PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru –560 085, Karnataka, India

A Project Report
On

Implementation of GPT models for Text Generation in Healthcare Domain

Submitted in fulfillment of the requirements for the
Project phase -1

Submitted by
Anirban Karak
SRN: PES2202100939

Under the guidance of
Kaustuv Kunal
Principal Data Scientist, Great Learning

February-April 2023

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM - M. Tech in Data science and Machine Learning

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM - M. Tech in Data science and Machine Learning

CERTIFICATE

This is to certify that the Dissertation entitled

'Implementation of GPT models for Text Generation in Healthcare Domain'

is a bonafide work carried out by

Anirban Karak
SRN: PES2202100939

In partial fulfillment for the completion of 4th semester coursework in the Program of Study MTECH in Data science and Machine Learning under rules and regulations of PES University, Bengaluru during the period Feb 2023 – Apr. 2023. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the 4th semester academic requirements in respect of project work.

Signature with date & Seal	*Signature with date & Seal*	*Signature with date & Seal*
Internal Guide	Chairperson	Dean of Faculty

Name and Signatures of the Examiners

1.

2.

3.

DECLARATION

I hereby declare that the Project Phase - 1 entitled "**Implementation of GPT models for Text Generation in Healthcare Domain**" has been carried out by me under the guidance of **Kaustuv Kunal** and submitted in partial fulfilment of the course requirements for the award of degree of **Master of Technology in Data Science and Machine Learning** of **PES University, Bengaluru** during the academic semester Feb 2023 – Apr 2023. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

SRN Number **Student Name**

Abstract:

This paper highlights the potential of using generalized language models to extract structured texts from natural language descriptions of workflows in various industries like healthcare domain. Despite the criticality of these workflows to the business, they are often not fully automated or formally specified. Instead, employees may rely on natural language documents to describe the procedures. Text generation methods offer a way to extract structured plans from these natural language documents, which can then be used by an automated system. This paper explores the effectiveness of using generalized language models, such as GPT-2, to perform text generation directly from these texts. These models have already shown success in multiple text generation tasks, and the paper's initial results suggest that they could also be effective in text generation in healthcare domain. In fact, the paper demonstrates that GPT-2 can generate results that are comparable to many current state-of-the-art text generation methods. This suggests that generalized language models have the potential to significantly improve the efficiency and accuracy of text generation, particularly in industries where workflows are repetitive and sequential.

TABLE OF CONTENTS:

LIST OF FIGURES:

LIST OF TABLES:

1. INTRODUCTION

Lifestyle diseases are continuously on the rise, due to the sedentary lifestyle of human beings, in today's world. Research and development in the medical community is progressing at a rapid rate but the number of doctors and health professionals produced is lesser compared to the rise in world's population. They can only spend a limited amount of time to diagnose the patients and often times unable to answer all the queries of the patients due to lack of time. The patients then reach out to the healthcare professionals who would benefit from having the information related to the usage of the medicine handy, to assist the patients. On knowing exactly why the medicine was prescribed, the benefits as well as the side effects of these medicines, the patients too will have a better chance of recovery.

Out of 8 billion people, 400 million people visit hospitals to get health checkup, every year. The next year when they revisit again, the healthcare professional will want to see their previous year's diagnosis. Doctors have limited time to cater to these huge population, and it will be impossible for an individual to tally the reports of health checkup of previous years (along with other issues like major operation- an individual might have undergone in previous years) in a short time, and this also has a possibility of misinterpretation of report leading to error. Thus having the key information regarding the medicine prescribed in the last year along with the usage, will greatly help in time saving.

To overcome this issue, in this paper, I tried to explore the possibility that given a medicine name, the ability of GPT-2 to correctly interpret the data and provide answers regarding the usage of this medicine. The focus of this research paper is to explore the potential of utilizing GPT-2, a cutting-edge transformer-based language model, for generating structured text from natural language texts. Through investigations, I have discovered that these models are able to achieve comparable, and in some cases superior scores, in comparison to previous methods.

To test the efficacy of GPT-2 in this context, I have analyzed natural language texts from PubMed dataset and evaluated the model's performance based on its Rouge score - a widely used quantitative measure for this task[1]. Furthermore, I have compared the results with those previously published for text generation models, which utilize a range of techniques such as reinforcement learning and deep learning. The findings suggest that GPT-2 holds immense promise as a powerful tool for generating structured texts from natural language text prompts.

A background on the project is provided in Section 1.1. A concise problem statement is described in Section 1.2, Section 1.3 introduces the materials and methods used in this study.

1.1 Background

As natural language computer applications continue to advance, the potential for their use in healthcare-related contexts grows, especially with the release of the Generative Pre-trained Transformer-2. However, it is important to carefully consider the benefits and drawbacks of such technology before deploying it in areas that have traditionally relied on human-to-human interaction. While the idea of such technology being used in healthcare settings may sound futuristic, it was not too long ago that the idea of a powerful computer that could fit in the palm of hand seemed fantastical. GPT-2 and other similar technologies are getting closer to passing the Turing Test, which evaluates whether their language is indistinguishable from that produced by humans. The possibility of this technology being used in healthcare has generated both excitement and caution. However, as with any new technology, there are unrealistic, realistic, and challenging-but-possibly-unwise applications and developments that need to be considered before implementing GPT-2 in eHealth.

The potential of AI-powered natural language processing (NLP) applications in healthcare is vast, including the ability to navigate complex electronic health record (EHR) systems, automate documentation with human review, prepare orders, and perform other routine tasks. However, unlike a traditional chatbot, these applications are capable of more complex text conversations, allowing for more natural-sounding question-and-answer exchanges. This increased complexity could lead to more personalized experiences for patients in non-critical healthcare encounters, such as online chat support or assistance with telehealth equipment setup. Interestingly, the developer of one such application, OpenAI, originally intended it for use by companies and organizations to improve customer service chatbots or perform similar tasks.

Transparency is also critical for datasets and to disclose the limitations in language training activities. A GPT-2 application will need to be given conversation endpoints so that it leads the inputs rather than having the healthcare professional control the focus of the interaction, for form-completion tasks, it will also need additional guidance to determine whether the information a patient shares addresses the relevant usage. IT support personnel, or those in similar roles, will need to learn how to shape the inputs that will deliver the most relevant answers or results from a given interaction. For GPT-2 priming using few-shot learning, a commitment to transparency would require publishing any customized parameters. In high-risk applications in healthcare, including any doctor-facing tools, such sharing must be mandatory.

1.2 The Problem Statement

Healthcare professionals sometimes face challenge to provide accurate information to patients, in absence of a trained professional like doctors. Remembering the usage of all the prescribed medicines along with their benefits and side effects is a challenging task. GPT-2 might be useful in these scenarios, as it has proven to be useful in abstractive text generation. In this paper, I

explored the possibility that when medicine name is passed to GPT-2's prompt, it will be able to summarize and generate text-based output regarding detailed description of the usage of the medicine. This will resolve manual retrieval challenges as well as the accuracy of the model will not be compromised[2]. Incorrect diagnosis or delay in treatment will be overcome by GPT-2's text generation.

1.3 Text Generation

Text generation is a fascinating field of natural language processing (NLP) that involves creating new text based on a given input. This technology uses advanced algorithms to generate text that is often indistinguishable from human-written content. As a type of artificial intelligence (AI), text generation can produce coherent and grammatically correct sentences that are comparable to what a human would write. The applications of text generation are varied and diverse. For example, it can be used for summarization tasks, where it generates concise and meaningful summaries of lengthy documents or articles. It can also be utilized in dialogue systems to create engaging and interactive conversations with users. Furthermore, text generation is an essential component of machine translation, where it helps to convert written text from one language to another accurately. Overall, text generation is a powerful and innovative technology that is changing the way we interact with language. It has the potential to revolutionize various industries, including publishing, advertising, and customer service. As AI continues to evolve, we can expect text generation to become even more sophisticated and ubiquitous in our daily lives.

1.3.1 Process of Text Generation

Text generation is a sophisticated process that combines machine learning algorithms and natural language processing techniques[3]. To generate text, these algorithms are trained on massive collections of text, such as books, articles, and other relevant sources. As the algorithms learn, they can identify patterns and relationships within the data, which they then use to generate new text based on a given input. Text generation is a complex process that typically involves two main steps:

1. The first step involves taking the input text and creating a representation of it. This representation serves as the foundation for generating the output text. The algorithm analyzes the input text and identifies key themes, concepts, and patterns that are then used to construct the output.
2. The second step of the process involves evaluating and improving upon the generated text. This is done using a range of techniques, such as grammar and syntax checking,

spell checking, and other methods. The goal is to ensure that the output text is not only accurate but also well-written, concise, and easy to understand.

1.3.2 Benefits of Text generation

Text generation is a powerful tool that has a wide range of applications. It can improve various tasks, including summarization, dialogue systems, and machine translation. Additionally, it is an effective way to generate content for websites, blogs, and other online sources.

- One of the most exciting aspects of text generation is its ability to create personalized content for customers. By tailoring the content to the customer's interests, text generation can significantly improve engagement and customer experience.
- In addition to creating content, text generation can also be used to generate data for research and analysis. This data can provide valuable insights into customer behavior, market trends, and other important data points. By using text generation to analyze this data, businesses can make informed decisions and stay ahead of their competitors.
- In healthcare domain, via text generation we can save time of medical professionals as they will not have to be involved in mundane tasks of checking tons of data manually. Since it will be automated with the help of GPT-2, the answers to all their queries will be readily available.

1.3.3 Challenges of Text Generation in healthcare domain

Text generation is a challenging task that involves numerous complexities. Some of the main challenges include:

- Firstly, generating text that is both grammatically correct and coherent can be difficult. To achieve this, algorithms must be able to analyze the syntax and structure of the input text, as well as understand the context in which it is being used.
- Secondly, generating text that is natural and human-like requires a deep understanding of the nuances of language. Algorithms must be able to recognize idioms, figures of speech, and other linguistic nuances to generate text that sounds natural and realistic.
- Thirdly, generating text that is relevant to the given input requires a thorough analysis of the text and an understanding of the intended audience. The algorithms must be able to identify the main themes and concepts in the input text and use this information to generate relevant and useful output.

- Finally, generating text that is accurate and consistent is crucial. Algorithms must be able to generate output that is free from errors and inconsistencies to ensure that the resulting text is both useful and trustworthy.

1.4 Proposed Solution

The methodology for Implementation of GPT2 models for Text Generation in Healthcare Domain proposed in this thesis works in several stages, and the mechanism used is input data to GPT-2 for fine tuning with Adam Optimizer.

From healthcare domain, PubMed (medicine) raw data was collected, which was preprocessed by different preprocessing techniques such as null removal, duplicate removal, special character removal, punctuation removal emoji removal etc. Next, tokenization had been performed on to the preprocessed data. By using random splitting, I created the training and validation dataset from the tokenized dataset also created the segment tokens for the splitted datasets[4]. A special token called (<endoftext>) was added after each of the line segments and then added the padding. Each segment tokens have been shuffled for better result and also created the tensor object for GPT-2 model. GPT-2 tokenizer was used for tokens embedding. I trained the GPT-2 model on the tokenized dataset and also performed the hyperparameter tuning. The GPT-2 model takes the input text such as medicine name and generates the text based on the usage of the medicine. The model summarized the data and predicted Boolean value as an output.

On testing with actual data, it was observed that the prediction was close to gold summary. For cases, where the model failed to predict, the model could be fine-tuned further with larger sample of data and need to be retested and the accuracy need to be measured.

2. LITERATURE SURVEY

GPT-2 is a transformer model that has been pretrained on a vast amount of English data in a self-supervised manner, meaning that it has learned from the raw texts without the need for human labeling. This approach allows the model to leverage publicly available data and generate inputs and labels through an automated process that involves predicting the next word in a given sentence. The model takes in sequences of text of a particular length as inputs and generates the same sequence as targets, with each token shifted by one to the right. To ensure that the predictions for a token only rely on its preceding inputs and not future tokens, the model uses an internal masking mechanism. Through this training process, the model has developed an internal representation of the English language that can be used to extract useful features for various downstream tasks. However, the model performs best at the task for which it was pretrained, which is generating text from a given prompt. Previous work on text generation from descriptions has revolved around specific models for action extraction, some of them trained on largely task-specific preprocessed data and use sequence-to-sequence models and inverse reinforcement learning to generate instructions from natural language corpora. Similarly, a reinforcement learning model is used to extract word actions directly from free text (i.e., the set of possible actions is not provided in advance) where, within the RL framework, actions select or eliminate words in the text and states represent the text associated with them[5]. This allows them to learn the policy of extracting actions and plans from labeled text. In a same fashion, Reinforcement Learning was used as a policy gradient algorithm and a log-linear model to predict, construct, and ultimately learn the sequence of actions from text. Other works define a system of tools through which they crawl, extract and denoise data from plan-rich websites and parse their actions and respective arguments with statistical correlation tools to acquire domain knowledge. However, to the best of my knowledge this paper is the first work to assess the performance of a general-purpose NLP language model on text generation tasks in healthcare domain where in the model can quickly summarize and generate output.

2.1 Evolution of GPT

Currently, there are vast quantities of textual data available, in healthcare domain, that contains long strings of text that need to be summarized. The importance of text generation is due to several reasons, including the retrieval of significant information from a long text within a short period, easy and rapid loading of the most important information, and resolution of the problems associated with the criteria needed for summary evaluation. Due to the evolution and growth of automatic text generation methods, which have provided significant results in many languages, these methods need to be reviewed and summarized. Applications such as search engines and news websites use text generation.

Text generation can be divided into several categories based on function, genre, summary context, type of generation, and number of documents. Deep learning analyses complex problems to facilitate the decision-making process. Deep learning attempts to imitate what the human brain can achieve by extracting features at different levels of abstraction. Typically, higher-level layers have fewer details than lower-level layers. The output layer will produce an output by nonlinearly transforming the input from the input layer. The hierarchical structure of deep learning can support learning. The level of abstraction of a certain layer will determine the level of abstraction of the next layer since the output of one layer will be the input of the next layer. In addition, the number of layers determines the deepness, which affects the level of learning.

Machine translation systems rely on language models to learn how to convert strings from one language to another. In modern GPT systems, Transformers play a vital role[6]. These models take input sentences from users and generate an intermediate representation (IR) that captures their syntactic structure but not their meaning. Next, the IR goes through a sequence of linear algebra operations (referred to as feature engineering) before being inputted into a neural network for translation into another language via backpropagation through time (BPTT).

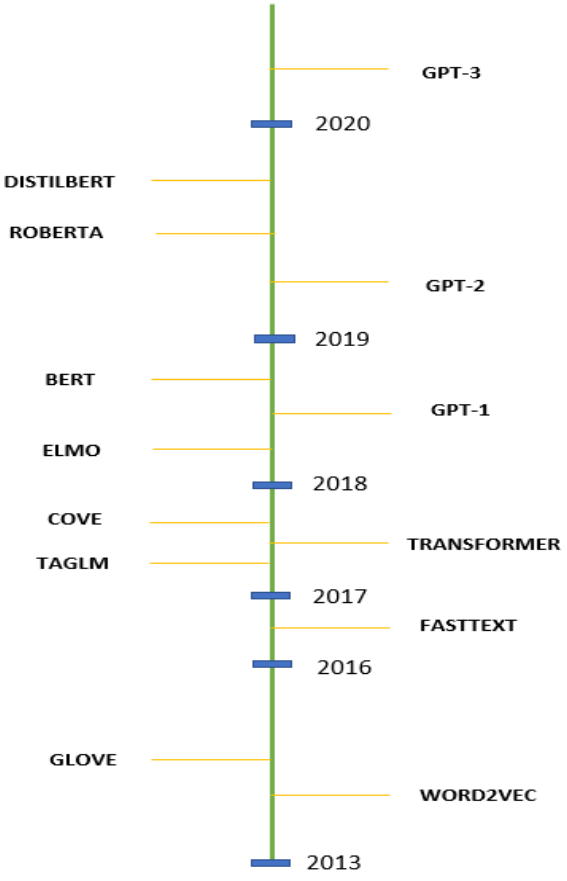


Fig 2.1- Evolution of GPT

2.1.1 History of GPT:

Established in 2015 as a non-profit organization, OpenAI initiated the development of GPT as part of its research agenda to promote and create "friendly AI" that would serve the betterment of humanity. The model was first introduced in 2018, containing 117 million parameters. The following year, OpenAI released an even more sophisticated version, GPT-2, featuring 1.5 billion parameters. In comparison, GPT-3, the latest version, boasts a massive 175 billion parameters, surpassing its predecessor by more than 100 times and outperforming other similar programs by ten times.

