Undergraduate Final Project

A Transformer-based Architecture Neural Network Approach to Email Message Autocomplete

Mateus Fernando Felismino da Silva Patriota

Advised by
Prof. Dr. Tiago Figueiredo Vieira
Prof. Dr. Baldoino Fonseca dos Santos Neto

Universidade Federal de Alagoas
Institute of Computing
Maceió, Alagoas
November 9th, 2023

UNIVERSIDADE FEDERAL DE ALAGOAS
Institute of Computing

A TRANSFORMER-BASED ARCHITECTURE NEURAL NETWORK APPROACH TO EMAIL MESSAGE AUTOCOMPLETE

Undergraduate Final Project submited to the Institute of Computing at Universidade Federal de Alagoas as a partial requirement for obtaining a degree of Computer Engineer.

Mateus Fernando Felismino da Silva Patriota

Advisor: Prof. Dr. Tiago Figueiredo Vieira
Co-advisor: Prof. Dr. Baldoino Fonseca dos Santos Neto

Examining Board:

Tiago Figueiredo Vieira	Prof. Dr., IC-UFAL
Baldoino Fonseca dos Santos Neto	Prof. Dr., IC-UFAL
Bruno Georgevich Ferreira	MSc., DEI-UP

Maceió, Alagoas
November 9th, 2023

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos – CRB-4 – 2062

P314t Patriota, Mateus Fernando Felismino da Silva.
A transformer-based architecture neural network approach to email message autocomplete / Mateus Fernando Felismino da Silva Patriota. – 2023.
44 f. : il. color.

Orientador: Tiago Figueiredo Vieira.
Coorientador: Baldoino Fonseca dos Santos Neto.
Monografia (Trabalho de Conclusão de Curso em Engenharia da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2023.

Bibliografia: f. 42-44.

1. Redes neurais. 2. Processamento de linguagem natural. 3. Mensagens eletrônicas – Preenchimento. I. Título.

CDU: 004. 773.3

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Instituto de Computação – IC
Campus A. C. Simões - Av. Lourival de Melo Mota, BL 12
Tabuleiro do Martins, Maceió/AL - CEP: 57.072-970
Telefone: (082) 3214-1401

Trabalho de Conclusão de Curso – TCC

Formulário de Avaliação

Curso: Engenharia de Computação

Nome do Aluno
Mateus Fernando Felismino da Silva Patriota

Nº de Matrícula
18112338

Título do TCC (Tema)
A TRANSFORMER-BASED ARCHITECTURE NEURAL NETWORK APPROACH TO EMAIL MESSAGE AUTOCOMPLETE

Banca Examinadora

Tiago Figueiredo Vieira
Orientador

Documento assinado digitalmente
gov.br TIAGO FIGUEIREDO VIEIRA
Data: 14/11/2023 14:24:17-0300
Verifique em https://validar.iti.gov.br

Baldoino Fonseca dos Santos Neto
Co-orientador

Documento assinado digitalmente
gov.br BALDOINO FONSECA DOS SANTOS NETO
Data: 15/11/2023 09:30:36-0300
Verifique em https://validar.iti.gov.br

Assinatura

Bruno Georgevich Ferreira
Membro da banca

Documento assinado digitalmente
gov.br BRUNO GEORGEVICH FERREIRA
Data: 14/11/2023 15:09:28-0300
Verifique em https://validar.iti.gov.br

Data da Defesa	Nota Obtida
10/11/2023	10,00 (dez)

Observações

Coordenador do Curso
De Acordo

Documento assinado digitalmente
gov.br LEANDRO DIAS DA SILVA
Data: 22/11/2023 18:13:05-0300
Verifique em https://validar.iti.gov.br

Assinatura

Acknowledgements

First of all, I would like to thank God for the life of my family and for all the opportunities given to me during my bachelor studies. I owe eternal gratitude to my family for all their love and support. Especially my mom Adriana and my Grandma, Dona Preta, for dedicating their lives to my development as a human being and my education. I would also like to thank my brothers Vitor and João Paulo for all their friendship and partnership.

I want to express gratitude for the friendships I cultivated throughout the academic period, they were fundamental throughout the process, João Pedrinho, Hiago, Hugo, Cláudio, Ruan, Bruna, Derek, Adrielly, Roger, Jhonnye, Cabral, Lucas Massa, Marcus, Igor, and Luana. Also, I am immensely grateful for having met, during my undergraduate studies, my girlfriend, Maria Antônia.

I am grateful to the entire teaching staff and other employees at the Computing Institute. I would like to thank, in particular, the professors who guided me, Dr. Tiago Figueiredo Vieira and Dr. Baldoino Fonseca. I would also like to thank Professor Thiago Damasceno Cordeiro for his important support role towards the institute's students in general.

November 9th, 2023, Maceió - AL.

Resumo

Este estudo tem como propósito o desenvolvimento e exploração de uma arquitetura de rede neural fundamentada em *transformers*, com o intuito de aprimorar a tarefa do preenchimento de mensagens de e-mail. O presente trabalho detalha o abrangente processo de concepção, treinamento e avaliação da referida arquitetura, ao mesmo tempo em que investiga o impacto de diversos hiperparâmetros e camadas. O enfoque central está voltado para a criação de um modelo *transformer* capaz de capturar as complexas interdependências de longo alcance inerentes às comunicações por e-mail. Antes de proceder com o treinamento, uma extensa etapa de limpeza dos dados é realizada. Considerando as restrições de recursos de hardware, a citada arquitetura é submetida à uma fase inicial de treinamento em um extenso conjunto de textos da web de acesso público. Isso é seguido por uma meticulosa avaliação em um conjunto de testes independente. Posteriormente, efetua-se o processo de ajuste fino utilizando um conjunto de dados específico relacionado ao contexto de mensagens eletrônicas, compreendendo e detalhando os efeitos de diferentes hiperparâmetros no desempenho do modelo refinado. Métricas de desempenho são apresentadas tanto antes quanto depois desse procedimento, permitindo uma comparação direta do impacto do ajuste fino na qualidade das respostas geradas. Desta forma, proporciona-se uma compreensão da aplicabilidade dos modelos do tipo *transformer* no contexto do preenchimento automático de e-mails, sendo possível compreender limitações, identificar áreas de melhoria, definição de parâmetros e desenvolvimento de base sólida para a implementação à nível de código desse tipo de tecnologia em aplicações de comunicação por e-mail.

***Palavras-chave*: Processamento de linguagem natural, Redes neurais, Transformers, Arquitetura generativa, Preenchimento de Mensagens.**

Abstract

This research endeavors to develop and investigate an innovative neural network architecture based on transformers, aiming to enhance the automatic completion of email messages. The study outlines the comprehensive process of architecture creation, training, and evaluation, while exploring the impact of diverse hyperparameters and layers. The central focus lies in crafting a transformer model proficient in capturing intricate long-range dependencies inherent in email communications. Prior to training, an extensive data cleansing procedure is executed. Given hardware resource constraints, the architecture undergoes preliminary training on a substantial corpus of publicly available web texts, followed by rigorous evaluation on an independent test dataset. Subsequent fine-tuning is performed on individualized user email data, accompanied by a thorough analysis of hyperparameter effects on the performance of the fine-tuned model. This analysis encompasses a comparative assessment of performance metrics both before and after the fine-tuning process. Through these objectives, this study aspires to study the construction of the architecture mentioned in relation to the email autocompletion mechanisms, taking advantage of the resources of neural networks based on transformers. In this way, an understanding of the applicability of transformer models in the context of email message generation is provided, allowing for the identification of limitations, areas of improvement, parameter tuning, and the development of a solid foundation for the code-level implementation of this technology in email communication applications.

