Politecnico di Torino


Politecnico
di Torino


Master's Thesis

**Master of Science in**
**Communications and Computer Networks Engineering**

# Attention-based Summarization Approach of Clinical Notes

**Supervisor:**

**Prof. Maurizio Morisio**

**Candidate:**

**Rizvan Saatov**

**Internship Tutor:**

**Dott. Giuseppe Rizzo (LINKS Foundation)**

October 2021

## Keywords

Electronic Health Records (EHR), Clinical Notes, Recurrent Neural Networks, Artificial Intelligence, Attention Mechanism, MIMIC Dataset, Global Vectors, Word Embedding, BERT model, Machine Learning, Medical Records, Long Short-Term Memory, Text Processing, Transformer model, Natural Language Processing, Extractive summarization, Abstractive summarization

# Abstract

Deployment of Machine Learning for understanding clinical notes in the healthcare sector is crucial to extract meaningful phrases based on disease types. It is tough for human beings to summarize large documents of text manually. Summarization and evaluation of text are considered challenging tasks in the NLP community. We developed CUI Machine learning models to summarize clinical notes based on multi-head attention mechanisms and evaluate the summaries by applying evaluation metrics.

In this thesis work, we propose a multi-head attention-based mechanism to perform extractive summarization of meaningful phrases in clinical summaries from the MIMIC-III dataset.

This research helps highlight and perceive helpful information from clinical notes to a physician, and this step will increase treatment quality and support doctor's tasks. We conclude with optimal results compared to statistical-based models, proposing certain limitations and employing new evaluation metrics from a different perspective for future work. We achieved a provable score, which indicates that the BERT (Bidirectional Encoder Representations from Transformers) based model output accuracy is better than the statistical-based solution model and can be employed to summarize extractive summarization methods.

# Acknowledgements

I would like to express my great appreciation to my supervisor Dott. Giuseppe Rizzo (LINKS Foundation) for being available all the time to provide valuable feedback. His insightful and motivational feedback pushed me to sharpen my thinking and brought my work to a higher level.

I am affirming gratefulness to my supervisor, Prof. Maurizio Morisio, for being my supervisor and his availability on my research throughout the entire period.

I would like to thank my friend Neel Kanwal (University of Stavanger) for the guidelines and suggestions during my master's degree. Furthermore, his previous work was a tremendous support to carry on my further research.

I would like to pay my special regards and many thanks to my uncle Rahim Saatov for his full support and for sharing his great advice during my student timeline. I would also like to thank my family for their wise counsel, especially to my grandmother Rubaba Saatova and my uncle Dunyamil Saatov.

My appreciation also goes out to my friends for their encouragement and support all through my studies. Specially thanks to Moein Aghcheli and Mahammad Khalilov.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**RNN** Recurrent Neural Network

**MIMIC** Medical Information Mart for Intensive Care

**ICD** International Classification of Diseases

**EHR** Electronic Health Records

**HIPPA** Health Insurance Portability and Accountability Act

**ML** Machine Learning

**NLP** Natural Language Processing

**BOW** Bag Of Words

**CBOW** Continuous Bag Of Words

**TFIDF** Term Frequency Inverse Document Frequency

**GloVe** Global Vectors

**RL** Reinforcement Learning

**LSTM** Long Short-term Memory

**RNN** Recurrent Neural Networks

**GPU** Graphic Processing Unit

**ANN** Artificial Neural Network

**JSD** Jensen–Shannon Divergence

**KLD** Kullback–Leibler Divergence

**CUI** Character User Interface

**BERT** Bidirectional Encoder Representation from Transformer

**OOV** Out of Vocabulary

# CHAPTER 1

# Introduction

## 1.1 Overview

The advancement in our day-to-day life is because of the advent of the latest technologies implemented in different spheres, from using a smartphone to switch on a light to using a beard trimmer. Today, empirical research and experiments result from common phenomena. Some of them are unexplainable and different from previous trends. New technological resolution to these phenomena is more adaptable and cognitive now than they used to be decades ago. High speeding processing, memory performance, and electronic bottlenecks have been widening at a much higher speed. This solution produces space for Artificial Intelligence and Machine Learning to be superimposed on the technology landscape more firmly and apparently than before day by day.

Mostly, human communications nowadays are more text-based because of social media. Many algorithms can utilize natural language to extract meaningful understanding. The processing of this natural language for human-computer interaction is known as Natural Language Processing (NLP). It is used to understand commonly used language and feed it to algorithms. The involvement of NLP is generalized into two main streams, Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLP is a field of artificial intelligence that gives machines the ability to read, understand and derive meaning from human languages. Furthermore, NLP enables the recognition and prediction of diseases based on electronic health records and patient's speech; applying this capability helps us during our research work.

## 1.2 Health Records

The healthcare sector is one of the most crucial public care departments in any country. It monitors national health and governs the public health of the coming generations. An Electronic Health Record (EHR) is a digital version of a patient chart, an inclusive snapshot of its medical history. EHR is a longitudinal record of the patient's health information consisting of structured vitals, medications, labs, procedures and unstructured progress notes, discharge summaries, diagnostic test reports information like date of birth, height, weight, blood-pressure, blood-sugar level, and other suggestions to doctors [43]. In some cases, when this history becomes chronic, we have a lot of text related to prescriptions, medical

procedures, and discharge suggestions. EHR frameworks are supposed to keep track of information precisely in a timestamped manner. It helps to avoid the use of any allergic drug or practice in the future. It is substantially saving billions of dollars for any economy worldwide. One of the key features of an EHR is that health information can be created and managed by authorized providers in a digital format capable of being shared with other providers across more than one health care organization.