Earlier models, including BERT, had demonstrated the potential of the text generator approach, showcasing the remarkable capabilities of neural networks[7] in producing extensive pieces of text, which were previously deemed impossible.

To prevent any potential issues, OpenAI cautiously released access to the model in increments to observe its usage. During the beta phase, users had to apply to use the model, and access was initially free of charge. However, the beta phase ended in October 2020, and OpenAI launched a tiered credit-based pricing model, ranging from free access to 100,000 credits or three months of access to more substantial charges of hundreds of dollars per month for more significant access.

In the same year, Microsoft invested $1 billion in OpenAI, becoming the sole licensee of the GPT-3 model's underlying technology. This means that Microsoft has exclusive access to GPT-3's underlying model. This partnership provides Microsoft with the opportunity to integrate GPT-3 into its own products and services.

In November 2022, ChatGPT was launched, and it was initially free for public use during its research phase. This launch brought GPT-3 more mainstream attention, providing many non-technical users with the opportunity to try the technology.

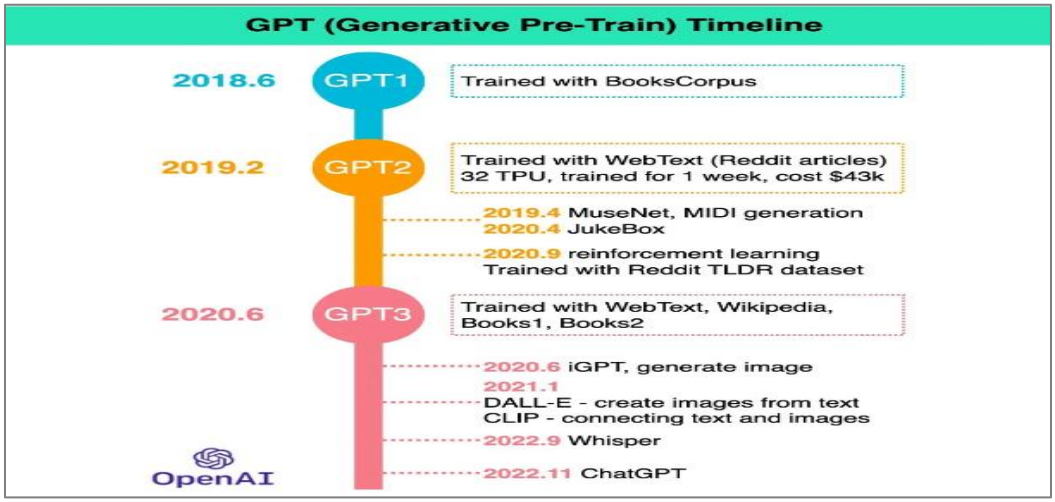


Fig 2.2- History of GPT *Source:* OpenAI

2.1.2 Future of GPT:

As GPT-3 continues to gain popularity, researchers and engineers are already working on even more advanced models. Some open-source efforts aim to provide a non-licensed alternative to Microsoft's exclusive ownership of the GPT-3 model. OpenAI is also planning to release more specialized versions of their models trained on a wider variety of texts.

Different use cases and applications of the GPT-3 model are also being explored. However, Microsoft's exclusive license presents challenges for developers who want to integrate the capabilities of GPT-3 into their applications[8]. Despite this, Microsoft has expressed interest in incorporating a version of ChatGPT into popular applications like Word, PowerPoint, and Power Apps.

It remains uncertain how GPT-3 will evolve in the future, but it is likely that it will continue to be used in various generative AI applications. Experts predict that there will be exponential technical advancements in the generative AI space, with continued investment from tech giants such as Microsoft, Google, Apple, and Nvidia.

GPT-4 is likely to be released in 2023, and the model will be trained on 100 trillion parameters and has the potential to surpass all GPT models that exist today.

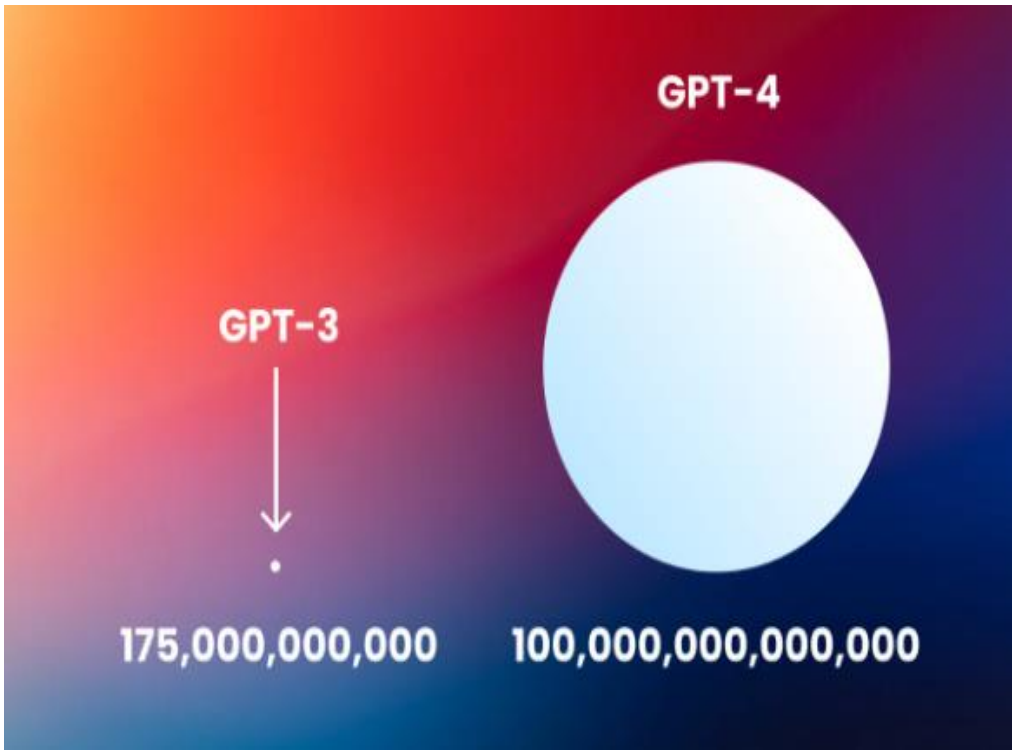


Fig 2.3- Future of GPT

Source: OpenAI

2.2 GPT models

The OpenAI team has developed a series of deep learning-based language models called GPT models. These models can perform several NLP tasks, such as question-answering, textual entailment, and text summarization, among others, without supervision. Moreover, these language models can accomplish these tasks with very few or even no examples. Surprisingly, they often perform as well as, if not better than, state-of-the-art models trained with supervision.

2.2.1 GPT - 1

OpenAI' s researchers unveiled a groundbreaking approach to natural language understanding (NLU) in 2018. Their framework involved a single model that can handle multiple tasks, achieved through a combination of generative pre-training and discriminative fine-tuning. This innovative approach had the potential to revolutionize the field of NLU and bring us one step closer to machines that can truly understand language.GPT-1 made history by becoming the first model to read text and provide answers to questions about it. Its training data consisted of Wikipedia articles, which helped GPT-1 acquire a human-like language proficiency for answering questions. The advent of GPT-1 marked a significant milestone in AI development as it enabled computers to grasp textual information in a more intuitive manner than previous models. Nevertheless, despite its groundbreaking capabilities, GPT-1 has some limitations that restrict its scope of application.

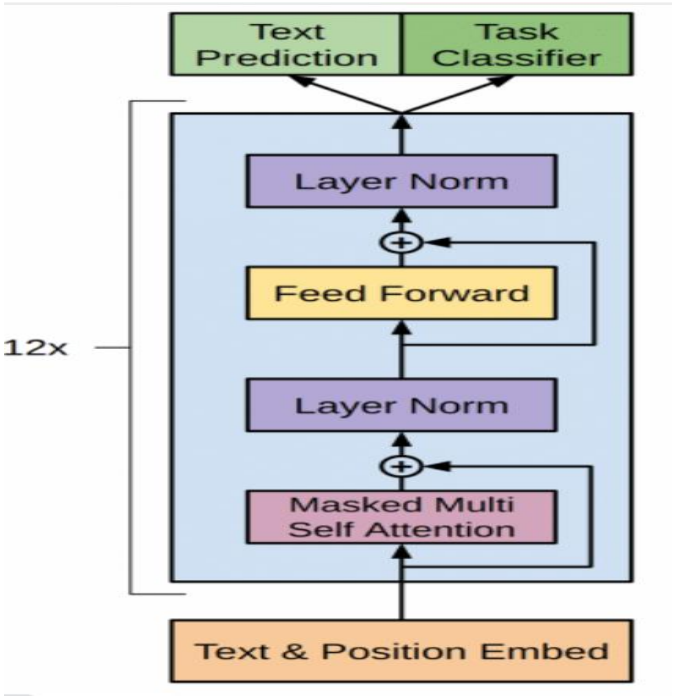


Fig 2.4 – Architecture of GPT-1 model

Source: ResearchGate

2.2.2 GPT - 2

GPT-2 is a powerful transformers model that has been pre-trained on a massive corpus of English data in a self-supervised manner. Unlike supervised learning where humans annotate the data, GPT-2 learned from raw text inputs without any human intervention. This allowed the model to utilize vast amounts of publicly available data for its training. During pre-training, GPT-2 was trained to predict the next word in each sentence, which required it to learn the underlying patterns and structures of the English language.

The model operates by taking sequences of text of a specific length as inputs, and predicting the same sequence shifted by one token to the right as the target. The model uses a masking mechanism internally to ensure that it only uses the inputs from the past tokens for each prediction.

Thanks to its pre-training, GPT-2 has acquired a comprehensive understanding of the English language that can be leveraged to extract features useful for various downstream tasks. However, it excels at the task it was specifically pre-trained for, which is generating coherent and fluent texts based on a given prompt.

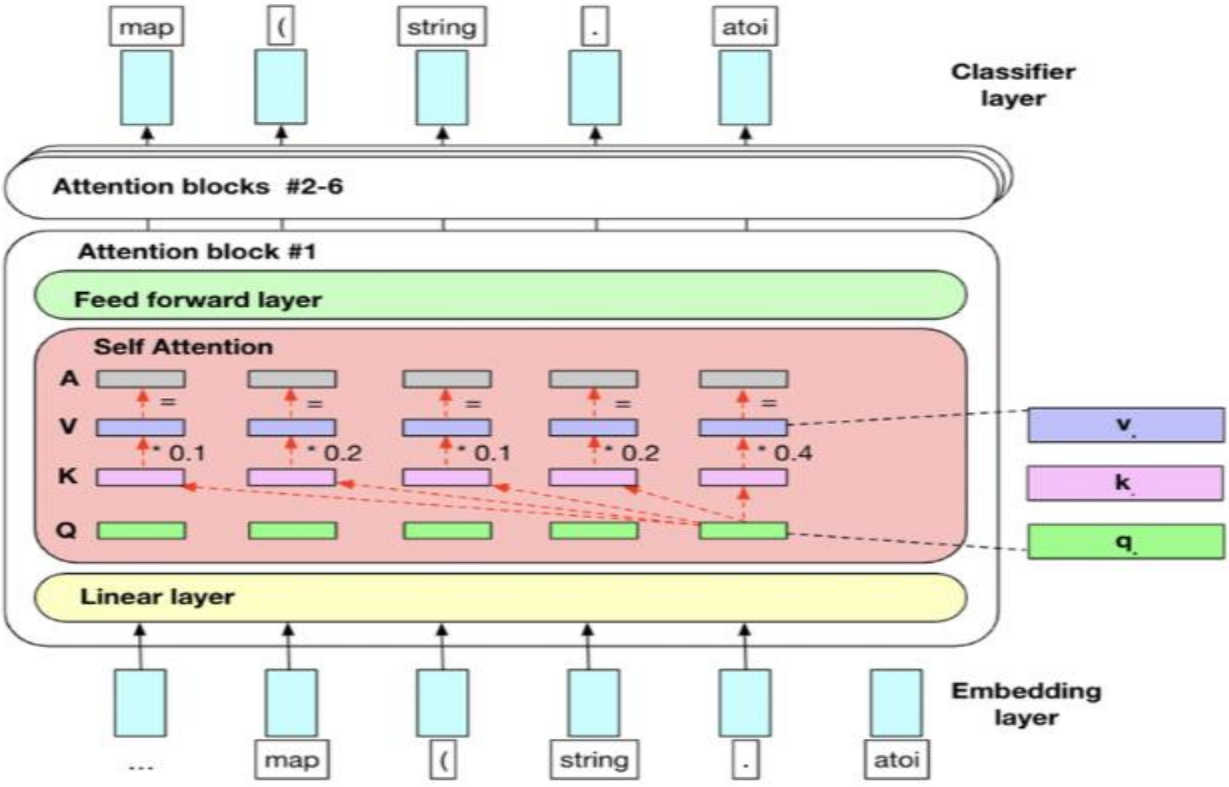


Fig 2.5– Architecture of GPT-2 model

Source: ResearchGate

2.2.3 GPT - 3

In 2020, the world witnessed the release of Generative Pre-trained Transformer 3 (GPT-3), a remarkable autoregressive language model that uses deep learning to generate human-like text. By feeding it an initial text prompt, the model can produce a continuation of that prompt that appears as if it were written by a human.

GPT-3 has an architecture that consists of a decoder-only transformer network with an impressive context of 2048 tokens, making it one of the largest language models to date with 175 billion parameters. To achieve this size, the model requires a storage capacity of 800GB. GPT-3's training utilized generative pre-training, where it was trained to predict the next token in a sequence based on the previous tokens. This approach enabled the model to demonstrate strong zero-shot and few-shot learning on various tasks.

The creators of GPT-3 described how the model improved the performances of natural language processing (NLP) in language understanding by employing a process of "generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task." By doing so, it eliminated the need for human supervision and time-consuming hand-labeling, paving the way for more efficient and effective NLP.

