism

Attention Function

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

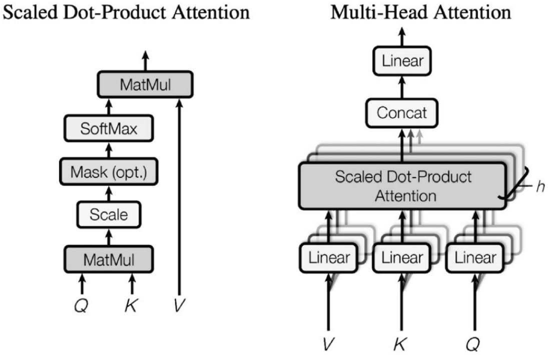
- We can create three new matrices: **Query matrix, Key matrix and Value matrix**.
- The **weight matrices** W^Q, W^K and W^V, are randomly initialized, their optimal values will be learned during training.

The process involves calculating the dot product between the query and each key, dividing the result by the square root of the dimension of the keys (d_k), and then applying a softmax function to obtain the weights assigned to the corresponding values.

Attention Function

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Figure: Left: self-attention. Right: multi-head self-attention.



Source: Vaswani 2017 [2].

Multi-Head Attention

Fundamental component of the Transformer model architecture, introduced by Vaswani et al. [2].

MultiHead Attention

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^0$$

where:

- $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

- The input is first linearly transformed into three sets of **queries, keys, and values**, with each set associated with a specific **attention head**.
- These transformations are parameterized by weight matrices.
- Each attention head then performs the self-attention process, computing the weights for the given queries and keys, and combining the values.

Residual Connections and Layer Normalization

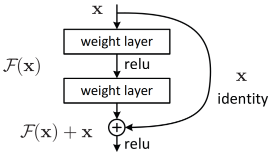
Residual Connection

$$y = F(x) + x$$

where:

- x The input to a specific layer in the network.
- F Represents the transformation applied to the input by the layer.
- y The output of the layer after applying the transformation.

Figure: Residual learning: a building block



Source: Kaiming (2016) [15].

Optimization and Regularization

- These techniques are instrumental in enhancing the **performance and generalization** capabilities of complex models [16];
- Optimization methods are essential for fine-tuning model parameters to **minimize loss** functions and improve predictive accuracy;
- Regularization, on the other hand, helps **mitigate overfitting** by adding penalties to model complexity, thus promoting a balance between fitting the training data and generalizing to unseen data.

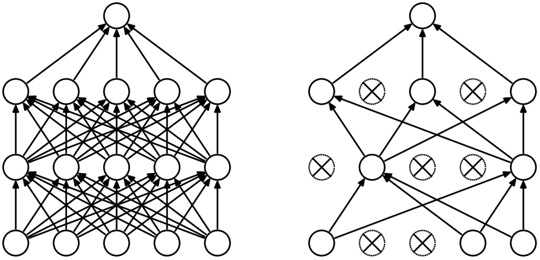
Adam Optimizer

- The Adam optimizer (Adaptive Moment Estimation) and its variant, **AdamW**, represent prominent optimization techniques widely employed in training deep neural networks, adapting learning rates individually for each model parameter, blending stochastic gradient descent (SGD) [17].
- Adam, in particular, maintains **adaptive moving averages** of gradients and squared gradients, facilitating automatic and dynamic learning rate adjustments.

Dropout

- The idea behind dropout is to **randomly drop out** (i.e., set to zero) a fraction of the neurons or units in a neural network during each training iteration.

Figure: Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left.



(a) Standard Neural Net

(b) After applying dropout.

Source: Srivastava (2014) [18].

Table of Contents

Hardware

- NVIDIA GeForce RTX 2060 GPU;
- PyTorch/CUDA;
- Total of 9.5 hours of training process.