***Keywords*: Natural language processing, Neural networks, Transformers, Generative architecture, Message Autocomplete.**

List of Figures

List of Tables

List of Symbols

θ Model parameters

$\mathcal{L}$ Loss function

∇ Gradient

η Learning rate

σ Sigmoid activation function

$\mathbb{R}$ Set of real numbers

$\mathbf{X}$ Feature matrix

$\mathbf{Y}$ Target variable

$\hat{\mathbf{Y}}$ Predicted values

$\odot$ Element-wise multiplication operation

$\mathbf{W}$ Weight matrix

$\mathbf{b}$ Bias vector

$\otimes$ Convolution operation

$\mathcal{O}$ Order of complexity

$\rightarrow$ Transformation or mapping

List of Abbreviations

AI Artificial Intelligence

NN Neural Network

DL Deep Learning

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

CNN Convolutional Neural Network

MLP Multilayer Perceptron

SILU Sigmoid Linear Unit function

DNN Deep Neural Network

BERT Bidirectional Encoder Representations from Transformers

GPT Generative Pre-trained Transformer

NLP Natural Language Processing

SGD Stochastic gradient descent

NLL Negative Log-Likelihood

Summary

Chapter 1

Introduction

1.1 Motivation

Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks (Hochreiter, 1997), have firmly established themselves as state-of-the-art approaches for addressing sequence modeling tasks, such as language modeling and machine translation (Ilya Sutskever and Le, 2014), employing encoder-decoder architectures (Yonghui2016 Wu et al., 2016). Nevertheless, these conventional architectures frequently encounter challenges, such as the management of long-range dependencies and computational inefficiencies inherent to their sequential nature.

In recent years, the Transformer architecture (Vaswani et al., 2017) has emerged as a groundbreaking alternative, showcasing remarkable success in a variety of natural language processing tasks. The attention mechanism has a relevant role in the Transformer architecture, granting it the ability to capture global dependencies and efficiently parallelize computations. This departure from recurrent connections has resulted in substantial enhancements in training efficiency and performance across intricate language-related tasks.

In this work, we introduce a solution to trackle the problem of Email Message Autocompletition utilizing a neural network architecture based on Transformers. Our goal is to create a system capable of effectively forecasting the subsequent words or phrases within an email message. By leveraging the robust attention mechanisms inherent in the Transformer architecture ((Ankur Parikh and Uszkoreit, 2016); (Vaswani et al., 2017)), we strive to surpass the constraints of conventional recurrent-based models and delve into the possibilities for achieving even greater accuracy and context-awareness in email message predictions.

One of the principal motivations behind this endeavor is to attain an in-depth comprehension of the structure and operation of a Transformer architecture. Constructing the model from scratch and investigating parameter and layer adjustments will yield valuable

insights into the effects of these modifications on the model's performance. Through this methodology, our objective is to make substantial enhancements in the Email Message Autocomplete task, empowering the model to generate suggestions the are contextually relevant for users.

Furthermore, another pivotal aspect we aim to investigate is the fine-tuning process of the Transformer-based architecture (Zhou Zongwei et al., 2017). Fine-tuning enables us to adapt a pre-trained model, initially trained on a different and general corpus of data, to the specific task at hand – Email Message Autocomplete in our case. Through fine-tuning, we can harness the knowledge and linguistic representations acquired from a broader dataset and then refine them using a smaller, task-specific dataset. This process frequently results in improved performance and superior generalization for the designated task (TOO et al., 2019). It allows us to effectively tailor the Transformer's attention mechanisms and parameters to capture the intricacies of email message completions, thereby optimizing the model's capacity to predict contextually relevant suggestions in various email contexts and for individual users.

1.2 Objectives

1.2.1 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. The process encompasses in-depth training of this architecture, followed by an evaluation of its performance using a dataset of email messages.

1.2.2 Specific Objectives

1. To design a transformer-based neural network architecture that is able to learn long-range dependencies in email messages.

2. To pre train the proposed architecture on a large dataset of open web texts.

3. To evaluate the performance of the proposed architecture on a held-out test set.

4. To fine-tune the proposed architecture on a smaller dataset of user-specific email messages.

5. To investigate the effects of different hyperparameters on the performance of the fine tuned model.

1.3 Work Organization

This work was organized into six chapters that report the steps followed during the development of the architecture of Transformer neural networks, going into theoretical details and application of the technologies involved (Chapter 2). Chapter 3 provides a theoretical foundation, explaining the machine learning principles underpinning the Transformer architecture. It also outlines the approach taken to construct the Transformer code and the strategies used for hyperparameter optimization. Additionally, the chapter discusses the evaluation metrics employed to assess the performance of the developed models.

In Chapter 4, the step-by-step process of building the neural network architecture is elaborated. This includes details about acquiring the dataset, generating a calibration curve, preprocessing the data, constructing curve fitting models, and optimizing them. The subsequent chapter, Chapter 5, is dedicated to presenting and analyzing the results obtained from the experiments conducted during this research. Here, the performance and effectiveness of the developed Transformer-based models are thoroughly examined, along with a discussion of their implications and potential applications. Finally, in the concluding section 6 , the key findings and contributions of this work are summarized. The conclusion also sheds light on the broader significance of the study, its limitations, and avenues for future research in the domain of Transformer-based neural networks and Large Language Models.

Chapter 2

Literature Review

2.1 The Problem

The realm of artificial intelligence (AI) has experienced a relative huge development, significantly influencing numerous sectors such as email messaging systems. One notable advancement is the introduction of autocomplete technology in email interfaces. Unfortunately, at the point of writing this report (October 2023), there is sparse information relating to the latest developments in this area. Email autocomplete is an innovative interaction design that aims to predict user input, enhancing the usability and efficiency of email correspondence. The absence of adequate information on this topic signifies the need for more research, exploration, and development within this field.

2.2 Natural Language Processing and Machine Learning Approach

Email autocomplete task represents an innovative interaction design that harnesses AI algorithms, particularly Natural Language Processing (NLP) and Machine Learning (ML), to predict and suggest the completion of a user's sentences (Shafique and Qaiser, 2014). While specific advancements in auto-email completion remain relatively limited in recent reports, these AI techniques play a pivotal role in semantic prediction tasks, hinting at their potential to shape the future of auto-complete email technology.

NLP, with its ability to understand, interpret, and generate human language in a meaningful and grammatically correct manner, aligns seamlessly with the requirements of efficient auto-complete technology. This technology aims to predict user input accurately and sensibly within an email context, thereby improving user experience and productivity.

In contrast, ML algorithms can be trained on extensive text datasets, allowing them to learn specific writing styles, common phrases, and patterns. This acquired knowledge can be harnessed to predict what the user is likely to type next with increasing accuracy over

time. The adaptability of AI and its algorithms to user context, language, and patterns holds immense potential for the evolution of auto-complete email technology.

2.3 Markov Chains

In addition to NLP and ML, Text Generation with Markov Chains is an approach that can be integrated into email auto-completion systems. Markov Chains are probabilistic models that capture the likelihood of transitioning from one state (word) to another in a sequence. This technique has been employed in various text generation tasks, including autocomplete, creative writing, machine translation, text summarization, question answering, and chatbots (Abdelwahab and Elmaghraby, 2018). It's powerful tool that can be used for a variety of tasks, including education, entertainment, and research.

It can generate contextually relevant suggestions for email auto-completion by analyzing the user's input and predicting the most likely next word or phrase based on the preceding context. While Markov Chains may not exhibit the same semantic understanding as NLP-based approaches, they can provide quick and contextually relevant suggestions, especially for common phrases and expressions.

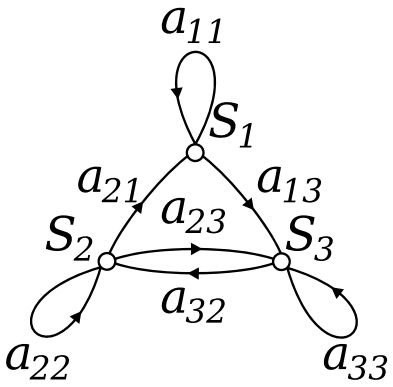


Figure 2.1: Simple Markov chain graph representation. Source: Ching and Ng (2006).