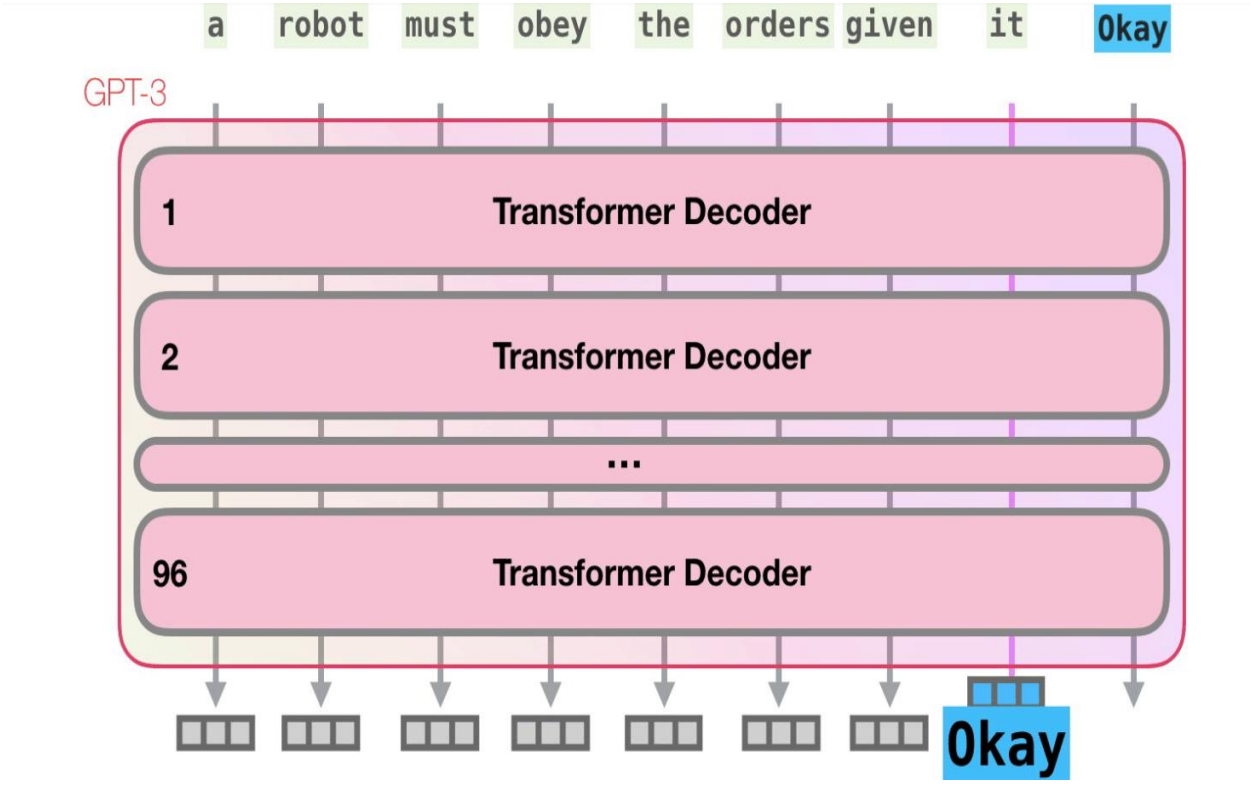


Fig 2.6 – Architecture of GPT-3 model

Source: ResearchGate

2.3 Comparison of GPT models

With the evolution of GPT, we see that each model performs better than its predecessor, due to its training on larger data. A brief comparison can be seen as below:

	GPT-1	GPT-2	GPT-3
No. of decoder blocks	12	48	96
Tokens	512	1024	2048
Corpus for training	Books	webText	WebText + books
Size of corpus	5GB	40GB	600GB
No. of parameters	117M	1.5B	175B

Table 2.1 – Comparison of GPT 1 vs GPT 2 vs GPT 3

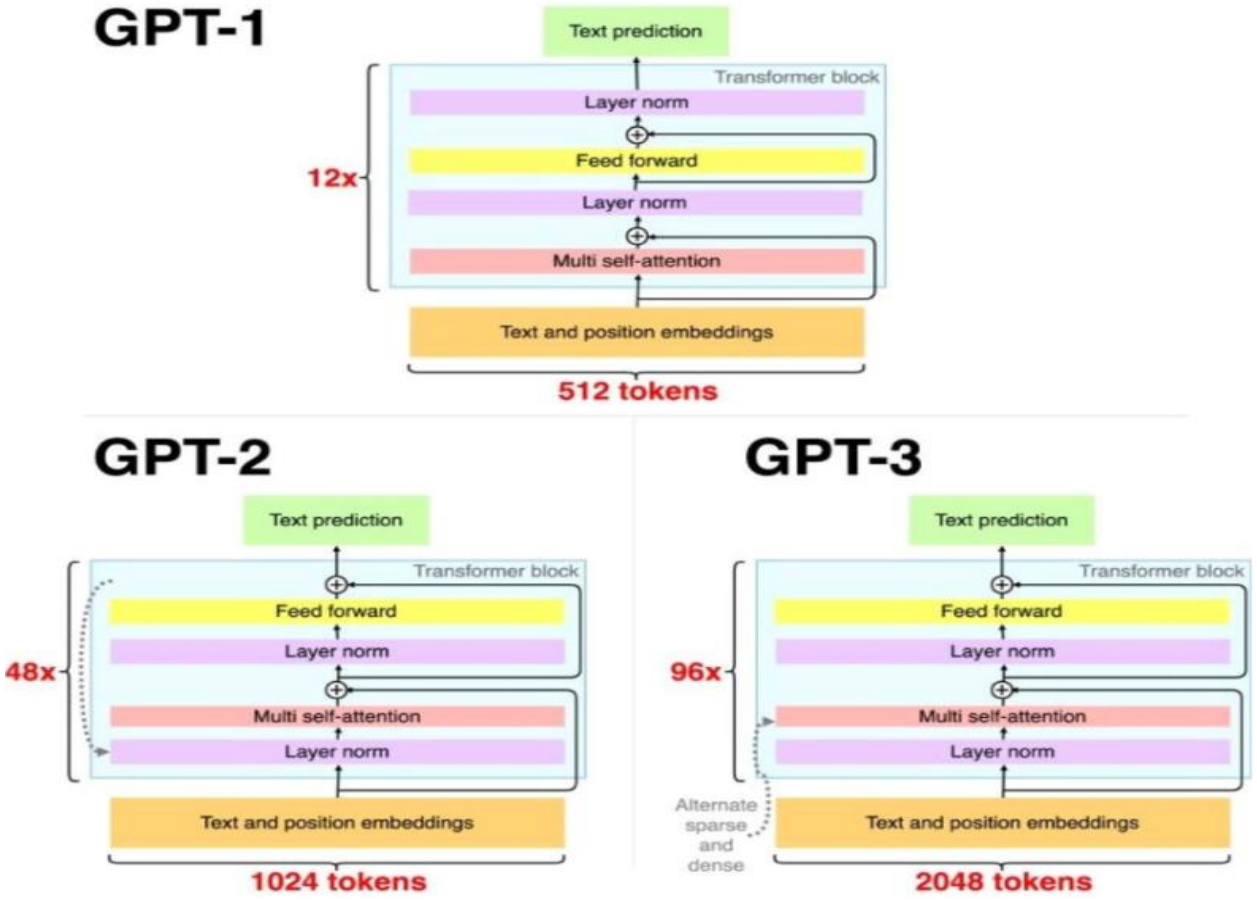


Fig 2.7 – Comparison of GPT models architecture

Source: ResearchGate

2.4 Analysis of GPT-2

The analysis of GPT-2 can be broken down into several different areas, including:

- Performance on language tasks: GPT-2 has been shown to perform exceptionally well on a variety of natural language processing tasks, such as language translation, language modeling, question answering, summarization, and more. The model's impressive performance can be attributed to its large size and training on a diverse and extensive dataset[9].
- Bias and ethical concerns: Because GPT-2 is trained on a vast dataset of web pages, it may inadvertently pick up and reproduce biases present in the source material. There are also concerns about the potential for malicious use of the model, such as generating realistic fake news, propaganda, or other forms of disinformation.
- Interpretability: GPT-2's inner workings can be difficult to interpret, partly because of the complex nature of the model itself and partly because of its size. However, researchers are exploring ways to increase the interpretability of GPT-2 and other large neural language models.

The improvements in GPT-2 are summarized as below:

1.Learning Objectives:

- **In-Context learning -** Language models with large parameter sizes learn to recognize patterns and develop other skills using the text data they are trained on. While their primary objective is to predict the next word based on the context words, these models also begin to identify patterns in the data that help them minimize the loss for the language modeling task. This ability becomes particularly valuable when the model is faced with zero-shot task transfer, where it is presented with only a few examples or a description of what it needs to do. In such cases, the language model matches the pattern of the examples with what it has learned in the past for similar data and uses that knowledge to perform the tasks. This is a powerful capability of large language models, which increases as the number of parameters in the model grows.
- **Few shot, zero shot and One-shot Learning -** The few-shot setting involves providing the model with a task description and as many examples as will fit into the context window of the model. In the one-shot setting, the model is provided with exactly one example, while in the zero-shot setting, no example is provided. As the capacity of the model increases, its few-shot, one-shot, and zero-shot capabilities also improve.

2. Dataset:

GPT-2 is a language model based on a transformer architecture, boasting 1.5 billion parameters, and trained on a large dataset of 8 million web pages. The model's objective is straightforward: predict the next word based on the preceding words in a given text. The dataset used for training is diverse and includes examples of various tasks in different domains, making the model capable of handling a broad range of natural language processing tasks.GPT-2 is a more extensive version of its predecessor, GPT, with over 10 times the number of parameters and trained on more than 10 times the amount of data. This large-scale training has resulted in a model that can generate high-quality text, complete tasks with remarkable accuracy, and adapt well to various downstream tasks.

3. Implementation Details:

GPT-2 (Generative Pre-trained Transformer 2) is a language model developed by OpenAI in 2019. It is based on the transformer architecture, which is a type of neural network that was introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017.Here are some key implementation details of GPT-2:

1. **Transformer architecture:** GPT-2 is based on the transformer architecture, which consists of an encoder and a decoder. The encoder processes the input text and creates a representation of it, while the decoder generates the output text based on that representation.
2. **Pre-training:** GPT-2 is a pre-trained language model, which means it was trained on a large amount of text data before being fine-tuned on a specific task. The pre-training was done using a variant of the unsupervised learning technique called self-supervised learning, where the model tries to predict the next word in a sequence of text[10].
3. **Dataset:** GPT-2 was trained on a large and diverse dataset of text called the WebText dataset. This dataset consists of over 45 terabytes of text from web pages, books, and other sources.
4. **Architecture details:** GPT-2 has 1.5 billion parameters, making it one of the largest language models at the time of its release. It has 48 layers of transformers, with each layer having 16 attention heads. The model also uses a technique called "layer normalization" to normalize the output of each layer.
5. **Fine-tuning:** After pre-training, GPT-2 can be fine-tuned on a specific task, such as text classification or language generation. Fine-tuning involves training the model on a smaller dataset of task-specific examples.
6. **Decoding strategy:** GPT-2 uses a beam search algorithm to generate text. Beam search is a technique used in natural language processing to find the most likely sequence of words given a set of probabilities.

4. Performance:

GPT-2 (Generative Pre-trained Transformer 2) is a language model developed by OpenAI in 2019. It is one of the largest and most powerful language models available, with 1.5 billion parameters. Here are some key performance metrics of GPT-2:

1. **Language modeling:** GPT-2 achieved state-of-the-art results on a number of languages modeling benchmarks, including the One Billion Word Benchmark and the WikiText-103 dataset. It also set new records on the LAMBADA and ReCoRD tasks.
2. **Text generation:** GPT-2 is capable of generating high-quality natural language text. In fact, it was so good at generating text that OpenAI decided not to release the full version of the model due to concerns about its potential misuse.
3. **Zero-shot learning:** GPT-2 is also capable of zero-shot learning, which means it can perform a task without any additional training. For example, it can answer questions, translate text, or summarize articles without being specifically trained on those tasks.
4. **Fine-tuning:** GPT-2 can be fine-tuned on a specific task, such as text classification or language generation. Fine-tuning involves training the model on a smaller dataset of task-specific examples. In many cases, GPT-2 achieved state-of-the-art performance on these tasks after fine-tuning.
5. **Comprehension:** GPT-2 is capable of understanding and generating complex language, including idiomatic expressions, metaphors, and wordplay.

5. Applications:

The applications of GPT-2 are as below:

- **Text generation:** GPT-2 is capable of generating high-quality natural language text, making it useful for a wide range of applications. It can be used to generate text for chatbots, virtual assistants, automated news articles, and more.
- **Text completion:** GPT-2 can be used to complete text, making it useful for applications such as predictive typing, autocomplete, and grammar correction.
- **Content creation:** GPT-2 can be used to create content for marketing campaigns, social media posts, and other forms of digital content.
- **Natural language processing research:** GPT-2 has been used extensively in natural language processing research to advance the field. Researchers have used GPT-2 to study language modeling, transfer learning, and other topics in NLP.
- **Text classification:** GPT-2 can be fine-tuned for text classification tasks such as sentiment analysis, spam detection, and topic classification.
- **Question answering:** GPT-2 can be fine-tuned for question answering tasks, making it useful for applications such as chatbots and virtual assistants.

6. Limitations:

As an AI language model, GPT-2 has some limitations, some of which include:

- Limited Contextual Understanding: GPT-2 can only understand the context of a sentence or paragraph to a certain extent, and it may not be able to capture complex relationships between concepts in the text.
- Limited Domain-Specific Knowledge: While GPT-2 has been trained on a large corpus of text, it may not have knowledge of specific domains such as medicine, law, or engineering.
- Biased Outputs: GPT-2 can generate biased outputs, reflecting the biases inherent in the training data it was trained on. This can lead to discriminatory or harmful results.
- High Computing Power: Training and fine-tuning GPT-2 requires significant computing resources, making it difficult for researchers and individuals with limited resources to use the model.
- Limited Multilingual Support: GPT-2 has been trained on a large amount of English language data, but it may not perform as well in other languages or may require more data and training to achieve comparable performance.

2.5 Summary

While GPT-2 is indeed a significant improvement over its predecessors, it is important to note that the model still has limitations and areas for improvement. For example, it is known to sometimes produce outputs that are factually incorrect or nonsensical, and it can perpetuate biases that exist in the data it was trained on.

While there are concerns about potential misuse of the model, OpenAI has implemented measures such as limiting access and monitoring usage to help mitigate these risks. It is likely that as the technology advances and more powerful models are developed, these types of concerns will become even more pressing and require ongoing attention and vigilance from developers and users alike. While it is true that AI language models like GPT-2 still have limitations in accurately generating human-like text, it is also worth noting that they have made significant strides in recent years.

There is still much work to be done in the field of AI language generation. Challenges such as dealing with bias and improving contextual understanding continue to be areas of focus for researchers. Additionally, while the rapid advancement in GPT models is promising, it is important to note that breakthroughs may not happen as quickly or as easily as some may hope. Overall, while AI language generation may not yet be at the point of dealing a significant blow to human language, the field is advancing quickly and holds great promise for the future. In this paper, I will try to leverage GPT-2 for text generation in healthcare domain, keeping all the benefits and risks in mind and will try to find ways to optimize the model performance.

3. SYSTEM REQUIREMENT SPECIFICATIONS

The requirements that are fulfilled by the project are discussed in detail, in this section. The perspective endeavored by the project, the features it is intended to offer, and the scope of the project is discussed in the upcoming sections. The Environment in which the product is executed, the assumptions and dependencies involved along with the testing requirements are discussed below. This chapter outlines the expectations of the project and the elements required to fulfil these expectations.

3.1 Project Scope

The population of the world is increasing with every passing day and due to the sedentary lifestyle (a lifestyle when someone spends six or more hours per day sitting or lying down, and they lack significant physical movement in their daily life), many lifestyle diseases are prevalent in young adults today like diabetes, cancer etc. Whereas the older generation suffer from health-related ailments like arthritis, dementia etc. The infants suffer from auto immune diseases, virus, and bacteria related diseases very frequently due to increase in pollution and overall lack of hygiene.

While the diseases are increasing, simultaneously the improvements and success of research in healthcare is also expanding, and people nowadays are getting more conscious due to increased knowledge and education about the symptoms. People prefer to get tested once a year and seeing the business prospect, many corporate held hospitals are providing lucrative offers to the multi-national companies for its employees as well as their families for a discounted price.

Since everything has its pros and cons, similarly with the awareness mentioned above, the number of health checkups done around the world are significantly increasing, and many life threatening diseases are getting diagnosed and prevented at nascent stages, but since the number of doctors and nurses are not increasing at a similar rate, it becomes too difficult for them to deal with this huge amount of data, because when the patient comes for a health checkup for second year onwards, the healthcare professionals need to also look into their reports from the previous year. Now, imagine if a family of four, getting health checkup for five consecutive years, and for single member there is a report of approx. 30 pages per health checkup. For consecutive 5 years it will be 30*5 =150 pages and for a family of 4, it will be 600 pages of data, that needs to be checked before their sixth annual checkup. This number multiplied by $^{1}/_{20}{}^{th}$ of the world's population can easily become a nightmare for the limited number of healthcare professional that we have, which is 1 doctor per 250 patients in current scenario.The only way to handle this issue with ease is to automate the process, which is where this paper comes in. Given a medicine name as an input, the GPT-2 model will scan through the data, process the data (keeping only relevant parts and removal of punctuations, emojis, stop

words, special characters etc). Next, tokenization had been performed on to the preprocessed data. By using random splitting, I created the training and validation dataset from the tokenized dataset also created the segment tokens for the splitted datasets. A special token called (<endoftext>) was added after each of the line segments and then added the padding. Each segment tokens have been shuffled for better result and also created the tensor object for GPT-2 model. GPT-2 tokenizer was used for tokens embedding. I trained the GPT-2 model on the tokenized dataset and also performed the hyperparameter tuning. The GPT-2 model takes the input text such as medicine name and generates the text based on the usage of the medicine. Additional benefit of this process will be that only the model needs the latest data of the current year, and it will not need to be trained in previous years data, as training had already been done on them. So, even with the increase in data, the performance of the model will not be challenged.

3.2 Project Perspective

This section explains the approach undertaken to fulfil the objectives of the project. As previously discussed in section 2, several approaches have already been tested for the text generation and the perks of the same are compared. After analyzing the various approaches available in the literature survey, which were fruitful yet could be more effective and efficient, I propose a perspective which uses the already available solutions with a humble yet valuable additional strategy, to make prompt generation more accurate but in lesser time.