Hyperparameters

Among the crucial parameters in the Transformer model we can cite:

- Block and Batch Size;
- Learning Rate;
- Weight Decay;
- Embedding Size;
- Model Size;

Model Pre-training

- Training Data and Batching;
- Evaluation Metrics;
  - Loss Function;
  - Perplexity;
  - Hallucination;

Table: Number of characters in each Dataset, used in pre training phase.

Dataset	**Data Size (characteres)**
COCA	13.0M
NOW	13.0M
iWEB	13.0M
OpenWebTextCorpus	19.0M
Enron Email Dataset	12.0M
TOTAL	72.0M

Log Perplexity

$$logPP(W) = \frac{-1}{m} \sum_{i=1}^{m} log_2(P(w_i))$$

where:

w_i i-th word in the test set.

P SoftMax Probabilistic function.

m Token size.

Model Fine-tuning

- The data utilized for the fine-tuning process within the context of email communication was sourced from the **Enron dataset** (16k emails);
- Hyperparameter Updates and **Layer Freezing**;
    - Use less data;
    - Make process more efficient;
    - Prevent overfitting.

Table of Contents

Code Fragment 5.1: GenerativeLanguageModel layers structure. Source: Author; 2023.

```
class GenerativeLanguageModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.token_embd_table = nn.Embedding(vocab_size, n_embd)
        self.position_embd_table = nn.Embedding(block_size, n_embd)
        self.blocks = nn.Sequential(
            *[Block(n_embd, n_head=n_head) for _ in range(n_layer)])
        self.ln_f = nn.LayerNorm(n_embd)
        self.lm_head = nn.Linear(n_embd, vocab_size)

        self.apply(self._init_weights)
```

`GenerativeLanguageModel` class, a pivotal component of our implementation. This class serves as the cornerstone of our generative language model, encapsulating several crucial element.

Results and Discussions

Code Fragment 5.2: Self Attention block layer structure. Source: Author; 2023.

```
class Block(nn.Module):
    def __init__(self, n_embd, n_head):
        super().__init__()
        head_size = n_embd // n_head
        self.sa = MultiHeadAttention(n_head, head_size)
        self.ffwd = FeedFoward(n_embd)
        self.ln1 = nn.LayerNorm(n_embd)
        self.ln2 = nn.LayerNorm(n_embd)

    def forward(self, x):
        x = x + self.sa(self.ln1(x))
        x = x + self.ffwd(self.ln2(x))
        return x
```

The core of the model resides within the `Block` class Layer, where multiple transformer blocks are arranged sequentially to process the input data.

Results and Discussions

Code Fragment 5.3: Self Attention block layer structure. Source: Author; 2023.

```
class FeedFoward(nn.Module):
    def __init__(self, n_embd):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(n_embd, 4 * n_embd),
            nn.SiLU(),
            nn.Linear(4 * n_embd, n_embd),
            nn.Dropout(dropout),
        )

    def forward(self, x):
        return self.net(x)
```

Code Fragment 5.4: Self Attention block layer structure. Source: Author; 2023.

```
class MultiHeadAttention(nn.Module):
    def __init__(self, num_heads, hd_size):
        super().__init__()
        self.heads = nn.ModuleList()
        self.proj = nn.Linear(hd_size)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        out = torch.cat([h(x) for h
            in self.heads], dim=-1)
        out = self.dropout(self.proj(out))
        return out
```

In the left: simple linear layer followed by a non-linearity; Right we have the `MultiHeadAttention` class which take care of the head of self attention set.