A discrete-time Markov process is a sequence of stochastic variables $X_1, X_2, X_3, \ldots$ exhibiting the Markovian property, which means that the likelihood of transitioning to the next state solely relies on the current state and doesn't depend on preceding states:

$$\Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \Pr(X_{n+1} = x \mid X_n = x_n) \tag{2.1}$$

Markov chains are often described by a sequence of directed graphs, where the edges of graph n are labeled by the probabilities of going from one state in time n to other states in time $n + 1$. Can be seen more detailed in 2.1.

Despite the apparent lack of specific recent developments in auto-email completion,

this does not necessarily signify stagnation in the field. It is plausible that advancements are happening discreetly or within proprietary technology that is not publicly disclosed, suggesting that the sector is likely in an innovation phase with research and development taking place behind the scenes.

2.4 GPT-2

The GPT-2 model, as introduced in 2019 from OpenAI (Radford et al., 2019), is a substantial advancement in natural language processing. It is an unidirectional transformer architecture, pre-trained through language modeling on an extensive dataset comprising around 40 gigabytes of textual information.

The core idea behind GPT-2 is to predict the next word in a given text based on all the preceding words. What makes GPT-2 especially remarkable is the scale of its training data and the size of the model itself. It boasts a huge 1.5 billion parameters and was trained on a vast dataset comprising 8 million web pages. This large-scale training approach allows GPT-2 to not only excel at the primary language modeling task but also to exhibit competence in various tasks spanning diverse domains. In essence, GPT-2 represents a significant evolution from its predecessor, GPT, with over ten times the number of parameters and training data, making it a powerful and versatile language model.

While GPT-2 undeniably represents a remarkable leap in natural language processing, it also comes with its set of limitations (Floridi and Chiriatti, 2020). One of the primary concerns is the generation of biased or inappropriate content, as it is essentially an autoregressive model that generates text based on patterns it has learned from its training data, which can sometimes include harmful or biased language present on the internet. Additionally, GPT-2 can occasionally produce text that is factually incorrect, as it doesn't possess an inherent understanding of the accuracy of the information it generates. Moreover, its large size and computational requirements make it challenging for smaller organizations and researchers to use effectively. Addressing these limitations remains an ongoing challenge in the development of advanced language models.

2.5 Bidirectional Encoder Representations from Transformers

The BERT (Bidirectional Encoder Representations from Transformers) model, introduced in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" (Devlin, 2018) represents a groundbreaking development in natural language processing.

BERT employs a bidirectional transformer architecture, which is a departure from traditional unidirectional models. It is pre-trained using a massive corpus of text data, allowing it to capture contextual information by considering both the left and right context of each word. This bidirectionally greatly enhances its understanding of language.

What sets BERT apart is its ability to perform a range of language understanding tasks with high accuracy after fine-tuning. This versatility is achieved because BERT is pre-trained on a substantial amount of text, providing it with a deep understanding of language semantics and context.

While BERT has undoubtedly revolutionized natural language processing, it is not without its limitations (Acheampong et al., 2021). One significant constraint is its computational intensity and large memory requirements, which can make it challenging to deploy in resource-constrained environments. Additionally, BERT's pre-training process involves substantial amounts of text data, which can raise concerns about privacy and data usage. Furthermore, BERT may struggle with out-of-domain or low-resource languages, as its pre-training data is predominantly in English, and its performance can significantly drop when applied to languages or domains for which it has not been fine-tuned.

2.6 Google's Help Me Write

Help Me Write is a feature that seamlessly integrates AI into the realm of professional communication and documentation as service/product. This tool operates within Gmail, email platform developed by Google, and Google Docs, opular collaborative word processing tool, and is designed to facilitate the creation of polished, error-free text and draft emails with AI. It accomplishes this by generating drafts of content based on user prompts, thereby saving time and enhancing the overall quality of written materials. The core technology powering Google Help Me Write is based on Generative Pre-trained Transformer (GPT) models. These AI models are trained on vast datasets of text and code, enabling them to decipher the intricacies of human language patterns. Consequently, GPT models can generate text that adheres to grammatical conventions while maintaining semantic coherence.

Chapter 3

Theoretical Foundation

3.1 Natural Language Processing (NLP)

Nowadays the vast reservoir of natural language text contains a wealth of knowledge. However, the sheer volume makes it increasingly challenging for humans to uncover meaningful insights within the constraints of time. Enter automated Natural Language Processing (NLP), which strives to efficiently and accurately undertake this task, akin to how a human might approach it (albeit within limitations imposed by the volume of text). As an interdisciplinary field that combines elements of linguistics, computer science, and artificial intelligence to enable machines to effectively understand, interpret, and generate human language. With the advancement of technology and the ever-increasing volume of textual data available, NLP plays a crucial role in various applications, ranging from recommendation systems to chatbots and sentiment analysis. In our context, NLP leverages its capabilities to understand, interpret, and generate human language to enhance the email composition process. NLP algorithms can analyze the text as the user starts composing an email and suggest the next word or phrase in real time. This feature is especially helpful in reducing typing efforts and increasing the speed of email composition. It considers the context of the email, the recipient, and the user's writing style to make contextually relevant suggestions.

3.2 Neural Networks and Deep Learning

Neural networks, inspired by the structure and function of the human brain, which was originated from the simplified mathematical model of biological neurons established by McCulloch and Pitts in 1943 (McCulloch, 1943). At its core, a neural network is a computational model comprised of interconnected nodes, known as neurons, organized in layers. The information flows through these neurons, and the network learns to perform specific tasks through the process of training. The structure of a neural network typically

consists of an input layer, one or more hidden layers, and an output layer, we can see an example in the Figure 3.1. The neurons in each layer are connected to the neurons in the subsequent layers, and each connection is associated with a weight. The input layer receives the data, which then propagates through the network, undergoing linear and non-linear transformations in the hidden layers, eventually producing an output in the output layer.

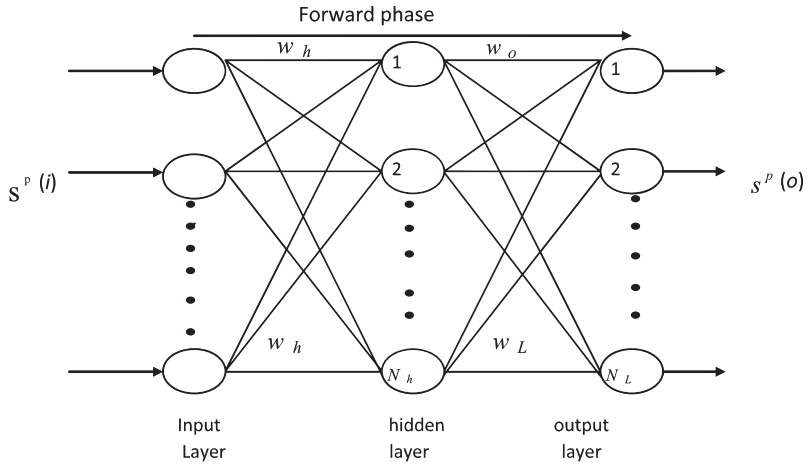


Figure 3.1: Feed-forward neural network. Source: Hemeida A. M. et al. (2020)

The learning process in neural networks involves two main phases: the forward pass and the backward pass. In the forward pass, the input data is fed into the network, and the output is generated. During the backward pass (training phase), using a loss function the output is compared to the desired target, and the network's performance is measured using a loss function. The gradients of the loss with respect to the weights are computed using the backpropagation algorithm, and these gradients guide the update of weights through optimization techniques like stochastic gradient descent (SGD) (LeCun, 1998).

The capability of neural networks to automatically learn complex representations from data, known as feature learning, has enabled their widespread application in various domains, such as image recognition, natural language processing, and game playing. Convolutional Neural Networks (CNNs) (LeCun, 1943) and Recurrent Neural Networks (RNNs) (Medsker, 2001) are notable examples of specialized architectures that have achieved remarkable success in their respective domains.