3.2.1 Project Features

The major feature offered by the project is generation of output from large amount of medical data in a significantly less time. The project can only be executed on medical data, the accuracy might be challenged if done on data from other domains. Given a prompt with relevant context, the model will be able to generate response to the questions asked. The model needs to be trained whenever new data comes in, via a scheduler.

3.2.2 Assumptions, Dependencies and Constraints

The assumptions, dependencies and constraints set place in the development of the system are listed below:

1. Samples were collected from PubMed site https://pubmed.ncbi.nlm.nih.gov/download/ The database is maintained by NIH (National Institute of Health) also approved by FDA.
2. The training requires a large and diverse set of data, and the processing of this large and diverse set needs reasonable processor speed and RAM. The cloud-based solutions such as Google Collab Pro cater to the needs of developing this model, while the traditional computing may not be very flexible.

3. Since this research involves handling of medical data that are private and confidential, it is necessary to handle the data with utmost care to prevent unethical use.
4. One constraint of GPT-2 is that the token limit is 1024. We cannot pass large sentence.

3.3 Functional Requirements:

The basic functionalities expected of the project are listed below:

1. As a user, when I expect the model to generate an output, I assume that the model has already been pre-trained with a prompt and a context.
2. As a user, when I expect the model to predict with higher accuracy, I assume that the model has been pre-trained with large amount of medical corpus.
3. As a user, when I expect the model to generate text for a different medicine , I want to avoid bias, so I assume while training, the model was pre-trained on different variety of medicine data.
4. As a user, when I ask about a medicine to the model, I expect the model to generate a sentence with minimal of 30 words.

3.4 Non-Functional Requirements:

Non-functional requirements are the quality attributes or characteristics of a system that describe how well it performs its functions. In the case of GPT-2, some non-functional requirements may include:

1. **Performance:** GPT-2 should be able to generate text quickly and efficiently, with minimal latency and processing time.
2. **Scalability:** The model should be able to handle a large amount of data and be scalable to accommodate increasing amounts of data in the future.
3. **Reliability:** GPT-2 should be reliable and available for use when needed. It should have a high uptime and minimal downtime.
4. **Security:** The model should be secure and protect user data and information from unauthorized access or breaches.
5. **Maintainability:** The model should be easy to maintain and upgrade, with minimal disruption to ongoing operations.
6. **Usability:** The model should be easy to use and understand, with clear documentation and user-friendly interfaces.
7. **Portability:** GPT-2 should be able to run on different hardware and software platforms, making it easy to deploy and use in various environments.
8. **Compatibility:** The model should be compatible with other systems and technologies, allowing for seamless integration and interoperability.

3.4.1 BLEU Score:

BLEU (BiLingual Evaluation Understudy) is a metric that provides an automated way of assessing the quality of machine-translated text. It generates a score between zero and one, which represents the degree of similarity between the machine-translated text and a set of high-quality reference translations. A score of 0 indicates that the machine-translated output is dissimilar to the reference translation (low quality), while a score of 1 indicates that the machine-translated output is identical to the reference translations (high quality).

Studies have shown that BLEU scores are highly correlated with human judgment of translation quality. However, even professional human translators do not always achieve a perfect BLEU score of 1.0.

$$\text{BLEU} = \underbrace{\min\left(1, \exp\left(1 - \frac{\text{reference-length}}{\text{output-length}}\right)\right)}_{\text{brevity penalty}} \underbrace{\left(\prod_{i=1}^{4} \text{precision}_i\right)^{1/4}}_{\text{n-gram overlap}}$$

$$precision_i = \frac{\sum_{\text{snt} \in \text{ Cand-Corpus}} \sum_{i \in \text{snt}} \min\left(m^i_{\text{cand}}, m^i_{ref}\right)}{w^i_t = \sum_{\text{snt}' \in \text{ Cand-Corpus}} \sum_{i' \in \text{ snt}} , m^{i'}_{\text{cand}}}$$

where,

- m^i_{cond}- number of i-gram in candidate like translation of reference
- m^i_{ref}- number of i-gram in translation of reference
- w^i_t - number of i gram in translation of candidate

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Fig 3.1- Interpretation of BLEU Score

Source: Google ML

3.4.2 ROUGE Score:

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is a set of metrics used for evaluating the quality of summaries produced by automatic text summarization algorithms. ROUGE measures the overlap between the generated summary and a set of reference summaries (usually created by human experts). Specifically, it computes precision, recall, and F1 scores based on the n-grams (sequences of n consecutive words) in the summary and reference summaries. There are several variants of ROUGE, each using a different method for computing n-grams and aggregating the scores. Some of the commonly used ROUGE variants include ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 measures the overlap of unigram (single word) sequences between the generated summary and reference summaries. ROUGE-2 measures the overlap of bigram (two-word) sequences, and ROUGE-L measures the longest common subsequence between the generated summary and reference summaries. ROUGE scores are often used in research on automatic text summarization to compare the performance of different summarization algorithms or to tune the parameters of a single algorithm. They can also be used to evaluate the quality of summaries generated by machine translation systems or other natural language processing applications.

1. **Recall:**

   The ROUGE metric's recall score measures the degree of overlap between the n-grams found in the reference and the model output. It does so by calculating the ratio of overlapping n-grams to total of n-grams in reference. This score indicates the proportion of relevant n-grams that were captured by the model's generated summary. It is represented mathematically as:

$$\frac{ngramsfoundin(model + reference)}{\text{total } n\ grams\ present\ in\ the\ reference}$$

2. **Precision:**

   Precision is computed in a similar manner to recall, except that it divides the overlapping n-grams by total of n-grams in the model output, rather than in the reference summary. This metric measures the proportion of n-grams in the model generated summary that were correctly included in the reference summary.. It is represented mathematically as:

$$\frac{ngramsfoundin(model + reference)}{total\ n\ grams\ present\ in\ the\ model}$$

3. **F1-score:**

   The F1 score is a metric that combines both precision and recall into a single value, calculated as the harmonic mean of these two metrics, and is defined as:

$$2 * \frac{precision * recall}{precision + recall}$$

3.5 System Requirements:

Initially the pre-processing, feature engineering and the model fitting was performed on a Laptop with the below mentioned hardware specification. As the size and dimensions of the dataset grew the model was moved to the Google Colab Pro Plus.

3.5.1 Hardware Requirements

The initial development was done on a MacBook with the following specifications:

RAM	16GB
Processor	Apple M1 chip, CPU @3.2GHz
Hard Disk	250GB
Number of CPUs	4 high performance “Firestorm” and 4 energy efficient “IceStorm “ cores = 8 core CPU
Number of GPUs	8 core GPU
Operating System	macOS Ventura 13.2.1

Table 3.1 – Hardware Requirements

3.5.2 Software Requirements

The software requirements of the project are tabulated below. During the later stages of development the system was developed on the Google Colab Pro Plus and the cloud resources used are as below:

Cloud environment	Google Colab Pro Plus
Development language	Python 3.8
Deep Learning Framework	Pytorch,Tensorflow
Third party Libraries	Pandas Numpy NLTK Metaplotlib Seaborn tqdm metapub transformer
Log monitoring Framework	Tensor Board

Table 3.2 – Software Requirements

4. PROPOSED METHODOLOGY

The study presented in this thesis describes a novel method of deriving the usage of a medicine when the medicine name is entered as input to GPT-2 . As described in the Chapter 2, there are studies in the past that targeted various problems in healthcare domain, but no system has tried to explore this problem statement. As a part of this thesis, I tried to implement the model with some minor modifications to improve the performance of the model.

The architecture of the system is described in 4.1. algorithm is described in 4.2, followed by a detailed description of the software design in 4.3.

4.1 System Architecture:

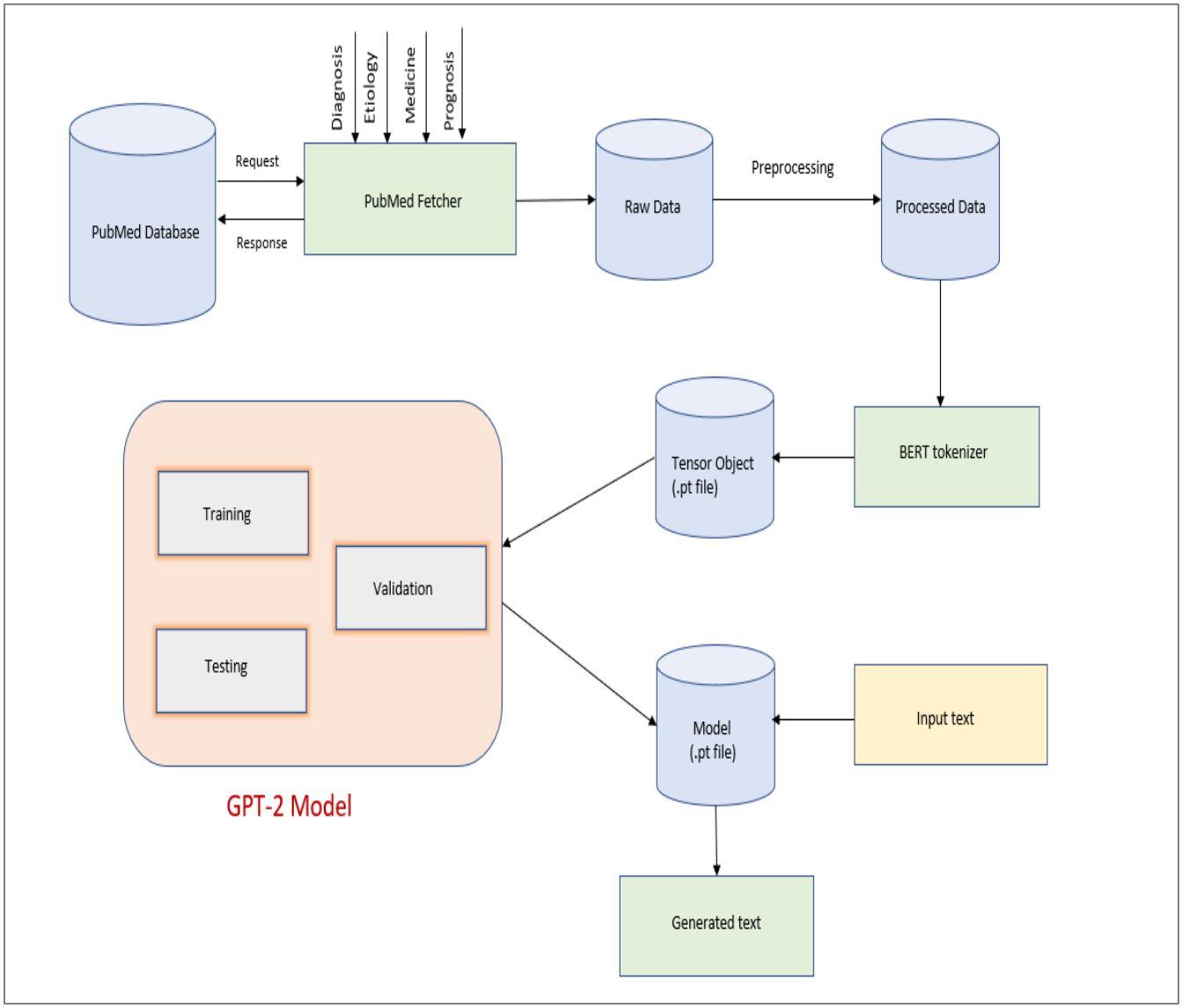


Fig 4.1– Architecture Diagram of proposed System

4.2 System Algorithm:

The algorithm of the system is as below:

1. *Collect the PubMed (medicine) data.*
2. *Process the raw data (null removal, special character removal, emoji removal etc).*
3. *Tokenize the process data.*
4. *Create training and validation dataset from tokenized data.*
5. *Create the segment tokens.*
6. *Return the last token before the padding.*
7. *For better result shuffle the tuple.*
8. *Create the tensor object for GPT-2 model.*
9. *Load the pretrained Distill GPT-2 model.*
10. *Add the special token using GPT-2 tokenizer.*
11. *Resize the tokens embedding as GPT-2 supports max length of 1024.*
12. *Load the previously saved tensor data using data loader and passing to the GPT2 model for training.*
13. *Fine tune the GPT-2 model using hyperparameter tuning.*
14. *GPT-2 takes the input text such as medicine name and generate the text.*

4.3 System Design:

PubMed is a free search engine developed by the National Library of Medicine (NLM) at the National Institutes of Health (NIH) in the United States. It provides access to a vast collection of biomedical literature, including scientific articles, books, and conference proceedings. PubMed contains over 32 million citations and abstracts from MEDLINE, a bibliographic database of life sciences and biomedical information, as well as additional citations from PubMed Central (PMC), a digital archive of free, full-text biomedical and life sciences journal literature. We cannot directly download the data. To download the data I used PubMed Fetcher python library which takes different kinds of key words such as diagnosis, medicine etc and gives the related data, which was saved in csv format. The raw data was not cleaned, so different kinds of pre-processing such as null removal, duplicate removal, special character removal, emoji removal etc. had been performed to clean the data. Apart from cleaning, one important step has been performed which is Part-of-Speech (POS) tagging. It is a process of labeling each word in a

sentence with its corresponding part of speech, such as noun, verb, adjective, adverb, preposition, conjunction, or interjection. By identifying the part of speech of each word in a sentence, the computer can better understand the grammatical structure of the sentence, which is essential for accurately interpreting the meaning of the text. POS tagging was performed using python NLTK library .The final step before model training was tokenization. BERT tokenizer was used for tokenization. The BERT tokenizer is based on WordPiece, a sub word tokenization method that breaks down words into smaller units of meaning. This allows BERT to handle out-of-vocabulary (OOV) words and to capture more fine-grained details of language. Training, Validation, and testing was performed on tokenized data using Distilled GPT2. After the training ,model was saved in .pt file format in secondary disk for future use. Now if any input text is passed to the model it will generate the relevant text.

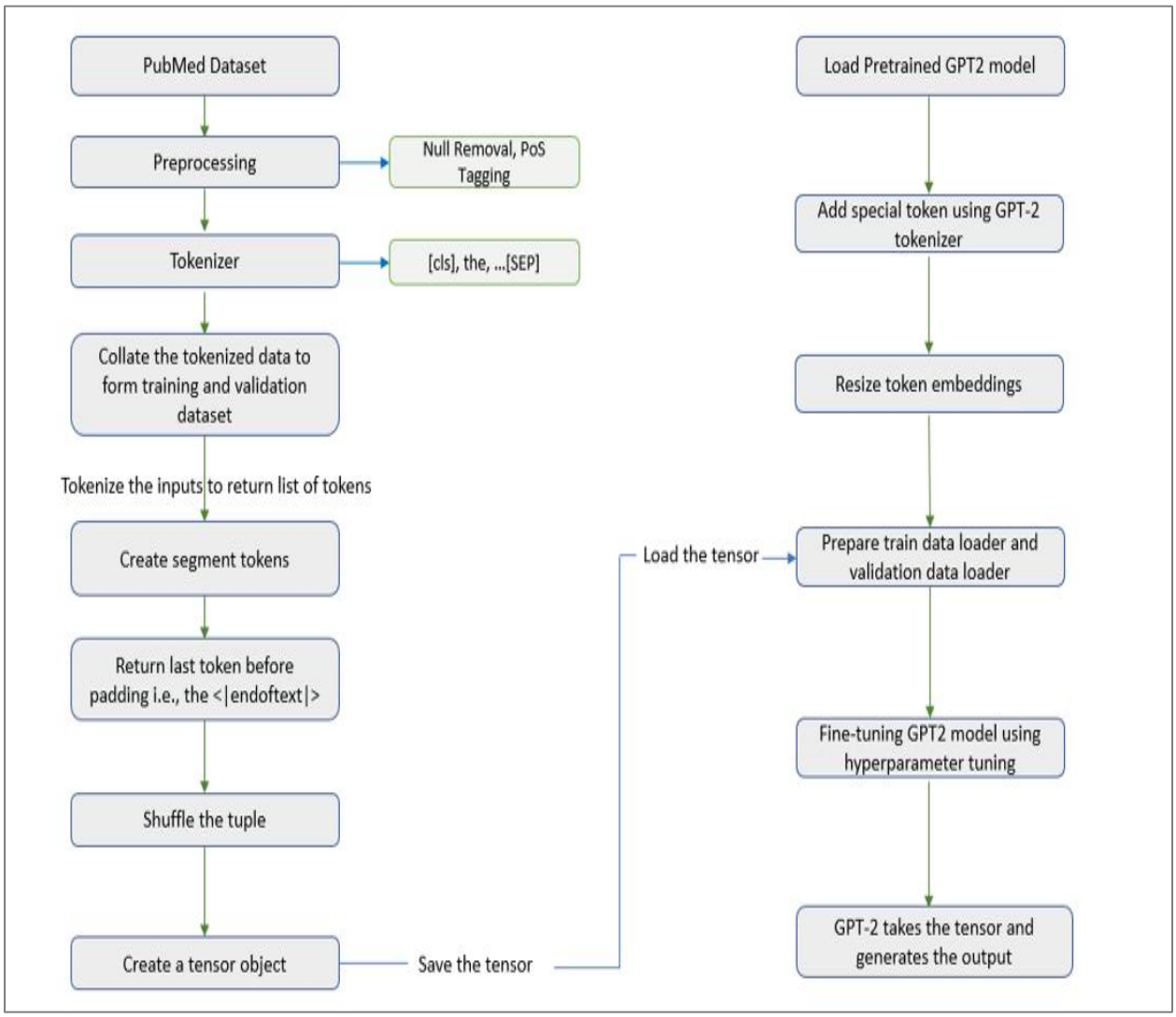


Fig 4.2– Block Diagram of the proposed System

5. IMPLEMENTATION DETAILS

This section describes the details of the implementation for the text generation part of the system.