GPT-2 as a baseline for hyperparameter comparison [8]

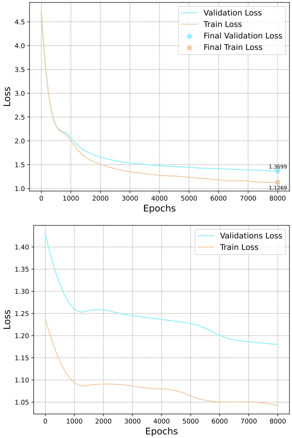
- The parameters were chosen considering the baseline search space and also the limitations faced:

Hyperparams	**Search Space**	**Selected Value**	**Base-line (GPT-2)**
Learning Rate	[0.0001, 0.00001]	0.00001	0.00001
Block Size	[128, 1024]	384	1024
Batch Size	[32, 512]	56	512
Number of Epochs	[6000, 8000]	8000	-
Dropout Rate	[0.1, 0.2]	0.1	0.1
Number of Layers/Heads	[4, 12]	8	12
Embedding dimension	[156, 768]	384	768
Weight Decay	[0.01, 0.000001]	0.000001	0.01
Optimizer	'Adam', 'AdamW'	AdamW	-
Activation function	-	SiLU	GeluNew

Table: Hyperparameter Tuning Results

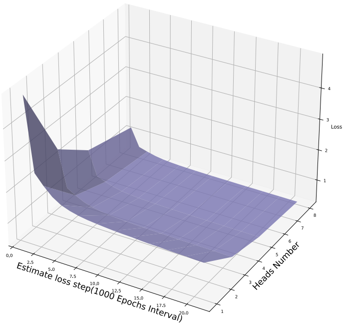
The loss curves demonstrate a consistent **downward trend**, indicating that the model is effectively learning (figure 5.2).
Additionally, another plot the reduction in loss during the fine-tuning phase.

Figure: Loss curves during the pre-training/fine-tuning process.



Results and Discussions

Figure: Loss as a function of epochs for different numbers of heads.



Source: Author, 2023.

One of the main changes tested refers to the amount of attention heads. It was possible to observe the improvement of the model in relation to the number of these.

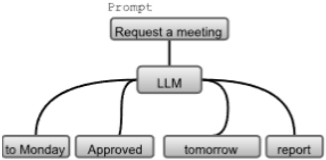


Figure 5.3: Outputs generated by the model given **Request a meeting** text as input.

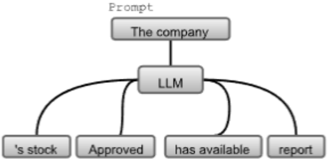


Figure 5.4: Outputs generated by the model given **The company** text as input.

This process showcases the model's proficiency in conjuring fresh and contextually relevant vocabulary, making it a valuable asset in tasks that require adaptive text generation.

Conclusion

The discussed work has endeavored to **develop and explore** an neural network architecture based on Transformers to enhance **email message autocomplete task**.
With the aim of studying the architecture and developing a system with reasonable parameters we've focus on:

- Executed a rigorous **data cleansing** procedure, followed by initial training on a substantial web text corpus;
- Craft a Transformer model capable of effectively capturing intricate **long-range dependencies** inherent in email communications.
- **Fine-tuning** on user-specific email data allowed us to tailor the model to individual preferences;
- Analysis of **hyperparameters** effects on fine-tuning performance.

The approach could deliver a **result satisfactorily** considering the hardware limitations;

Conclusion

Possible future improvements:

- Improve capabilities regarding hardware;
- Apply others **optimization** and **regularization** techniques;
- Incorporation of more extensive and diverse training data;
- Try **another areas** besides the email context.

References I

- Oriol Vinyals Ilya Sutskever and Quoc VV Le. Sequence to sequence learning with neural networks. pages 3104–3112, 2014.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- Jae Shin Zhou Zongwei et al. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.
- Umair Shafique and Haseeb Qaiser. A comprehensive study on natural language processing and natural language interface to databases. *International Journal of Innovation and Scientific Research*, 9(2):297–306, 2014.
- Omar Abdelwahab and Adel Elmaghraby. Deep learning based vs. markov chain based text generation for cross domain adaptation for sentiment classification. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 252–255. IEEE, 2018.
- Antonella FRANZESE, Monica; IULIANO. Hidden markov models. 2019.
- Wai-Ki Ching and Michael K Ng. Markov chains. *Models, algorithms and applications*, 2006.
- Wu J. Radford et al. Language models are unsupervised multitask learners. *OpenAI blog 1(8), 9*, 2019.
- Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

References II

Chang M. W. Lee K. Toutanova K. Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Francisca Adoma Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. Transformer models for text-based emotion detection: a review of bert-based approaches. *Artificial Intelligence Review*, pages 1–41, 2021.