3.2.1 Large Language Model (LLM)

The NLP has witnessed significant advancements in recent years, with the emergence of powerful language models like BERT (Bidirectional Encoder Representations from Transformers) (Devlin, 2018) and GPT (Generative Pre-trained Transformer)(Radford et al., 2019), as discussed in 2.

In essence, the LLM undergoes a pre-training phase wherein it learns to predict the subsequent word in a sequence without explicit labels, leveraging a substantial unlabeled dataset. The transformer architecture, distinguished by its attention mechanisms, facilitates the model in effectively capturing long-range dependencies and contextual information. The model transforms words into embeddings—high-dimensional vectors representing the semantic meaning of words and their contextual nuances.

During pre-training, the model acquires a set of parameters that encode statistical patterns and structures within the language. These parameters can subsequently undergo fine-tuning on specific downstream tasks, utilizing smaller labeled datasets to enable the model's adaptation to task-specific intricacies.

Post-training, the LLM finds application in diverse NLP tasks, such as generating responses, predicting subsequent words in a sequence, or undertaking various language-related endeavors based on the acquired patterns during training. The LLM's efficacy lies in its capacity to pre-train on extensive unlabeled data and then fine-tune for particular tasks, a characteristic that underscores its success in numerous natural language understanding and generation applications.

These language models, such as BERT and GPT, are built on the transformer architecture and have revolutionized NLP tasks. They are considered unsupervised multitask learners, as they can learn from vast amounts of text data without the need for task-specific annotations. This ability allows them to handle various NLP tasks, such as sentiment analysis, text classification, and machine translation. Moreover, language models like BERT and GPT are also recognized as few-shot learners, enabling them to adapt quickly to new tasks with minimal training data Brown et al. (2020).

3.2.2 Transformer Architecture

The Transformer is a model architecture introduced in the paper "Attention is All You Need" by Vaswani et al, (Vaswani et al., 2017). It changed natural language processing tasks by utilizing self-attention, allowing the model to capture long-range dependencies in input data without relying on recurrent or convolutional structures. The architecture 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, see in Figure 3.2.

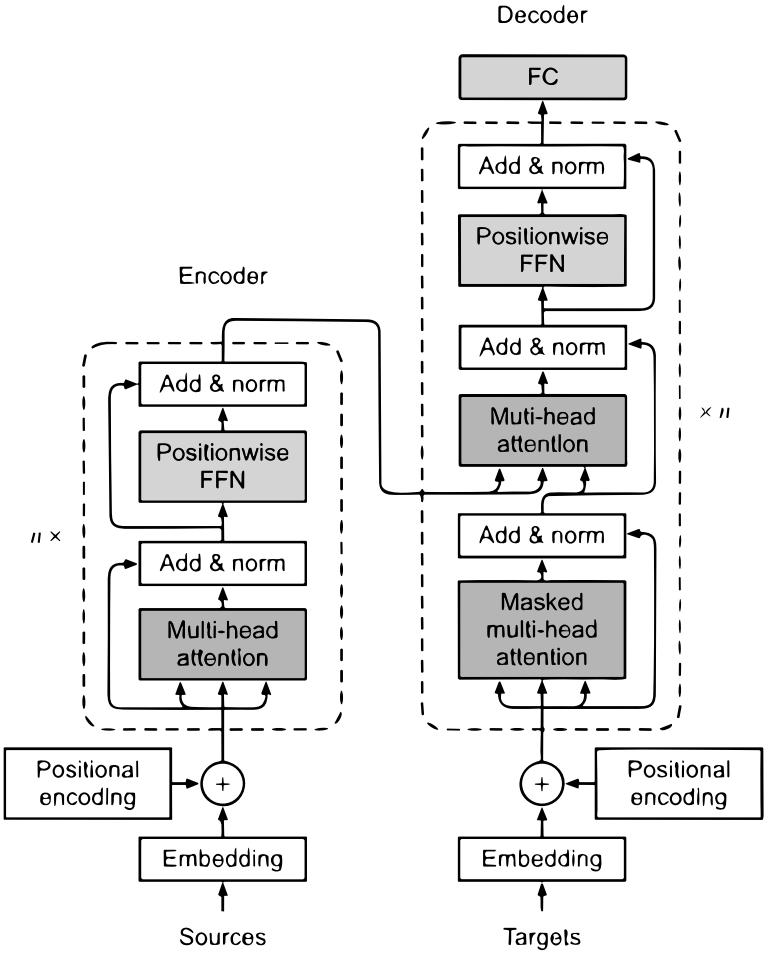


Figure 3.2: The Transformer - model architecture. Source: Vaswani et al. (2017), modified.

Positional encodings are added to provide information about word order, and feed-forward neural networks introduce non-linearity. Layer normalization and residual connections stabilize training and facilitate the flow of gradients during backpropagation. The Transformer has been highly successful in various NLP tasks, with models like BERT and GPT achieving state-of-the-art results on benchmarks, showcasing its power and versatility in natural language processing.

3.2.3 Attention Mechanisms

The attention idea was firstly proposed in (Vaswani et al., 2017) as discussed in the last subsection (3.2.2) and rapidly provided great advances in the field of natural language processing (Devlin, 2018).

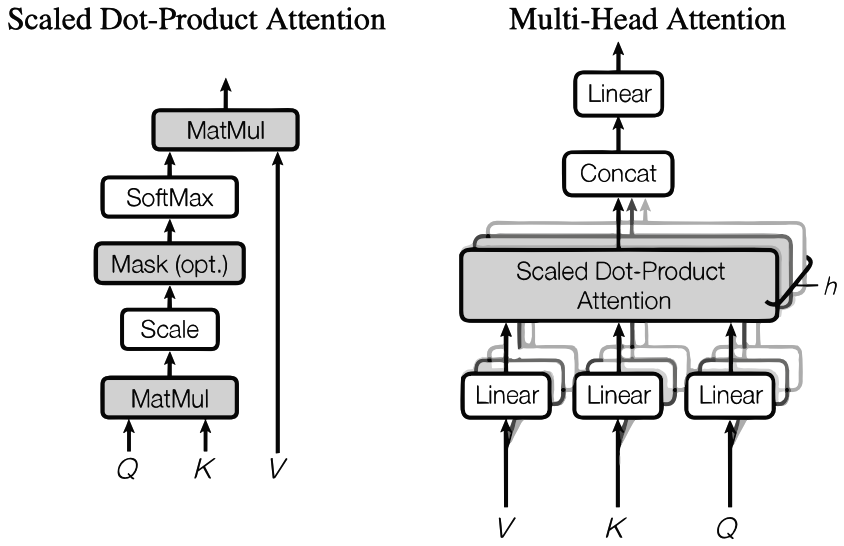


Figure 3.3: Left: self-attention. Right: multi-head self-attention. Source: Vaswani et al. (2017)

An attention function is a mechanism that takes a query vector and a set of key-value pairs as inputs and produces an output vector. 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. This attention mechanism, known as **Scaled Dot-Product Attention**, as we can see in Figure 3.3, ensures that the output is a weighted sum of the values based on the compatibility between the query and keys.

To efficiently compute the attention function for multiple queries, we organize the queries, keys, and values into matrices Q, K , and V, respectively, and obtain the matrix of outputs by performing matrix operations on these packed representations. A matrix is computed of outputs as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.1}$$

The equation 3.1 involves taking the softmax of the dot product between the query and key matrices divided by the square root of the dimension of the keys, and then multiplying the result by the value matrix. This allows models to assign different levels of attention to different parts of the input data, a fundamental concept that has significantly improved the performance of various natural language processing tasks.

The softmax operation applied to the dot product of query and key matrices acts as a kind of "selector." It assigns higher weights to key-value pairs that are more compatible or relevant to the given query. In essence, this means the model can focus its attention on the parts of the input data that are most informative or contextually relevant to the

task at hand. This adaptability to varying degrees of relevance within the input data is a game-changer.