5.1 Data Collection:

To train the GPT2 model on medical corpus, required a huge set of medical data. There were few authentic sources which provided verified medical data. PubMed search engine was one of them, but it didn't allow to download the data directly. PubMedFetcher, a python-based library was used to download the data. PubMedFetcher has a special method called pmids_for_query which accepts two parameter- keywords and number of articles. It returns a unique PubMed id. Now PubMed id was passed to another PubMedFetcher method called article_by_pmid and it returned title and abstract. Title and abstract were the two columns that plays a crucial role in data preparation. Seven different kind of keywords such as diagnosis, prognosis, medicine, sepsis, prediction, etiology, and therapy were used to fetch the PubMed data and it fetched 1000 unique rows for each keyword. All the fetched data was stored into dataframes. Finally all the data frames were merged into single data frame. The final data frame was stored into disk for future use.

5.2 Tokenization:

- **Loading the tokenizer:** BertTokenizer with pre-trained 'bert-base-cased' model was used to load the three files- *vocab.txt, tokenizer_config.json* and *config.json*. The vocab.txt acted as a dictionary for token classification.

Fig 5.1 – Loading the tokenizer

- **Collating the tokenized NER data and labels:** An important aspect was to collate tokenized sentence and labels. The NER data was used for bert token classification. BertForTokenClassification was used to load the pre-trained 'bert-base-cased' which further downloaded the *pytorch_model.bin* file. The size of the *pytorch_model.bin* was 436M.

5.3 Fine tuning the token classifier:

In this step I fine-tuned the token for token classification. Adam optimizer was used with learning rate $3e^{-5}$, eps $1e^{-8}$ and three no decay values- bias, gamma, and beta for fine tuning parameters. Total number of training steps was **number of batches * number of epochs** for fine tuning. While fine tuning to prevent the "*exploding gradient*" problem, the norm was clipped from the gradient using clip_grad_norm_ method. Loss was calculated for each epoch.

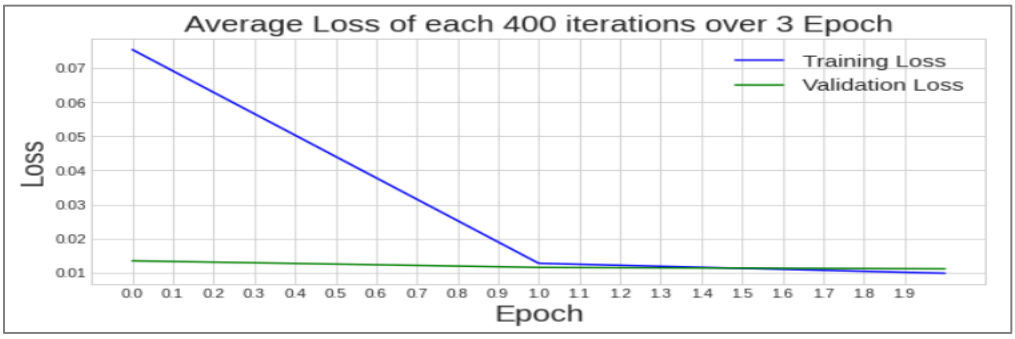


Fig 5.2 – Average loss vs epoch for token classification

To test the token classification I took some random sentence and passed to the model and it generated the corresponding token and labels.

```
test_sentence = """
Psoralen and isopsoralen, furocoumarins isolated from the plant Psoralea corylifolia L., were demonstrated to exhibit in vitro inhibitory actions on
monoamine oxidase (MAO) activities in rat brain mitochondria, preferentially inhibiting MAO-A activity over MAO-B activity. This inhibition of enzyme
activities was found to be dose-dependent and reversible.
"""
```

```
JJ          [CLS]
NNP         Psoralen
CC          and
NNP         isopsoralen
,           ,
NNS         furocoumarins
VBN         isolated
IN          from
DT          the
NN          plant
NNP         Psoralea
NNP         corylifolia
NNP         L
NNP         .
,           ,
VBD         were
VBN         demonstrated
TO          to
VB          exhibit
IN          in
JJ          vitro
NN          inhibitory
NNS         actions
IN          on
NN          monoamine
NNP         oxidase
LRB         (
NNP         MAO
RRB         )
NNS         activities
```

```
IN          in
NN          rat
NN          brain
NN          mitochondria
,           ,
RB          preferentially
VBG         inhibiting
NNP         MAO
NNP         -
NNP         A
NN          activity
IN          over
NNP         MAO
NNP         -
NNP         B
NN          activity
NN          .
DT          This
NN          inhibition
IN          of
NN          enzyme
NNS         activities
VBD         was
VBN         found
TO          to
VB          be
JJ          dose
JJ          -
JJ          dependent
CC          and
JJ          reversible
NN          .
NN          [SEP]
```

Fig 5.3 – Classification of words

Next, I generated the confusion matrix for true label and prediction label. The true label came from validation data loader.

```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 9.88091187e-01, 1.47442441e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 1.96826178e-03, 9.72567351e-01, ...,
        6.15081806e-05, 5.53573625e-04, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        9.92896799e-01, 8.16459830e-05, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 6.54236179e-04, ...,
        3.27118090e-04, 9.97710173e-01, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

Fig 5.4– Confusion matrix

Below, is the heatmap of confusion matrix with POS tagging.

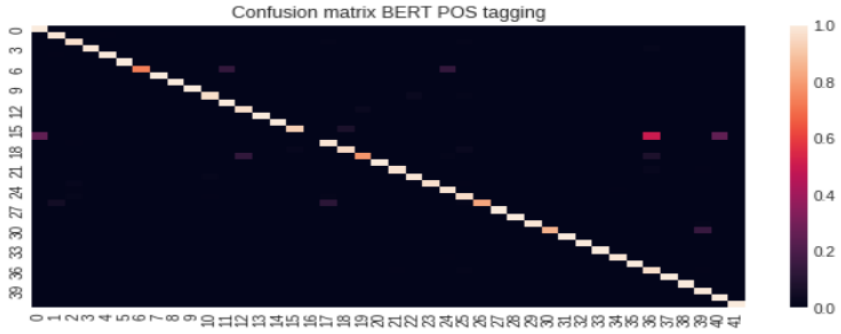


Fig 5.5– Heatmap of the confusion matrix

As can be seen from the above heatmap, majority of the plot is dark, meaning that there is rarely any correlation among the words, whereas we see that there is a comparatively lighter shade diagonally, meaning that the words are auto-correlated. Finally, I am saving the BERT embedding in the disk for future use.

```
OrderedDict([('bert.embeddings.position_ids',
              tensor([[  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
                        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,  27,
                        28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,  40,  41,
                        42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,  53,  54,  55,
                        56,  57,  58,  59,  60,  61,  62,  63,  64,  65,  66,  67,  68,  69,
                        70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  80,  81,  82,  83,
                        84,  85,  86,  87,  88,  89,  90,  91,  92,  93,  94,  95,  96,  97,
                        98,  99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111,
                       112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125,
                       126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139,
                       140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153,
                       154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
                       168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
                       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
                       196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209,
                       210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223,
                       224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
                       238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251,
                       252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265,
                       266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279,
                       280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293,
                       294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307,
                       308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321,
                       322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335,
                       336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349,
                       350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
                       364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377,
                       378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391,
                       392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405,
                       406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419,
                       420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433,
                       434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447,
                       448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461,
                       462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475,
                       476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
                       490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503,
                       504, 505, 506, 507, 508, 509, 510, 511]], device='cuda:0')),
             ('bert.embeddings.word_embeddings.weight',
              tensor([[-0.0005, -0.0416,  0.0131,  ..., -0.0039, -0.0335,  0.0150],
                      [ 0.0169, -0.0311,  0.0042,  ..., -0.0147, -0.0356, -0.0036],
                      [-0.0006, -0.0267,  0.0080,  ..., -0.0100, -0.0331, -0.0165],
                      ...,
```

Fig 5.6– BERT embeddings

5.4 Pre-processing the raw text for GPT-2:

a.) As part of Pre-processing the actions that had been performed are - remove NaN, duplicate, special character etc. I found out that the abstract column of PubMed data had some duplicates. *dropna* method from pandas was used to remove the duplicate.

b.) Next step was POS tagging. Part-of-speech (POS) tagging, also known as grammatical tagging, is the process of assigning a part of speech (such as noun, verb, adjective, adverb, etc.) to each word in a sentence. POS tagging is a fundamental task in natural language processing (NLP) and is used in many applications such as text classification, machine translation, and information retrieval. The POS tag for a word depends on its definition, context, and relationship with other words in the sentence. For example, in the sentence. "The cat sat on the mat," the word "cat" is a noun, "sat" is a verb, "on" is a preposition, and "the" is a determiner. The list for the POS tagging consists of ["FW", "JJ", "NN", "NNS", "NNP", "VB", "VBD", "VBG", "VBN", "VBZ", "VBP"]. *averaged_perceptron_tagger* from NLTK was used for tagging.