Pitts W. McCulloch, W. S. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5, 115-133.*, 1943.

Bottou L. Bengio Y. Haffner P. LeCun, Y. Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11), 2278-2324.*, 1998.

Hassan S. A. Mohamed Hemeida A. M. et al. Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research. *Ain Shams Engineering Journal, 11(3), 659-675.*, 2020.

Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. Evaluation metrics for language models. 1998.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

References III

- Edna Chebet TOO et al. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 2019.
- Dipanjan Das Ankur Parikh, Oscar Täckström and Jakob Uszkoreit. A decomposable attentionmodel. *In Empirical Methods in Natural Language Processing*, 2016.
- Schmidhuber Hochreiter. Long short-term memory. *Neural computation, 9(8)*, 1997.
- Zhifeng Chen Quoc V Le Mohammad Norouzi Wolfgang Yonghui2016 Wu, Mike Schuster et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arxiv preprint. *arXiv:1609.08144*, 2016.
- Ryder N Subbiah M Brown, Mann B et al. Language models are few-shot learners. *Advances in neural information processing systems, 33, 1877-1901.*, 2020.
- K. Knill and S. Young. Hidden markov models in speech and language processing. *Dordrecht: Springer Netherlands*, 1997.
- Biinghwang RABINER, Lawrence; JUANG. An introduction to hidden markov models. *ieee assp magazine*, 1986.
- Biinghwang RABINER, Lawrence; JUANG. An introduction to hidden markov models for biological sequences. *In: New comprehensive biochemistry. Elsevier.*, 1998.
- B. Rabiner, L. Juang. Hidden markov models for speech recognition—strengths and limitations. in speech recognition and understanding: Recent advances, trends and applications. *Springer Berlin Heidelberg*, 1992.
- Jain L. Medsker, L. *Recurrent neural networks. 5(64-67)*. Design and Applications, 2001.

References IV

Yang Y. He D. Zheng K. Zheng S. Xing C Xiong, R. et al. On layer normalization in the transformer architecture. *International Conference on Machine Learning (pp. 10524-10533). PMLR*, 2020.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyperparameters for fine-tuning. *arXiv preprint arXiv:2002.11770*, 2020.

Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*, 2019.

Bengio Y. Hinton G LeCun, Y. Deep learning. *Nature, 521(7553), 436-444.*, 1943.

Hieu Pham Minh-Thang Luong and Christopher D Manning. Effective approaches to attentionbased neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Tiago Ling, Wang, Marujo, et al. Finding function in form: Compositional character models for open vocabulary word representation. 2015.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus, 2023.

COCA. COCA - corpus of contemporary american english, 2023. Accessed: June 4, 2023.

NOW. NOW - news on the web, 2023. Accessed: June 4, 2023.

IWEB. IWEB - web samples, 2023. Accessed: June 10, 2023.

References V

Kaggle. Enron email dataset, 2023. Accessed: June 10, 2023.

Leif Azzopardi, Mark Girolami, and Keith Van Risjbergen. Investigating the relationship between language model perplexity and ir precision-recall measures. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 369–370, 2003.

Dietrich Klakow and Jochen Peters. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28, 2002.

Hengshuai Yao, Dong-lai Zhu, Bei Jiang, and Peng Yu. Negative log likelihood ratio loss for deep neural network classification. In *Proceedings of the Future Technologies Conference (FTC) 2019: Volume 1*, pages 276–282. Springer, 2020.