In sum, the attention mechanism's ability to assign varying levels of attention to different parts of the input data based on their relevance is a crucial breakthrough in natural language processing. It empowers models to understand context, capture long-range dependencies, and deliver state-of-the-art performance in tasks like machine translation, text summarization, and question-answering, making it an indispensable tool in the NLP toolkit.

3.2.4 Residual Connections and Layer Normalization

The residual connections, also known as skip connections, are an architectural design element introduced in the Residual Network model (Kaiming He, 2016). The main idea behind residual connections is to facilitate the training of very deep neural networks by allowing the gradients to flow more effectively during back propagation. In a standard deep neural network, each layer sequentially transforms the input data into a higher-level representation.

However, as the network becomes deeper, the vanishing or exploding gradient problem can arise. The gradients can become too small or too large, making it challenging for the model to learn effectively and leading to training difficulties. Address this issue by introducing shortcut connections that bypass one or more layers in the network. Instead of learning a direct mapping from the input to the output, residual connections learn the residual or the difference between the input and the desired output. These residuals are then added back to the input, effectively skipping over some layers (Kaiming He, 2016). The idea of residual connection can be see in Figure 3.4; The residual connection is a simple equation that can be represented as follows:

$$y = F(x) + x \tag{3.2}$$

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.

We alse use the called LayerNorm operation (Xiong et al., 2020) is typically applied after the linear transformation in each layer of the neural network. It normalizes the activations independently for each example in the batch and for each feature dimension,

making it different from Batch Normalization, which normalizes across the entire batch. In this paper we used the LayerNorm function implemented in pytorch library.

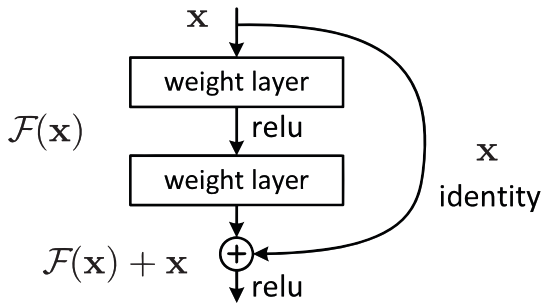


Figure 3.4: Residual learning: a building block. Source: Kaiming He (2016)

3.3 Optimization and Regularization

Optimization and regularization techniques play a pivotal role in the realm of machine learning and data science. These techniques are instrumental in enhancing the performance and generalization capabilities of complex models (Chen et al., 1998). 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.

3.3.1 Adam Optimizer

The Adam optimizer (Adaptive Moment Estimation) and its variant, AdamW, represent prominent optimization techniques widely employed in training deep neural networks, encompassing deep learning models such as convolutional neural networks (CNNs) and Transformer-based language models (Kingma and Ba., 2015). These optimizers are renowned for their adeptness in adapting learning rates individually for each model parameter, blending stochastic gradient descent (SGD) methods with first and second-order moment estimations.

Adam, in particular, maintains adaptive moving averages of gradients and squared gradients, facilitating automatic and dynamic learning rate adjustments. This adaptability to varying gradient scales across parameters equips Adam to handle a diverse array of optimization scenarios effectively, We have the equation that represents the Adam Optimizer, considering that w_t is the vector of weights updated in time t, 3.3.

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.3}$$

with

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.4}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.5}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.6}$$

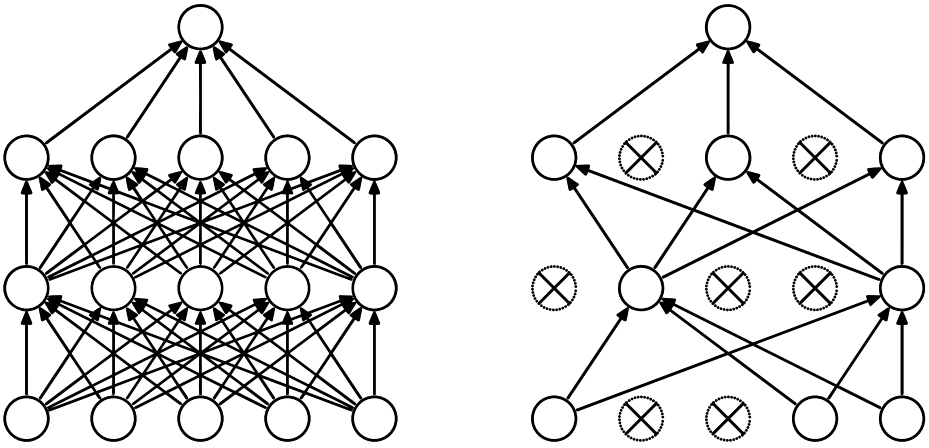
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.7}$$

However, the Adam optimizer may encounter convergence challenges, especially in tasks like training Transformer-based language models. Enter AdamW, a variant that introduces a weight decay term into the loss function. This weight decay term functions as a regularization penalty on model weights, curbing their growth during training. Consequently, it enhances training stability and mitigates overfitting issues that can manifest with the vanilla Adam optimizer, we can see the AdamW equation as follows:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \left(\frac{1}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t + w_{t,i} \theta_{t,i} \right), \forall t \tag{3.8}$$

3.3.2 Dropout

Dropout is a regularization technique commonly used in neural networks to prevent overfitting and improve generalization (Srivastava et al., 2014). Additionally, it enables an efficient approach to combining a vast number of diverse network architectures. When we mention "dropout" it means the act of temporarily excluding units, both hidden and visible, from the neural network structure. This exclusion involves removing the unit along with all its incoming and outgoing connections, as illustrated in Figure 3.5.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.5: 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. Source:(Srivastava et al., 2014).

Chapter 4

Methodology

4.1 Hardware

The model training process was conducted on a single machine equipped with an NVIDIA GeForce RTX 2060 GPU. Utilizing PyTorch, open-source machine learning framework primarily developed by Facebook's AI Research lab (FAIR), for constructing the neural network, alongside the native Python libraries stack for data preprocessing, each training step required approximately 1.2 seconds for the base models, as outlined next in the paper's . The base model was trained over a period of 8000 steps, totaling 5 hours. During the fine-tuning step, the processing time per step was reduced to 1.0 second. The fine-tuned models were trained for 8000 steps, which took approximately 4.5 hours.

4.2 Model Architecture

4.2.1 Hyperparameter

To train our LLM, focused on email message context, we conducted experiments using as a base the Transformer architecture (Vaswani et al., 2017) and some layers classes implemented in the PyTorch framework.

Among the crucial parameters in the Transformer model are the number of layers, which determines the depth of the model and its ability to capture complex patterns in the data. Additionally, the number of attention heads allows the model to attend to different positions of the input sequence simultaneously (Devlin, 2018), enabling it to understand diverse linguistic relationships. The embedding dimension plays a critical role in representing words in a continuous and dense vector space, impacting the model's ability to learn meaningful word representations.

Also, is very important to focus on the context size, influences how many surrounding words the model considers when performing language modeling. A larger context facili-

tates the capture of longer-range dependencies in the context of words. Also, the learning rate is a crucial hyperparameter, as it governs the rate at which the model updates its weights during pre-training and fine tuning, directly affecting the training process's stability and efficiency. By conducting extensive experiments and fine-tuning these parameters, we can optimize the performance of our LLM based on the Transformer architecture, making it effective in the email message completion task. A better explanation and numerical exposure of the tuning hypermarameters will be show in the result, Chapter 5.

4.2.2 Encoder and Decoder Stacks

The structure consists of two main parts: the encoder and the decoder. Both are constructed using attention blocks, feedforward layers, and residual connections with layer normalization. Is possible to see a simplified diagram in the Figure 4.1

Encoder: The encoder processes the input sequence and captures its representations. It comprises a stack of several identical layers. Each encoder layer consists of two main components: a multi-head attention layer and a feedforward layer (composed of fully connected neural networks).