```
Synthetic neutrophil inhibitor PMN apoptosis c...
SL422 TNF converting enzyme ADAM17 search TNF-...
Zenarestat Aldose reductase AKR1B1 effects zen...
Howiinol A GHM-10 DNA topoisomerase II TOP2 Ho...
zoxazolamine Calcium-activated potassium chann...
keyword POS str, dtype: object
```

Fig 5.7–Example of NLTK PoS tagging

	pmid	title	abstract	keyword_POS	keyword_POS_str
0	36906732	Dental caries and dental developmental defects...	PURPOSE: To evaluate the prevalence of dental ...	[[CLS], purpose, evaluate, prevalence, dental,...	[CLS] purpose evaluate prevalence dental carie...
1	36906724	Two novel compound heterozygous variants of th...	Autosomal recessive glutaric acidaemia type I ...	[[CLS], autosomal, recessive, glutaric, acidae...	[CLS] autosomal recessive glutaric acidaemia t...
2	36906721	Impact of two ketogenic diet types in refracto...	BACKGROUND: Ketogenic diet (KD) refers to any ...	[[CLS], background, ketogenic, diet, kd, refer...	[CLS] background ketogenic diet kd refers diet...
3	36906717	Investigation of the clinical utility of adhes...	To better understand the relationship among ce...	[[CLS], understand, relationship, cell, adhesi...	[CLS] understand relationship cell adhesion mo...
4	36906716	Downregulation of PSAT1 inhibits cell prolifer...	Phosphoserine aminotransferase 1 (PSAT1) has b...	[[CLS], phosphoserine, aminotransferase, 1, ps...	[CLS] phosphoserine aminotransferase 1 psat1 h...

Fig 5.8–PoS tagging on whole data (top 5 samples)

c.) Next, the dataset was partitioned using train_test_split method with 90:10 ratio into training and validation. Eight different tasks had been performed as a part of tokenization and collation.

Pseudocode:

1. load_words : It imports the dataframe with number of samples to choose, return a keyword (together with title) as strings and abstract (gold label for generation) and 3 distractors. This is a tuple of 5 strings and the 0th element in list is always the correct pair.

2. write_input_ids: tokenize the input and return a list of tokens in this format

'<|startoftext|> ' + keyword + ' <|summarize|> '+ abstract + ' <|endoftext|>'

'<|startoftext|> ' + keyword + ' <|summarize|> '+ distractor1 + ' <|endoftext|>'

'<|startoftext|> ' + keyword + ' <|summarize|> '+ distractor2 + ' <|endoftext|>'

'<|startoftext|> ' + keyword + ' <|summarize|> '+ distractor3 + ' <|endoftext|>'

3. write_token_type_labels: this function accepts list_input_ids and create a segment token for the sentence (either keyword, context, or padding segment)

4. write_lm_labels: It accepts list_input_ids and list_type_labels as a parameter and return the label for the lm(language model) head (only on the correct pair i.e. the 0th element of the list else wherever it is masked with [-100] token to prevent the model from computing cross entropy loss).

5. write_last_token: returns the last token before padding (i.e. the <|endoftext|>) this is recognized by the mc(multiple choice) head for the multiple-choice loss.

6. write_mc_label: It returns [1,0,0,0] list , because the correct pair is always the 0th element of the list

7. shuffle_batch: It will shuffle the tuple (list_input_ids, list_type_labels, list_last_tokens, list_lm_labels, list_mc_labels). After this the correct pair can be any element and finally returns shuffled NumPy array.

8. write_torch_tensor: create a tensor object from the NumPy array and returns torch_input_token, torch_segment, torch_mc_token, torch_lm_label and torch_mc_label.

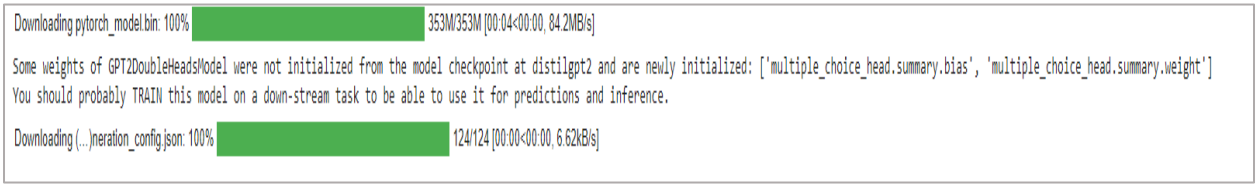
After successful execution of those eight functions I saved the training and validation dataset as a tensor object in the disk for the training.

count	inputdecoded input	inputdecoded input	inputdecoded input	inputdecoded input
0	50257<|startoftext|>	50257<|startoftext|>	50259<|keyword|>	-100 masked
1	58 [	58 [	50259<|keyword|>	-100 masked
2	5097 CL	5097 CL	50259<|keyword|>	-100 masked
3	50 S	50 S	50259<|keyword|>	-100 masked
4	60]	60]	50259<|keyword|>	-100 masked
5	479 k	479 k	50259<|keyword|>	-100 masked
6	1031 az	1031 az	50259<|keyword|>	-100 masked
7	259 in	259 in	50259<|keyword|>	-100 masked
8	349 ol	349 ol	50259<|keyword|>	-100 masked
9	269 c	269 c	50259<|keyword|>	-100 masked
10	1259 ty	1259 ty	50259<|keyword|>	-100 masked
11	4951 ros	4951 ros	50259<|keyword|>	-100 masked
12	259 in	259 in	50259<|keyword|>	-100 masked
13	589 ase	589 ase	50259<|keyword|>	-100 masked
14	22934 tyr	22934 tyr	50259<|keyword|>	-100 masked
15	352 1	352 1	50259<|keyword|>	-100 masked
16	837 ,	837 ,	50259<|keyword|>	-100 masked
17	513 3	513 3	50259<|keyword|>	-100 masked
18	532 -	532 -	50259<|keyword|>	-100 masked
19	19550 dip	19550 dip	50259<|keyword|>	-100 masked
20	831 hen	831 hen	50259<|keyword|>	-100 masked
21	2645 yl	2645 yl	50259<|keyword|>	-100 masked
22	1676 pro	1676 pro	50259<|keyword|>	-100 masked
23	6839 pan	6839 pan	50259<|keyword|>	-100 masked
24	274 es	274 es	50259<|keyword|>	-100 masked
25	46017 exhibiting	46017 exhibiting	50259<|keyword|>	-100 masked
26	26776 inhibit	26776 inhibit	50259<|keyword|>	-100 masked
27	652 ory	652 ory	50259<|keyword|>	-100 masked
28	4568 activities	4568 activities	50259<|keyword|>	-100 masked
29	15848 monop	15848 monop	50259<|keyword|>	-100 masked
30	831 hen	831 hen	50259<|keyword|>	-100 masked
31	349 ol	349 ol	50259<|keyword|>	-100 masked
32	589 ase	589 ase	50259<|keyword|>	-100 masked
33	19550 dip	19550 dip	50259<|keyword|>	-100 masked
34	831 hen	831 hen	50259<|keyword|>	-100 masked

Fig 5.9–Encoded and masked inputs

5.5 Training GPT2 model for text generation:

- First we load GPT2DoubleHeadsModel and GPT2Tokenizer for training.



- Training consists of two task train and evaluate. Before training *special token ('bos_token': '<|startoftext|>','eos_token':'<|endoftext|>','pad_token':'<pad>', 'additional_special_tokens': ['<|keyword|>','<|summarize|>'])* were added to the GPT2 tokenizer and resized the token embeddings. Training and validation data were loaded using pytorch data loader. The dataset was in the torch tensor format. It was bundled into a tuple of 5 items:
    1. input tokens
    2. segment tokens
    3. index for last token (this is used for multiple choice)
    4. the language model expected output tokens, the masked [-100] is used to mask away part that model doesn't have to output.
    5. the multiple-choice label among which one of the 4 items in the current batch is the correct choice.

- The train method accepts five parameters which are batch(tensor data),number of iterations, GPT2 model, optimizer, and scheduler. Adam optimizer was used with lr=$5e^{-5}$ and eps=$1e^{-8}$ for training. I used *get_linear_schedule_with_warmup* scheduler with 50 warm up steps from transformer.
- During loss calculation, three losses were calculated for the training - LM loss (Language model), MC loss (Multiple choice) and total loss.

$$\textbf{Total loss} = \frac{lm-loss*lm-coeff + mc-loss*mc-coeff}{accumulating\ gradient}$$

- I used clip_grad_norm_ from torch utils package to prevent the exploding gradient problem. Training was configured with 20 epochs.
- The evaluate method was used for validation. It took two params - validation tensor data and GPT2 model, to perform the validation.
- After successfully training the model, the training loss logs were saved in the disk for future purpose.

6. INTERMEDIATE RESULTS AND DISCUSSION

This chapter presents the intermediate results found for the procedures elaborated in the previous section, and also describes the datasets used for the project, and the evaluation metrics considered. The results presented here are intended only as a baseline and will be further worked upon to improve them in the next phase of the project.

6.1 Datasets:

PubMed is a free search engine maintained by the National Library of Medicine (NLM) that provides access to biomedical literature, including journal articles and book chapters. PubMed comprises more than 32 million citations for biomedical literature from MEDLINE, life science journals, and online books. PubMed is a crucial resource for healthcare professionals, researchers, and academics in the biomedical field. The PubMed dataset is a collection of metadata and abstracts of biomedical literature that are indexed in PubMed. The dataset contains structured information about articles, including the title, authors, abstract, publication date, journal name, and more. The PubMed dataset is available for free and can be accessed through the National Center for Biotechnology Information's (NCBI) website. Additionally, the dataset is available for download in various formats, including XML, CSV, and JSON. Researchers and data scientists can use the PubMed dataset to perform various analyses, such as text mining, natural language processing, and machine learning. The dataset can be used to identify trends in research, track the evolution of scientific concepts, and develop new tools for knowledge discovery. Additionally, the PubMed dataset can be used to build recommendation systems for healthcare professionals, academics, and researchers, helping them to discover relevant articles and stay up-to-date with the latest research.

0	pmid	Title	Abstract
0	36906732	Dental caries and dental developmental defects...	PURPOSE: To evaluate the prevalence of dental ...
1	36906724	Two novel compound heterozygous variants of th...	Autosomal recessive glutaric acidaemia type I ...
2	36906721	Impact of two ketogenic diet types in refracto...	BACKGROUND: Ketogenic diet (KD) refers to any ...
3	36906717	Investigation of the clinical utility of adhes...	To better understand the relationship among ce...
4	36906716	Downregulation of PSAT1 inhibits cell prolifer...	Phosphoserine aminotransferase 1 (PSAT1) has b...

Fig 6.1–PubMed Diagnosis Dataset

0	pmid	Title	Abstract
0	11169165	Inhibition of rat brain monoamine oxidase acti...	Psoralen Monoamine oxidase type A (MAO-A) Psor...
1	11781254	Selective inhibitor of Janus tyrosine kinase 3...	PNU156804 Janus kinase 3 (JAK-3) Janus kinase ...
2	12724039	Thiorphan enhances bradykinin-induced vascular...	Thiorphan Neutral endopeptidase (MME) Relaxati...
3	1330184	Agonist analysis of 2-(carboxycyclopropyl)glyc...	L-CCG-I Metabotropic glutamate receptor 4 (mGl...
4	12517773	The protein kinase C inhibitor Go6976 is a pot...	G6976 Protein kinase C (PRKC) In response to D...

Fig 6.2–PubMed Medicine Dataset

0	pmid	Title	Abstract
0	36906699	[Nodule of uncertain etiology on the scrotum].	NaN
1	36906694	Shared genetic architecture between attention-...	There is evidence linking ADHD to a reduced li...
2	36906691	Pediatric Myocarditis.	Myocarditis is a condition caused by acute or ...
3	36906532	Family structure and multisite musculoskeletal...	BACKGROUND: Family structure is suggested to b...
4	36906457	Beyond the amyloid cascade: An update of Alzhe...	Alzheimer's disease (AD) is a multi-etiology d...

Fig 6.3–PubMed Etiology Dataset

0	pmid	Title	Abstract
0	36906727	PBTK modeling of the pyrrolizidine alkaloid re...	Retrorsine is a hepatotoxic pyrrolizidine alka...
1	36906723	DBS-evoked cortical responses index optimal co...	Although subthalamic deep brain stimulation (D...
2	36906720	Organ dysfunction and mortality in preterm neo...	BACKGROUND: Organ dysfunction (ODF) in late-on...
3	36906716	Downregulation of PSAT1 inhibits cell prolifer...	Phosphoserine aminotransferase 1 (PSAT1) has b...
4	36906715	Immune priming with avelumab and rituximab pri...	Immune evasion, due to abnormal expression of ...

Fig 6.4–PubMed Prediction Dataset

0	pmid	Title	Abstract
0	36906734	Superinfections in COVID-19 patients receiving...	The risk of superinfections and associations w...
1	36906719	Atrial fibrillation as a new prognosis factor ...	A collaborative project in different areas of ...
2	36906716	Downregulation of PSAT1 inhibits cell prolifer...	Phosphoserine aminotransferase 1 (PSAT1) has b...
3	36906674	PDIA4 confers resistance to ferroptosis via in...	The prognosis of renal cell carcinoma (RCC) re...
4	36906670	Clinical and biomarker analyses of sintilimab ...	The prognosis of biliary tract cancer (BTC) re...

Fig 6.5–PubMed Prognosis Dataset

0	pmid	Title	Abstract
0	36906720	Organ dysfunction and mortality in preterm neo...	BACKGROUND: Organ dysfunction (ODF) in late-on...
1	36906606	Embracing complexity in sepsis.	Sepsis involves the dynamic interplay between ...
2	36906561	HMGB1 mediates synaptic loss and cognitive imp...	BACKGROUND: Microglial activation-mediated neu...
3	36906123	Primary empty sella syndrome-caused rhabdomyol...	We reported a case of a 68-year-old man who pr...
4	36905669	Variability in neurosurgical management and as...	OBJECTIVE: Posthemorrhagic hydrocephalus (PHH)...

Fig 6.6–PubMed Sepsis Dataset

6.2 Intermediate Results:

This model has the capability to generate text using beam search and top k sampling. Beam search is a popular heuristic algorithm used in natural language processing (NLP) for finding the most likely sequence of words in a sequence-to-sequence model, such as in neural machine translation or text generation tasks. The basic idea of beam search is to maintain a list of the top k most likely sequences of words, called the "beam," at each time step. The size of the beam, k, is a hyperparameter that determines how many sequences to keep track of. At each time step, the algorithm expands each sequence in the beam by generating all possible next words, and scores each resulting sequence based on the probabilities assigned to the candidate words by the model. The top k sequences with the highest scores are kept in the beam, while the others are discarded. The algorithm repeats this process for each subsequent time step until the end of the sequence is reached or a maximum length is reached. Finally, the sequence with the highest score in the final beam is chosen as the output. Beam search is often used instead of greedy search, which simply chooses the most likely word at each time step. Beam search can be slower than greedy search, but it often results in more accurate outputs because it explores a larger search space, whereas Top-k sampling is a technique used in text generation that involves selecting the k most likely words or tokens to be generated at each step, based on their probabilities as determined by the language model. Generally beam search takes more time than top-k sampling but it generates quality text.

Output with Beam Search:

```
1 generate_text('Bicalutamide', isBeamSearchReq=True)

Input to the model is : <|startoftext|> Bicalutamide <|generate|>

Generated text ::::
['Bicalutamide (Casodex), a non-steroidal anti-androgen, has been shown', 'to be effective in the treatment of patients with advanced prostate', 'cancer.
```

```
1 generate_text('Apricitabine', isBeamSearchReq=True)

Input to the model is : <|startoftext|> Apricitabine <|generate|>

Generated text ::::
['Apricitabine: A Review in Chronic Hepatitis C Genotype 1. < / FREETEXT', '> < / TITLE > ▄ < ABSTRACT > < FREETEXT > Apricitabine (Technivie ®)', 'is a nucleoside analogue inhibitor of the hepatitis C virus (HCV)
```

```
1 generate_text('Xylazine', isBeamSearchReq=True)

Input to the model is : <|startoftext|> Xylazine <|generate|>

Generated text ::::
['Xylazine (0. 0 5 mg / kg) and butorphanol (0. 0 5 mg / kg) were', 'administered intramuscularly (IM) to each dog. Anesthesia was induced', 'with ketamine (2. 2 mg / kg, IM) and diazepam (0. 0 5 mg / kg, IM).',
```

```
1 generate_text('Psoralen', isBeamSearchReq=True)

Input to the model is : <|startoftext|> Psoralen <|generate|>

Generated text ::::
['Psoralen plus UVA (PUVA) therapy for psoriasis. < / FREETEXT > < /', 'TITLE > ▄ < ABSTRACT > < FREETEXT > Psoralen plus UVA (PUVA) therapy', 'has been used for more than 3 0 years for the treatment of psoriasis.'
```

Fig 6.7 – Output with Beam Search

Output with Sampling:

```
generate_text('Bicalutamide', isBeamSearchReq=False)
```

```
Input to the model is : <|startoftext|> Bicalutamide <|generate|>

Generated text ::::
{'generated_text': 'Bicalutamide (1 mg / kg.body wt) was injected subcutaneously once daily for 6 0 days, and rats were anesthetized to obtain blood specimens. Serum testosterone, ALT / AST le

{'generated_text': 'Bicalutamide, which has not been associated with the development of gynecomastia, caused breast pain in one patient. One man was lost to follow up after he had had his seco

{'generated_text': 'Bicalutamide (1 0 μM). < / FREETEXT > < / PARAGRAPH > ▬ < PARAGRAPH > < FREETEXT > This observation was also consistent with the decreased AR nuclear localization in respo

{'generated_text': 'Bicalutamide treatment in the absence of androgens enhanced AR degradation. Therefore, we determined whether the observed enhanced degradation of AR was due to ubiquitin-me

{'generated_text': 'Bicalutamide-containing therapy improved the OS of CRPC patients at 1 2 years (6. 9 vs 1 2. 4%) with less treatment related side effects (side effects from bicalutamide / r
```

```
generate_text('Apricitabine', isBeamSearchReq=False)
```

```
Input to the model is : <|startoftext|> Apricitabine <|generate|>

Generated text ::::
{'generated_text': 'Apricitabine (FTC) is an effective antiviral against several RNA viruses, including hepatitis C virus (HCV). However, HCV resistance to FTC was found to evolve duri

{'generated_text': 'Apricitabine has been shown to be effective as maintenance therapy for chronic hepatitis B patients, particularly for HBeAg-negative patients. < / FREETEXT > < / PA

{'generated_text': 'Apricitabine (2 0 0 mg twice daily). A sustained virologic response (SVR) was obtained in 1 5 / 1 9 patients after 1 year of therapy. The response rate was higher'}

{'generated_text': 'Apricitabine is the first member of a novel 2 ’ -F-dC series of antiviral agents that inhibits replication of HBV. In a phase 2b proof-of-concept study of 3 4'}

{'generated_text': 'Apricitabine-resistant replicons were identified in this study, all of which contain mutations in the region of NS5B that codes for the polymerase catalytic domain.
```

```
generate_text('Xylazine', isBeamSearchReq=False)
```

```
Input to the model is : <|startoftext|> Xylazine <|generate|>

Generated text ::::
{'generated_text': 'Xylazine (1 mg / kg, IV) to cause a medetomidine-induced anesthesia followed by an isoflurane-induced anesthesia, the animals were moved into the magnetic resonance (MR) ro

{'generated_text': 'Xylazine, 0. 0 1%), mice were placed in a transparent, transparent cylinder (height: 3 6 cm, diameter: 2 0 cm) with 2 0 ml of warm water at'}

{'generated_text': 'Xylazine (0. 9 ml xylazine) per 1 5 kg of body weight. Once the animal was unconscious, the head was gently positioned to ensure that the eyes were in the plane of focus an

{'generated_text': 'Xylazine hydrochloride (Vetergesic), 2 mg kg - 1 per injection, was diluted in saline. < / FREETEXT > < / PARAGRAPH > ▬ < PARAGRAPH > < FREETEXT > All results are'}

{'generated_text': 'Xylazine-xylazine mixture was injected IV prior to the experiments. The lungs (without right upper lobe) were exposed by a dorsal thoracotomy. A catheter was inserted into
```

```
generate_text('Psoralen', isBeamSearchReq=False)
```

```
Input to the model is : <|startoftext|> Psoralen <|generate|>

Generated text ::::
{'generated_text': 'Psoralen-exposed and unexposed mouse spleen cells were stimulated with a B-cell-enriched mixture of monoclonal mouse antibodies B1 1B1 1 and 1 3. 2.1, which recognize idiot

{'generated_text': 'Psoralen plus ultraviolet A plus methotrexate / PUVA is a highly effective treatment modality for psoriasis that has recently been introduced into the U.K. There is a wide

{'generated_text': 'Psoralen plus Psoralen plus Wood. < / FREETEXT > < / TITLE > ▬ < ABSTRACT > < FREETEXT > This prospective study was carried out to evaluate the treatment of vitiligo by a

{'generated_text': 'Psoralen treatment of the psoriatic skin resulted in a pronounced skin photosensitivity. < / FREETEXT > < / ABSTRACT > ▬'}

{'generated_text': 'Psoralen-PUVA therapy for mycosis fungoides: a study of 1 2 patients with an average follow-up of 3 years. < / FREETEXT > < / TITLE > ▬ < ABSTRACT > < FREETEXT >'}
```

```
generate_text('Herbimycin A', isBeamSearchReq=False)
```

```
Input to the model is : <|startoftext|> Herbimycin A <|generate|>

Generated text ::::
{'generated_text': 'Herbimycin A (1 microM) or the protein tyrosine phosphatase inhibitor, sodium orthovanadate (1 mM), abolished agonist-induced down-regulation of the receptor protein and,

{'generated_text': 'Herbimycin A, which inhibits the EGF / receptor kinase, inhibited all these responses. Activation of the Na + / H + exchanger was also observed in Chinese hamster lung fi

{'generated_text': 'Herbimycin A (0. 5 μM) and MG1 3 2 (1 0 μM) in DMEM were added to the dishes 1 5 min after PDGF stimulation. The cells were then extracted with'}

{'generated_text': 'Herbimycin A treatment in the A5 4 9 cells (Fig. 4A). The result was found to be similar in an in vitro ubiquitin assay with a cell extract (Fig. 4D)'}