Decoder: The decoder generates the output sequence based on the representations learned by the encoder. Similar to the encoder, the decoder is also composed of a stack of several identical layers. Each decoder layer has three main components: a multi-head attention layer, a cross-attention layer (which attends to the encoder's context), and a feedforward layer.

The proposed Transformer architecture consists of a stack of N encoders. The output of one encoder is sent as input to the encoder above it. The final encoder returns the representation of the given source sentence as output. We feed the source sentence as input to the encoder and get the representation of the source sentence as output.

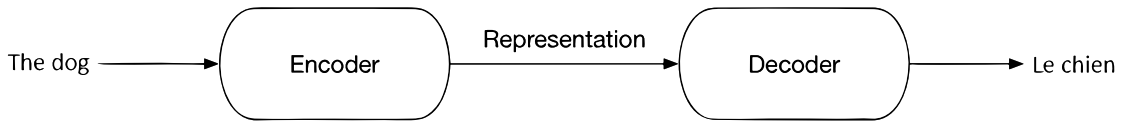


Figure 4.1: Simplified decoder-encoder diagram. Source: Author, 2023.

4.2.3 Multi-Head Attention

Multi-Head Attention is a fundamental component of the Transformer model architecture, introduced by Vaswani et al. (Vaswani et al., 2017). It extends the basic self-attention mechanism by incorporating multiple attention heads in parallel. Each attention

head independently learns different relationships and patterns from the input, providing the model with the ability to attend to multiple aspects of the data simultaneously, as discussed in the section .

In Multi-Head Attention, 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 separate weight matrices. Each attention head then performs the self-attention process, computing the attention weights for the given queries and keys, and combining the values accordingly. The outputs of all attention heads are concatenated and linearly transformed again to produce the final output of the Multi-Head Attention layer.

This idea enables the model to simultaneously focus on information from various representation subspaces at different positions in the input data. In contrast, a single attention head's approach of averaging inhibits the model's ability to capture diverse patterns and dependencies effectively. By using multiple attention heads, the model can attend to different aspects of the input simultaneously, enhancing its capability to learn complex relationships and extract meaningful information from the data. This ability to jointly attend to diverse subspaces greatly contributes to the success of multi-head attention in various tasks, making it a pivotal component of modern neural network architectures like the Transformer. See:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^0 \tag{4.1}$$

$$where\, head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{4.2}$$

The heads are resultants of the Attention Function, see the 3.1 equation. The weight matrix W^0 aims to perform a linear transformation on the concatenated vector, allowing the model to adjust and combine information from different attention heads before producing the final output. This final operation is crucial to ensure that the relevant information extracted from all attention heads is properly utilized in the model's learning process.

4.3 Model Pre-training

To train our Transformer model, focusing on email message completion, character-level tokens were employed. This approach involved representing each character in the messages as a token, allowing the model to capture fine-grained linguistic patterns and context at the character level (Ling et al., 2015). By using character-level tokens, the model gains the ability to generate more accurate and contextually appropriate completions, especially in cases where words are misspelled, rare, or when handling domain-specific jargon commonly

found in emails.

4.3.1 Pre-Training Data and Batching

The data utilized for pre-training our model were collected from several large-scale datasets, namely the Corpus of Contemporary American English (COCA, 2023), News on the Web (NOW, 2023), The International Web (IWEB, 2023), the OpenWebTextCorpus (Gokaslan and Cohen, 2023). Was also added part of the the Enron Email Dataset (Kaggle, 2023) giving some email data for the pre-trained model. We explore the impact of incorporating this varied and extensive data in the pre-training phase, aiming to enhance the model's ability to understand and generate human-like language in English. In addition, a batch size of 56 was used, representing how many independent sequences we will process in parallel.

Table 4.1: 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

4.4 Evaluation Metrics

4.4.1 Loss

In this study, we explore the application of Negative Log-Likelihood (NLL) as a loss function for our model. NLL has proven to be a popular choice in various machine learning tasks, particularly in classification problems. Its effectiveness lies in its ability to optimize the model's parameters by penalizing large errors and encouraging accurate predictions (Yao et al., 2020). By minimizing the NLL, our model aims to maximize the likelihood of generating the observed data, effectively transforming the learning process into a probability estimation task. This choice of loss function is well-suited for our research objectives, as it not only facilitates efficient training but also provides a principled framework to quantify uncertainty in our predictions, which is crucial for reliable decision-making in real-world applications.

4.4.2 Perplexity

The standard way to assess the quality of a statistical language model (LM) is typically expressed using perplexity (Azzopardi et al., 2003). Perplexity is a measure that quantifies the uncertainty or surprise of the model when predicting the next word in a sequence. It is calculated as the exponential of the negative normalized predictive likelihood under the model. In practical terms, perplexity provides an estimation of the expected word error rate, particularly in applications like speech recognition systems. In this paper we used the called log perplexity (Klakow and Peters, 2002):

$$logPP(W) = \frac{-1}{m} \sum_{i=1}^{m} log_2(P(w_i)) \tag{4.3}$$

Where:

w_i i-th word in the test set

P SoftMax Probabilistic function

m Token size

4.5 Model Fine-tuning

Within the machine learning and neural network context, the fine-tuning constitutes the process of adapting a pre-trained model to perform optimally on a specific task or dataset, thus tailoring it to meet distinct requirements, (Zhou Zongwei et al., 2017). This involves the judicious adjustment of the model's parameters, encompassing weights and biases, through the utilization of a lower learning rate during training on the target data. Fine-tuning strategically capitalizes on the knowledge and feature representations acquired during the initial pre-training phase, often executed on a diverse and extensive dataset. The aim is to harness this prior knowledge effectively and fine-tune the model to excel in a more specialized and task-specific domain. This approach not only expedites training but also holds immense potential for yielding highly proficient models, rendering it an indispensable technique across various domains, spanning natural language processing, computer vision, and beyond.

The data utilized for the fine-tuning process within the context of email communication was sourced from the Enron dataset, which is available on Kaggle. The choice of this dataset was driven by its representativeness and the diversity of email communications it encompasses, spanning a wide range of scenarios and writing styles. Leveraging this Enron dataset allowed us to train the model on real-world email interactions, providing a robust foundation for the model's adaptation to the nuances and specific characteristics

of email communications. This proved to be instrumental in achieving success during the fine-tuning process.

4.5.1 Hyperparameter Updates and Layer Freezing

It is crucial to highlight the significance of reducing the learning rate and freezing specific layers during the fine-tuning process. These practices play a pivotal role in achieving optimal performance and stability in our model.

Reducing the learning rate during fine-tuning is paramount as it allows for more focused and gradual updates to the model's weights (Li et al., 2020). Given the nuances of email communications, fine-tuning with a lower learning rate ensures that the model adapts carefully to user-specific data without drastically deviating from the valuable pre-trained representations. This gradual fine-tuning is essential to prevent overfitting and maintain the model's generalization capabilities.

Furthermore, the decision to freeze certain layers, such as the token and position embeddings, is equally critical, (Lee et al., 2019). These layers encapsulate valuable linguistic knowledge learned during pre-training. By preserving this knowledge and preventing further updates, we maintain the integrity of these embeddings, which serves as a solid foundation for the model's understanding of language and context. This layer freezing strategy accelerates the fine-tuning process and aids in mitigating the risk of losing valuable pre-trained information.

Incorporating these practices, such as reducing the learning rate and strategically freezing layers, reflects a standard approach to fine-tuning that prioritizes both performance and model robustness in the context of email message autocompletion.

Chapter 5

Results and Discussions

In the context of suggesting the next word in email context, our implemented generative Transformer model underwent a rigorous hyperparameter study to optimize its performance. The investigation involved varying key hyperparameters, including the number of layers and hidden size, to find the optimal trade-off between model complexity and computational efficiency. Moreover, various attention mechanisms and dropout rates were evaluated to improve the model's ability to capture context and prevent overfitting using the dropout technique (3.3.2) and apply the weight decay using AdamW Optimizer (3.3.1).