{'generated_text': 'Herbimycin A-sensitive degradation of ErbB3. This inhibition of ErbB3 is due to an increased association of ErbB3 with its ligand heparin-binding EGF-like growth factor.
```

Fig 6.8 – Output with Sampling

6.3 Model Performance Evaluation:

Model performance was measured using ROUGE score. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of summaries generated by automatic summarization systems. These metrics are based on comparing the generated summaries with reference summaries created by humans.ROUGE-1, ROUGE-2, and ROUGE-3 are three variants of the ROUGE metric, which differ in the length of the n-gram used for comparison between the generated and reference summaries.

ROUGE-1 evaluates the overlap between the unigram (individual words) in the generated summary and the reference summary. It measures the recall of the generated summary, i.e., the percentage of important words from the reference summary that were included in the generated summary.

ROUGE-2 evaluates the overlap between the bigrams (pairs of adjacent words) in the generated summary and the reference summary. It measures the recall of bigrams in the generated summary, i.e., the percentage of important bigrams from the reference summary that were included in the generated summary.

ROUGE-3 evaluates the overlap between the trigrams (triplets of adjacent words) in the generated summary and the reference summary. It measures the recall of trigrams in the generated summary, i.e., the percentage of important trigrams from the reference summary that were included in the generated summary.

In general, a higher ROUGE score indicates that the generated summary is more similar to the reference summary, and hence, a better-quality summary. The ROUGE metrics are commonly used in research studies to compare the performance of different summarization systems or to tune the parameters of summarization algorithms.

In this paper, ROUGE-1 score was between 0.1 to 0.5 and the median value was 0.4. ROUGE-2 score was between 0.2 to 1 and the median value was 0.6 . ROUGE-3 score was between 0.3 to 1 and the median value was 0.8.

metric	**Rouge Score**		
	Rouge-1	***Rouge-2***	***Rouge-3***
r	0.450045	0.549370	0.763382
p	0. 458857	0.660528	0.775071
f	0. 446232	0.591211	0.759433

Table 6.1 – Rouge score metrics

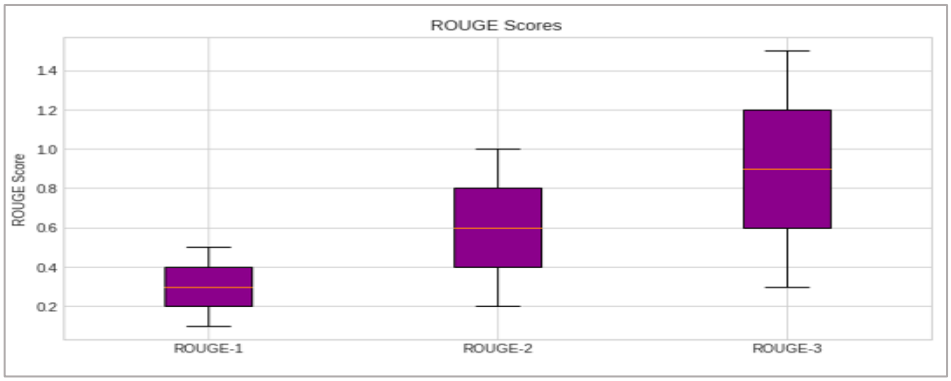


Fig 6.9 – Plotting Rouge score via box plot

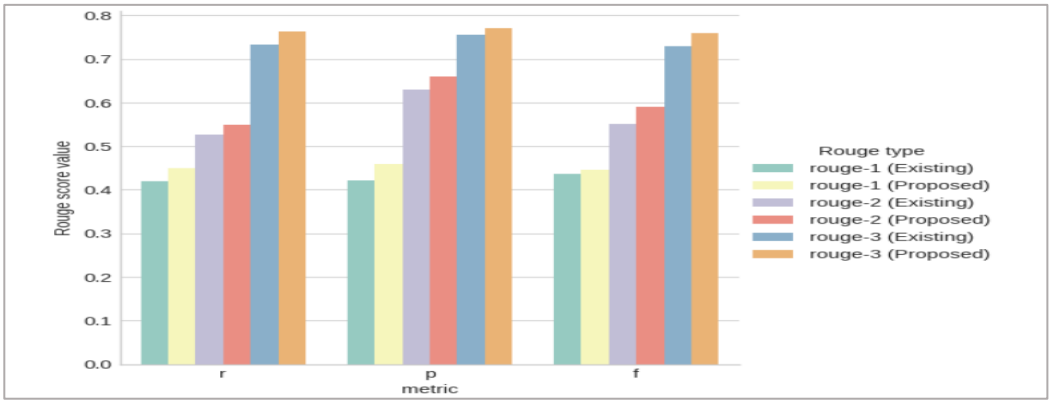


Fig 6.10 – Plotting Rouge score via bar graph (Comparison with base model)

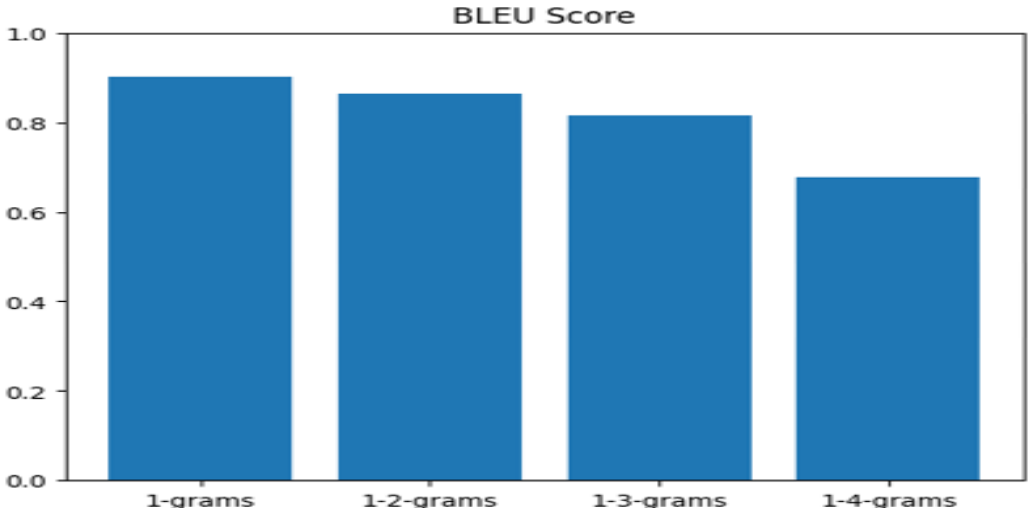


Fig 6.11 – Plotting BLEU score via bar graph

6.4 Model Comparison:

In this section, I have compared the loss with four different models which were GPT2 medium, GPT2 large, BioBERT and ClinicalBERT. Language model loss, multiple choice loss and total loss was calculated for GPT2 medium, GPT2 large model whereas total loss was calculated for the BioBERT and ClinicalBERT model as they didn't have the multiple-choice loss. Finally, total losses were compared with each other.

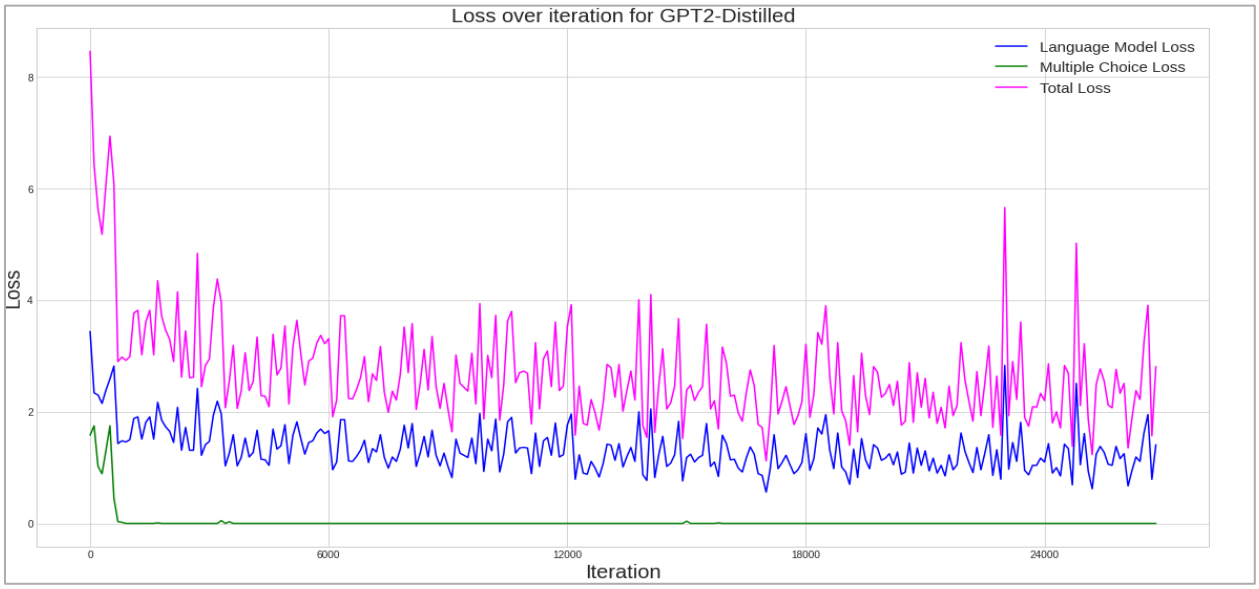


Fig 6.12– Loss of Distilled GPT-2

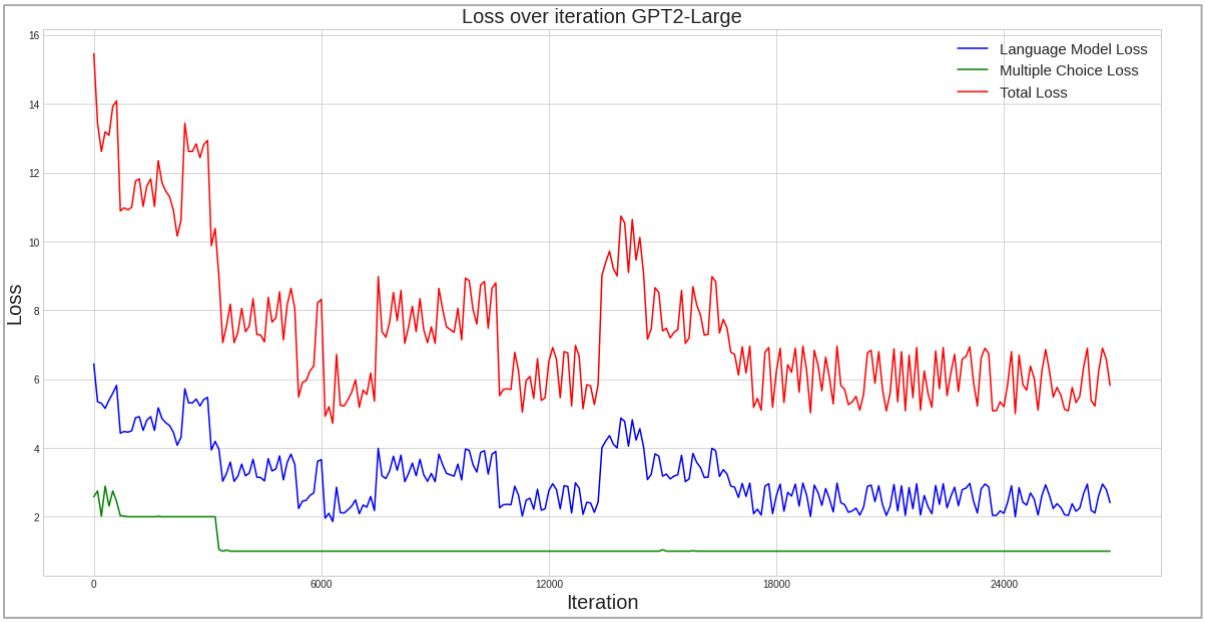


Fig 6.13 – Loss of GPT-2 Large Model

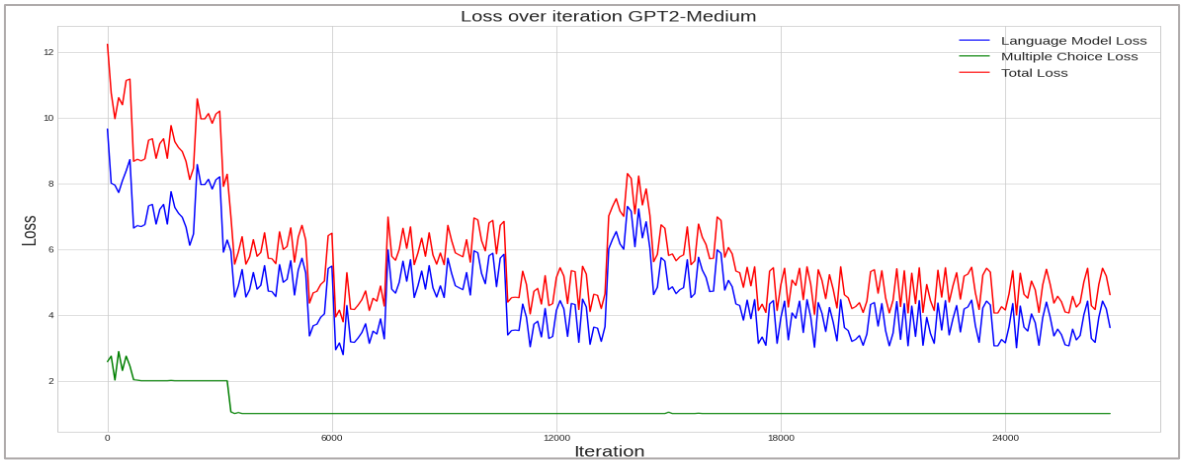


Fig 6.14 – Loss of GPT-2 Medium Model

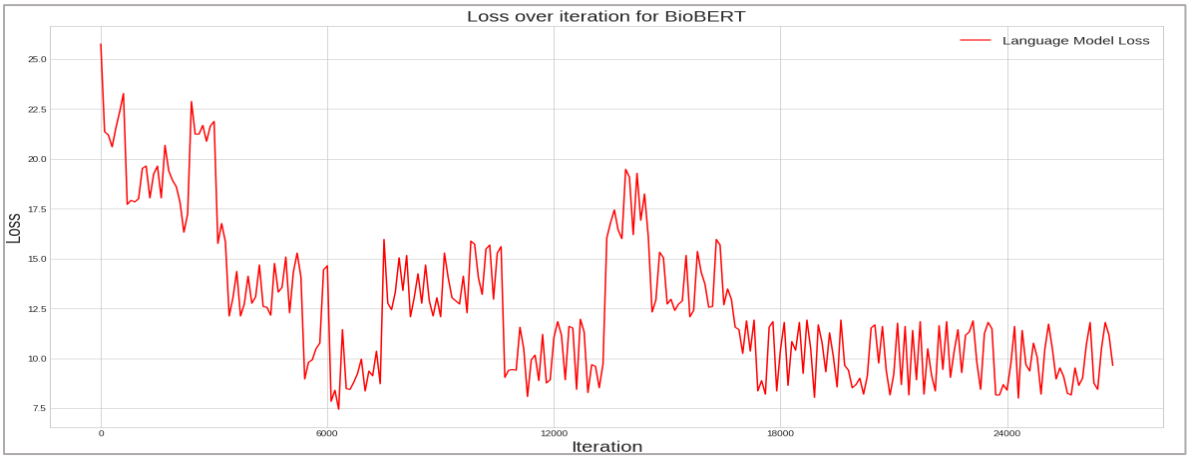


Fig 6.15 – Loss of BioBERT Model

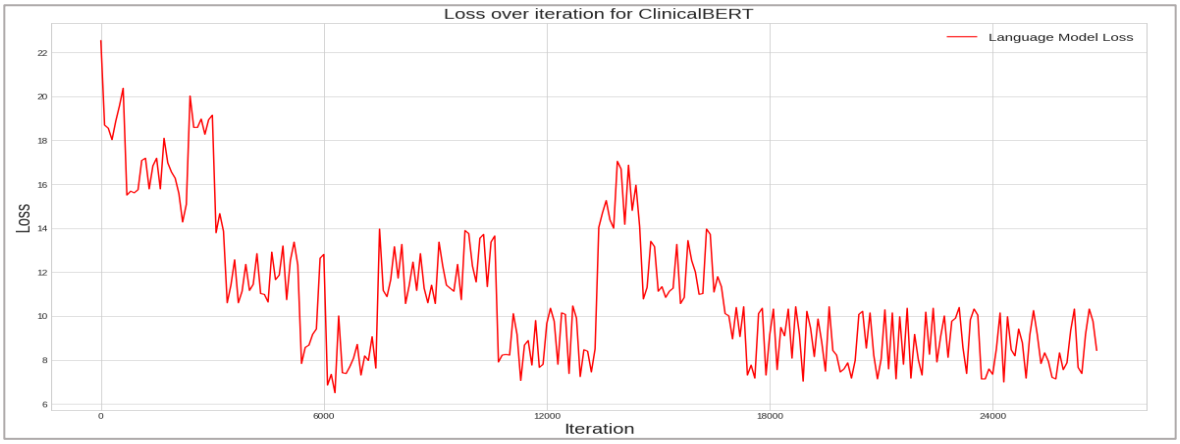


Fig 6.16 – Loss of clinicalBERT Model

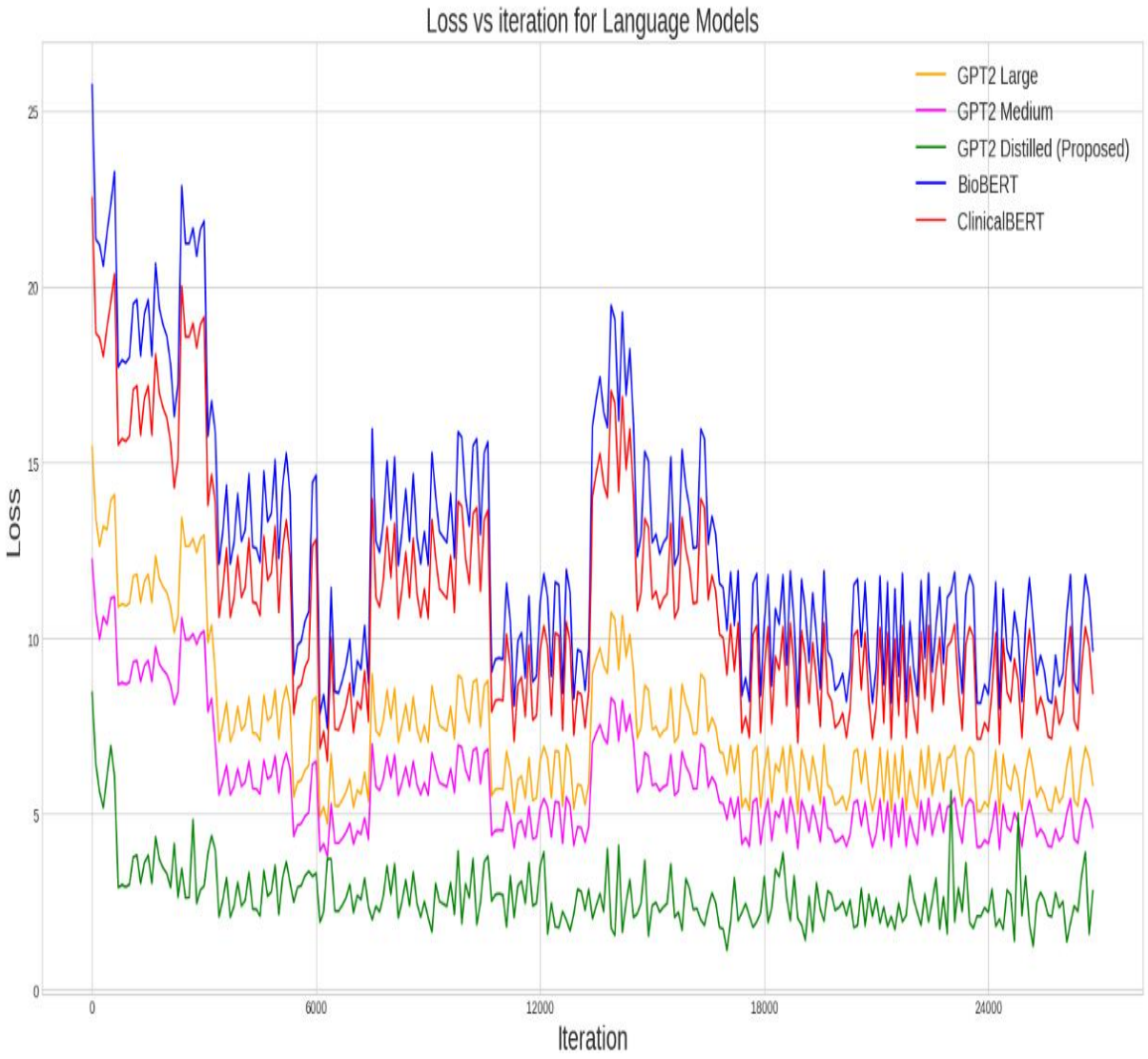


Fig 6.17 – Comparison of Loss with Distilled GPT-2 Model

From the graph we can see that the total loss for BioBERT is highest, it starts from 26 and ends at 10 whereas ClinicalBERT performs better than the BioBERT.

The total loss for ClinicalBERT starts at 23 and ends at 9. GPT2 large and GPT2 medium performance is slightly better than the BERT. For GPT-2 larger model loss starts at 15 and ends at 6 and GPT2 medium loss starts at 12 and ends at 4.

GPT2 Distilled out performed all the models. The loss starts at starts at 9 and ends at 1. The language model loss is in exponential form. After 600 iterations language model loss drop significantly. The multiple-choice loss drops to zero very quickly. Initially total loss was also high, and it followed the language model loss pattern and then drop significantly after 700 iterations.

7. CONCLUSION AND FUTURE SCOPE

Text generation is still a challenging task for deep learning NLP, especially when it comes to domain-specific corpora especially medical domain that differ from the pre-training data. Medical data are not easily available. There are few sites that publish public medical data.

I took the subset of PubMed data in few categories such as Diagnosis, Medicine, Prediction, Prognosis, Sepsis etc. To create the training data set for our text generation model, we merged 5 tensor objects, each containing 4 options in a multiple-choice format. These options corresponded to 4 keyword-summary pairings, with only one of the pairs being correct. The input sequences in the dataset had a shape of [4,1024], while the token type sequence was [4,1024]. Additionally, we included the gold abstract label for the generation task as [4,1024], the last token label as [4], and the multiple-choice answer as [1].The training dataset for the model being used for multiple choice tasks is composed of five tensor objects, which are multi-dimensional arrays of numerical values. These tensors are integrated to form the training dataset. The shape of input sequences in the tensor is [4,1024], indicating that it comprises 4 sequences of 1024 values each. The token type sequence tensor also has a shape of [4,1024], signifying that it contains 4 sequences of 1024 values each, used to identify the different token types in the input sequences. The tensor object used to store the gold abstract labels for the generation task has a shape of [4,1024]., indicating that it comprises 4 sequences of 1024 values each. These labels are used to signify the correct text for each input sequence. The last token label tensor has a shape of [4], it contains 4 values and is used to identify the last token of the input sequences. The multiple-choice answer tensor has a shape of [1], it contains 1 value and is used to identify the correct answer among the 4 choices. The model's objective is to predict the correct pair among these 4 choices.

While the model has not yet reached human-level performance, its results remain interpretable. We have measured the performance of our model with different GPT-2 model and BERT model. BioBERT and ClinicalBERT did not performed well as compared to GPT-2 medium and GPT-2 large. The total loss was more for BERT model as compared to GPT-2.

Among GPT-2 models, Distilled GPT-2 performed well. Distilled GPT-2 has the lowest loss among all. One potential method of improving could be to continue training the model using full PubMed data set along with AERS(Adverse Event Reporting System), VAERS(Vaccine Adverse Event Reporting System) and FAERS(FDA Adverse Event Reporting System) dataset. If we trained this model on diverse medical data then the quality of generated text will improve. The performance of the model could benefit if trained on larger datasets and by using higher GPU. To run this model on V100 GPU for 30k iterations, it took us around 30 hours, if we can use higher computation power and can run the model on entire dataset, the performance might be faster, and we can weed out those articles that have irrelevant information. We can use GPT-2 XL for training and fine tuning this required more computing resource.

References:

1. Virapat Kieuvongngam, Bowen Tan, and Yiming Niu. *Automatic text summarization of covid-19 medical research articles using bert and gpt-2.*arXiv preprint arXiv:2006.01997, 2020.
2. Luo, Renqian, Sun, Liai , Xia, Yingce , Qin, Tao Zhang, Sheng , Poon, Hoifung, Liu and Tie-Yan. *BioGPT: generative pre-trained transformer for biomedical text generation and mining.* Briefings in Bioinformatics, Oxford Academic, 2022.