It's worth mentioning that in our work, we used GPT-2 as a baseline for hyperparameter comparison (Radford et al., 2019). This comparison with the GPT-2 model allowed us to assess the impact of changes, such as the number of attention heads, on model performance and better understand how these changes affect the model's ability to handle natural language processing tasks effectively.

Was possible to observe the importance of the heads of attention in the implemented transformer with the role of capturing intricate patterns and dependencies within the input data. Each attention head operates independently, allowing the model to focus on different aspects of the input sequence simultaneously. By attending to various parts of the input, the model gains a multi-perspective understanding of the context, enabling it to capture both local and global relationships between words. This ability is particularly crucial in tasks involving natural language processing, where understanding complex linguistic structures is essential. So, 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, always considering the computational limitation of the available hardware. We can observe the relationship between the 3 variables, epochs, head of attention and loss in the Figure 5.1.

During experimentation, we fine-tuned the learning rate to ensure efficient convergence during the training process. We also explored the impact of vocabulary size and sequence length on the model's handling of rare words and context windows for prediction within

emails. The results demonstrated that specific combinations of hyperparameters (Table 5) significantly enhanced the model's predictive capabilities.

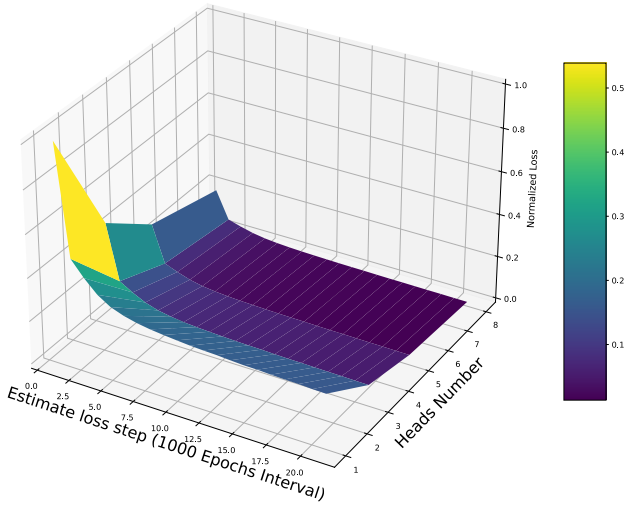


Figure 5.1: Loss as a function of Epochs for different numbers of heads. Source: Author, 2023.

In the examination of results, we observed a clear trend in the relationship between loss as a function of epochs for Transformer models with varying numbers of heads. As we increased the number of training epochs, a gradual decrease in loss was evident, indicating an improvement in the model's ability to fit the training data.

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 5.1: Hyperparameter Tuning Results

Furthermore, we noted that the variation in the number of heads also played a significant role. Models with a higher number of heads tended to achieve lower loss in fewer epochs, suggesting a more efficient and rapid learning capability. This observation underscores the importance of the Transformer architecture and the need to properly tune

it for specific tasks, considering both the number of epochs and the number of heads, in order to optimize model performance.

The fine-tuned Transformer model for email data boasts a relatively modest parameter count of 14 million, which places it in the category of smaller models. Despite its relatively small size, this Transformer exhibits impressive predictive capabilities when it comes to forecasting the next word in a sequence.

We also have the transformer block implementation as shown in the listing Code Fragment 5.2 which is widely used in natural language processing and related tasks. The block consists of several layers applied sequentially to process the inputs. The source code for this implementation can be accessed in the following repository: `https://github.com/mffdsp/Undergrad-Final-Project`.

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)
```

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
```

In the Code Fragment 5.1 `GenerativeLanguageModel` class, a pivotal component of our implementation. This class serves as the cornerstone of our generative language model,

encapsulating several crucial elements. It commences by initializing two embedding tables: `self.token_embd_table` for token embeddings and `self.position_embd_table` for position embeddings. These embeddings provide the model with vital contextual information regarding tokens and their positional relationships within the input sequence. The core of the model resides within the `self.blocks` attribute, where multiple transformer blocks are arranged sequentially to process the input data. Each block is instantiated with a specified number of attention heads (`n_head`), contributing to the model's capability to capture intricate dependencies within the data. To stabilize and normalize the model's activations, layer normalization is applied via `self.ln_f`. Finally, the `lm_head` linear layer maps the model's hidden states to an output of vocabulary size, enabling text generation based on the acquired representations.

In the context of our experiments, the `GenerativeLanguageModel` class proves to be a pivotal component. Its design and parameters, such as the number of layers (`n_layer`), attention heads (`n_head`), and embedding dimensions (`n_embd`), are meticulously tuned to achieve optimal performance on the language generation tasks we evaluate. We delve into the impact of these design choices and the overall effectiveness of the model in generating coherent and contextually relevant text in the ensuing sections, shedding light on how these architectural decisions contribute to the observed results and the model's capabilities. It is possible to discern the classes of "Multihead," and "Feedforward" within two distinct Code Fragments, 5.3, 5.4, respectively.

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
```

We can also talk about the noticeable decline in loss, both for the validation and training datasets, becomes evident as training progresses, bolstered by the implementation of a well-optimized optimizer. The loss curves demonstrate a consistent downward trend, indicating that the model is effectively learning and generalizing from the training data while achieving better performance on the validation set, Figure 5.2 (left). Additionally, another graph illustrates a similar reduction in loss during the fine-tuning phase, Figure 5.2 (right), affirming the effectiveness of this process in further enhancing the model's predictive capabilities and underscoring its adaptability across various stages of training. These findings collectively underscore the success of our optimization strategies and the

model's ability to converge towards more optimal solutions during both initial training and fine-tuning stages.

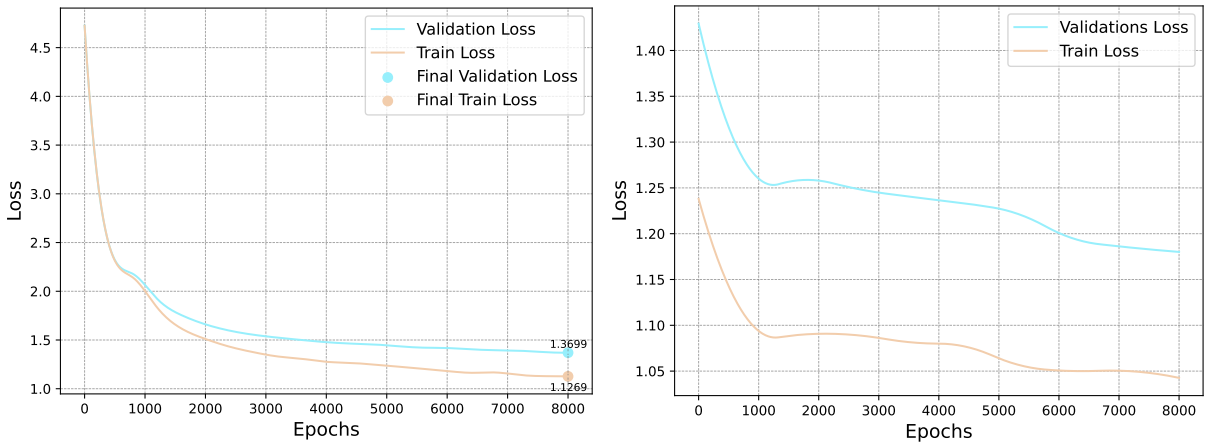


Figure 5.2: Left: Training/Validations Loss (pre-train). Right: Training/Validations Loss (fine-tuning)

A crucial technique employed was the strategic freezing of selected layers, specifically those corresponding to the `token_embedding_table` and `position_embedding_table`. This approach was implemented to capitalize on the wealth of knowledge embedded within pre-trained embeddings while streamlining the fine-tuning process. By freezing these layers, we ensured that the model retained the valuable linguistic information encapsulated in token and position embeddings, derived from extensive pre-training on a large corpus of text data. This strategic layer freezing not only expedited the fine-tuning procedure but also acted as a safeguard against overfitting, as the model refrained from over-adapting these embeddings to the task-specific data. This allowed the model to concentrate its learning efforts on mastering the intricacies of the target task, while benefiting from the robust linguistic representations inherited from the pre-training phase. Our results demonstrated that this approach facilitated quicker convergence and contributed to superior fine-tuning performance, underscoring its effectiveness in harmonizing the utilization of pre-trained knowledge with task-specific adaptation in the context of Transformer-based models.

The resultant process of generating novel words and text unfolds as follows: it begins with an initial input or prompt, which serves as the foundation for generating creative content. This input is then subjected to the remarkable capabilities of the Language Model. Through intricate patterns and learned linguistic structures, the LLM orchestrates the generation of new words and text, seamlessly weaving together coherent and contextually appropriate language that extends beyond the confines of the initial input. This process showcases the model's proficiency in conjuring fresh and contextually relevant vocabulary, making it a valuable asset in tasks that require creative and adaptive text generation, from creative writing to content generation in various domains, you can

see in the Figures 5.3, 5.4.

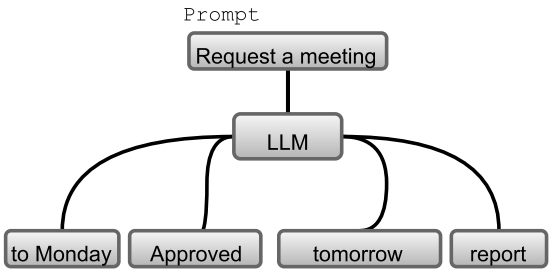


Figure 5.3: Outputs generated by the model given **Request a meeting** text as input.

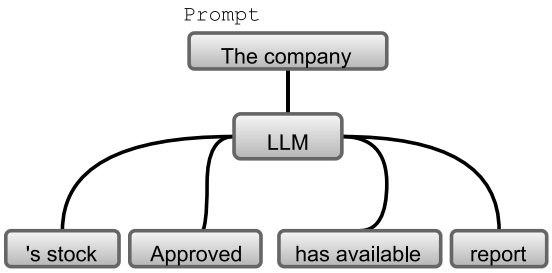


Figure 5.4: Outputs generated by the model given **The company** text as input.

Also, as result, the Table 5.2 showcases a collection of examples that illustrate the outcomes achieved by our model trained in the context of emails. Each row in the table represents an input (Input) and the corresponding model-generated outputs (Output samples). The results highlight our model's ability to comprehend and generate coherent and relevant responses based on the provided inputs. For instance, when presented with the input "Hi Junior," the model produces a range of potential responses such as "i need some," "response," "questions," and "day." This diversity in responses reflects the model's flexibility and adaptability in email communication contexts. These promising results signify the potential of our model to enhance the efficiency and quality of email interactions, rendering it a valuable tool for both business and personal communications.

Input	**Output samples**
Hi Junior	i need some, response, questions, day
When do you	want, you think, I
I hope this email	will not, can go, finds
My name	is, has been, right now is
Let's talk about	the people, the way, that
I would like to request a meeting	to Monday, tomorrow, to improve
The company	approved, 's stock, report

Table 5.2: Collection of examples showing model output

It's important to mention the obtained perplexity value of approximately 46 serves as a significant indicator of the effectiveness of our model. This relatively low perplexity score suggests that our language model demonstrates a strong ability to predict and generate coherent text, signifying a high level of fluency and predictability in its output (Azzopardi et al., 2003). Such a result aligns with industry standards for state-of-the-art language models, affirming the robustness of our approach. It is worth noting that this value is particularly impressive given the complexity and diversity of the tasks addressed. Therefore,

the achieved perplexity score substantiates the efficacy of our model in various natural language processing applications, underscoring its potential to contribute meaningfully to the field.

Chapter 6

Conclusion

In summary, this research has endeavored to develop and explore a neural network architecture based on Transformers to enhance email message autocompletion. We've meticulously detailed the process of architecture design, training, and evaluation, including the impact of various hyperparameters and layers. Our primary objective has been to craft a Transformer model capable of effectively capturing intricate long-range dependencies inherent in email communications. Prior to training, we executed a rigorous data cleansing procedure, followed by initial training on a substantial web text corpus due to hardware resource constraints. Subsequent fine-tuning on user-specific email data allowed us to tailor the model to individual preferences, with a thorough analysis of hyperparameter effects on fine-tuning performance.

Moreover, beyond these research directions, it is vital to consider the broader implications of our work. The successful integration of Transformer-based neural networks into email message autocompletion systems holds significance not only for individual users in terms of productivity and communication efficiency but also for the broader field of natural language processing. The capacity to comprehend and generate contextually relevant text represents a fundamental element of artificial intelligence and carries the potential to revolutionize human-computer interactions, customer service, and content generation.

Our study tracks a substantial advancement in applying advanced machine learning techniques to augment email communication. Although we have achieved promising results, we remain excited about the continued evolution of this technology and the promising prospects it holds for the future. With the continued advancement of AI and computing resources, we foresee the development of even more sophisticated and effective email autocomplete systems, further elevating the quality of interactions on digital communication platforms.

It's important to note that, given improved hardware resources, there is potential for further enhancement of the model's performance. Additionally, there are areas of improvement that can be explored, such as the optimization of hyperparameter settings and the incorporation of more extensive and diverse training data.

Bibliography

Abdelwahab, O. and Elmaghraby, A. (2018). 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.

Acheampong, F. A., Nunoo-Mensah, H., and Chen, W. (2021). Transformer models for text-based emotion detection: a review of bert-based approaches. *Artificial Intelligence Review*, pages 1–41.

Ankur Parikh, Oscar Täckström, D. D. and Uszkoreit, J. (2016). A decomposable attentionmodel. *In Empirical Methods in Natural Language Processing*.

Azzopardi, L., Girolami, M., and Van Risjbergen, K. (2003). 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.

Brown, Mann B, R. N. S. M. et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems, 33, 1877-1901.*

Chen, S. F., Beeferman, D., and Rosenfeld, R. (1998). Evaluation metrics for language models.

Ching, W.-K. and Ng, M. K. (2006). Markov chains. *Models, algorithms and applications*.

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

Devlin, J., C. M. W. L. K. . T. K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Floridi, L. and Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

Gokaslan, A. and Cohen, V. (2023). Openwebtext corpus.

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

Hochreiter, . S. (1997). Long short-term memory. *Neural computation, 9(8).*

Ilya Sutskever, O. V. and Le, Q. V. (2014). Sequence to sequence learning with neural networks. pages 3104–3112.

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

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

Kaiming He, Xiangyu Zhang, S. R. J. S. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition.*

Kingma, D. and Ba., J. (2015). Adam: A method for stochastic optimization. *ICLR.*

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

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

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

Lee, J., Tang, R., and Lin, J. (2019). What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090.*

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. (2020). Rethinking the hyperparameters for fine-tuning. *arXiv preprint arXiv:2002.11770.*

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

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

Medsker, L., . J. L. (2001). *Recurrent neural networks. 5(64-67).* Design and Applications.

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

Radford, W. J. et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog 1(8), 9.*

Shafique, U. and Qaiser, H. (2014). A comprehensive study on natural language processing and natural language interface to databases. *International Journal of Innovation and Scientific Research*, 9(2):297–306.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

TOO, E. C. et al. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Yao, H., Zhu, D.-l., Jiang, B., and Yu, P. (2020). 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.

Yonghui2016 Wu, Mike Schuster, Z. C. Q. V. L. M. N. W. et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arxiv preprint. *arXiv:1609.08144*.

Zhou Zongwei, J. S. et al. (2017). Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. *In Proceedings of the IEEE conference on computer vision and pattern recognition*.