3. Su, Nigel, Yixuan and Collier. *Contrastive search is what you need for neural text generation.* arXiv preprint arXiv:2210.14140, 2022
4. Chang, Ernie and Shen, Xiaoyu and Zhu, Dawei and Demberg, Vera and Su, Hui. *Neural data-to-text generation with lm-based text augmentation.* arXiv preprint arXiv:2102.03556, 2021
5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention is all you need.* corr abs/1706.03762 (2017). 2017.
6. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language models are unsupervised multitask learners*. 2018. URL- Language Models
7. Yang Liu and Mirella Lapata. *Text summarization with pretrained encoders.* arXiv preprint arXiv:1908.08345, 2019.
8. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the limits of transfer learning with a unified text-to-text transformer,* 2019.
9. Zhen Huang, Shiyi Xu, Minghao Hu, Xinyi Wang, Jinyan Qiu, Yongquan Fu, Yuncai Zhao, Yuxing Peng, and Changjian Wang*., "Recent trends in deep learning based open-domain textual question answering systems.,"* 2020.
10. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding,* 2018.
11. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. *Huggingface's transformers: State-of-the-art natural language processing,* 2019.
12. Derek Miller. *Leveraging bert for extractive text summarization on lectures*, 2019.
13. Dima Suleiman and Arafat Awajan. *Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges. Mathematical problems in engineering,* 2020.
14. Philipp Hartl and Udo Kruschwitz. *Applying automatic text summarization for fake news detection.* arXiv preprint arXiv:2204.01841, pages 1–8, 2022.
15. Abigail See, Peter J Liu, and Christopher D Manning. *Get to the point: Summarization with pointer-generator networks.* arXiv preprint arXiv:1704.04368, 2017
16. Dmitrii Aksenov, Julián Moreno-Schneider, Peter Bourgonje, Robert Schwarzenberg, Leonhard Hennig, and Georg Rehm. *Abstractive text summarization based on language model conditioning and locality modeling.* arXiv preprint arXiv:2003.13027, 2020.
17. Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter*, 2019
18. Shakshi Sharma, Ekanshi Agrawal, Rajesh Sharma, and Anwitaman Datta. *Facov: Covid-19 viral news and rumors fact-check articles dataset. In Proceedings of the International AAAI Conference on Web and Social Media,* volume 16, pages 1312–1321, 2022.
19. Ruixuan Zhang, Zhuoyu Wei, Yu Shi, and Yining Chen. *Bert-al: Bert for arbitrarily long document understanding,* 2019.

20. Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. *Cord-19: The covid-19 open research dataset,* 2020
21. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural machine translation by jointly learning to align and translate*. arXiv preprint arXiv:1409.0473, 2014.
22. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.
23. Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. *Transformer-xl: Attentive language models beyond a fixed-length context.* arXiv preprint arXiv:1901.02860, 2019.
24. Akbar Karimi, Leonardo Rossi, and Andrea Prati. *Adversarial training for aspect-based sentiment analysis with bert*. pages 8797–8803, 2021.
25. Mohamed El Ghaly Beheitt and Moez Ben Haj Hmida. *Automatic arabic poem generation with gpt-2*. pages 366–374, 2022.
26. Juan-Manuel Torres-Moreno. *Automatic Text Summarization*. Wiley, 2014.
27. Hritvik Gupta and Mayank Patel. *Method of text summarization using lsa and sentence based topic modelling with bert.* pages 511–517, 2021.
28. 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, 54(8):5789–5829, 2021.
29. Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. *On the use of bert for neural machine translation.* arXiv preprint arXiv:1909.12744, 2019.
30. Neel Kanwal and Giuseppe Rizzo. *Attention-based clinical note summarization. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing,* pages 813–820, 2022.
31. Milad Moradi, Georg Dorffner, and Matthias Samwald. *Deep contextualized embeddings for quantifying the informative content in biomedical text summarization*. Computer methods and programs in biomedicine, 184:105117, 2020.
32. Danqing Wang, Pengfei Liu, Ming Zhong, Jie Fu, Xipeng Qiu, and Xuanjing Huang. *Exploring domain shift in extractive text summarization.* arXiv preprint arXiv:1908.11664, 2019.
33. Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: *Evaluating generated text as text generation*. Advances in Neural Information Processing Systems, 34:27263–27277, 2021.
34. Anna Glazkova and Dmitry Morozov. *Applying transformer-based text summarization for keyphrase generation*. arXiv preprint arXiv:2209.03791, 2022.
35. Bhrugesh Joshi, Vishvajit Bakarola, Parth Shah, and Ramar Krishnamurthy. *Deepmine-natural language processing based automatic literature mining and research summarization for early-stage comprehension in pandemic situations specifically for covid-19*. bioRxiv, 2020.
36. Shakshi Sharma, Ekanshi Agrawal, Rajesh Sharma, and Anwitaman Datta. Facov: *Covid-19 viral news and rumors fact-check articles dataset.* In Proceedings of the International AAAI Conference on Web and Social Media, volume 16, pages 1312–1321, 2022.
37. Soham Poddar, Azlaan Mustafa Samad, Rajdeep Mukherjee, Niloy Ganguly, and Saptarshi Ghosh. *Caves: A dataset to facilitate explainable classification and summarization of concerns towards covid vaccines*. arXiv preprint arXiv:2204.13746, 2022.

38. Niculescu, Mihai Alexandru and Ruseti, Stefan and Dascalu, Mihai. *RoGPT2: Romanian GPT2 for Text Generation.* 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE,2021
39. Pascual, Damian and Egressy, Beni and Meister, Clara and Cotterell, Ryan and Wattenhofer, Roger, *A plug-and-play method for controlled text generation,* arXiv preprint arXiv:2109.09707, 2021.
40. Deng, Shumin and Zhang, Ningyu and Yang, Jiacheng and Ye, Hongbin and Tan, Chuanqi and Chen, Mosha and Huang, Songfang and Huang, Fei and Chen, Huajun, *LOGEN: few-shot logical knowledge-conditioned text generation with self-training,* arXiv preprint arXiv:2112.01404, 2021.
41. Bai, He and Shi, Peng and Lin, Jimmy and Tan, Luchen and Xiong, Kun and Gao, Wen and Liu, Jie and Li, Ming. *Semantics of the unwritten: The effect of end of paragraph and sequence tokens on text generation with GPT2.* arXiv preprint arXiv:2004.02251, 2020
42. Chan, Alvin and Ong, Yew-Soon and Pung, Bill and Zhang, Aston and Fu, Jie. *Cocon: A self-supervised approach for controlled text generation.* arXiv preprint arXiv:2006.03535, 2020
43. Gong, Heng and Sun, Yawei and Feng, Xiaocheng and Qin, Bing and Bi, Wei and Liu, Xiaojiang and Liu, Ting. *Proceedings of the 28th International Conference on Computational Linguistics.* Pages - 1978—1988, 2020.
44. Mager, Manuel and Astudillo, Ram{\'o}n Fernandez and Naseem, Tahira and Sultan, Md Arafat and Lee, Young-Suk and Florian, Radu and Roukos, Salim. *GPT-too: A language-model-first approach for AMR-to-text generation.* arXiv preprint arXiv:2005.09123, 2020.
45. Groenwold, Sophie and Ou, Lily and Parekh, Aesha and Honnavalli, Samhita and Levy, Sharon and Mirza, Diba and Wang, William Yang. *Investigating African-American Vernacular English in transformer-based text generation.* arXiv preprint arXiv:2010.02510, 2020.
46. McCoy, R Thomas and Smolensky, Paul and Linzen, Tal and Gao, Jianfeng and Celikyilmaz, Asli. *How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven*, arXiv preprint arXiv:2111.09509, 2021.
47. Rosati and Domenic. *SynSciPass: detecting appropriate uses of scientific text generation*. arXiv preprint arXiv:2209.03742, 2022.
48. Fang and Jingwu. *An Application of Customized GPT-2 Text Generator for Modern Content Creators.* University of California, Los Angeles, 2021.
49. Karpinska, Marzena and Akoury, Nader and Iyyer, Mohit. *The perils of using mechanical turk to evaluate open-ended text generation,* arXiv preprint arXiv:2109.06835, 2021.
50. Montesinos, Dimas Munoz. *Modern Methods for Text Generation.* arXiv preprint arXiv: 2009.04968, 2021.