With the rising popularity of clinical notes for text-processing, National central bodies started making it safer for research purposes. HIPPA (Health Insurance Portability and Accountability Act) has allowed medical institutes and research organizations to help de-identified patient records publicly. [44]

## 1.3 Goal

The purpose of this work is to develop a model that can summarize clinical notes and their evaluation. The available notes are very vague and contain a lot of unnecessary details. That makes it harder for physicians' work, and it is time-consuming. Data is processed before feeding to a Machine Learning model. This research work is also an academic requirement of the master's degree program. This practice will facilitate in developing research-based knowledge of real-world problems in a state-of-the-art manner. This document will utilize the existing state-of-the-art frameworks and develop a CUI Machine Learning model based on the Attention mechanism. We used doccano as a web-based annotation tool to see the visualization of clinical notes from the domain experts' point of view. Evaluation of text summarization is considered challengeable in NLP. In our case, we are supposed not to use metrics such as F score, ROUGE, BlUE, or based on a similar idea since we do not have a gold standard summary for comparison. We discussed new applied metrics and their features in chapter 5 & 6.

The main objective is to evaluate the summary based on text content. Results are described in several evaluation metrics. The trained model will provide a massive application of automated disease labeling for physicians based on available notes. It can further post-process the text and highlight the crucial phrases out of it.

# CHAPTER 2

# Related Work

In this section, we discuss the core idea of text summarization and the concept of attention-based summarization. In this section, we discuss the core idea of text summarization and the idea of attention-based summarization and technology innovation in text summarization.

## 2.1 Basic Definitions

Artificial Intelligence and Machine Learning are pretty hot buzzwords these days and often used interchangeably. Artificial intelligence refers to the simulation of human intelligence in machines programmed to think like humans and mimic their actions. Artificial Intelligence encompasses several fields, such as Natural Language Processing, Deep Learning, Computer Vision, Speech Recognition, and more, all of which have the common goal of making machines even more intelligent than humans. Machine Learning is a subset of Artificial Intelligence and one of the techniques available for realizing AI. These Machine learning algorithms are usually categorized into three, supervised, unsupervised, and reinforcement learning.

### 2.1.1 Supervised Learning

In machine learning, Supervised Learning is done using ground truth, and we have prior knowledge of what the output values for our samples should be. Hence, supervised learning aims to learn a function that, given a sample of data and desired outputs, best approximates the relationship between input and output observable in the data. This approach is widely used in sentiment analysis, text classification, and spam filtering. Some most common algorithms are Support Vector Machine, Naïve-Based and Nearest Neighbour for classification and Linear Regression, Decision Trees, Neural Networks for Regression [45].

### 2.1.2 Unsupervised Learning

Unsupervised machine learning algorithms are used to group unstructured data according to its similarities and distinct patterns in the dataset. The term “unsupervised” refers to the fact that the algorithm is not guided like a supervised learning algorithm. The unsupervised algorithm works with unlabeled

data. Its purpose is exploration. If supervised machine learning works under clearly defines rules, unsupervised learning is working under the conditions of results being unknown and thus needed to be defined in the process. Unsupervised Machine Learning has three primary objectives, dimensional reduction, Clustering, and Association. This technique is simply used in Sentence segmentation, Machine Translation, and Dependency Parsing. Some standard algorithms are fuzzy logic, Bayesian Clustering, Hidden Markov Model, PCA (Principal Component Analysis and LDA (Linear Discriminant Analysis) [45].

### 2.1.3 Semi-Supervised Learning

Semi-supervised learning is an approach to machine learning that considers both labeled data and unlabeled data in a problem-solving process. Semi-supervised learning falls between supervised learning and unsupervised learning. This approach addresses a problem where most samples of the training are unlabeled, even though only limited data points with labels are available. A semi-supervised model aims to classify some of the unlabeled data using the labeled information set. The advantage of this is that many unlabeled data points are willingly available in several areas. Applications where semi-supervised learning is used are nearly the same as supervised learning [45].

Semi-supervised learning models are becoming widely applicable in scenarios across a large variety of industries. Let's explore a few of the most well-known examples: Speech Analysis, Protein Sequence Classification, Web Content Classification.

### 2.1.4 Reinforcement Learning

Reinforcement learning is a machine learning technique that focuses on training an algorithm following the cut-and-try approach. The algorithm (agent) evaluates a current situation (state), takes action, and receives feedback (reward) from the environment after each act. Positive feedback is a reward, and negative feedback is punishment for making a mistake. This approach is widely used in robotics, record management, and finance, where the prime goal is to develop a policy. Reinforcement Learning helps agents learn by witnessing the available behaviors and their conduct using only an evaluative response called the return. The policy‘s ultimate goal is to increase its long-term success. Few well-known algorithms are Q-learning, SARSA, Deep-Q Network [46].

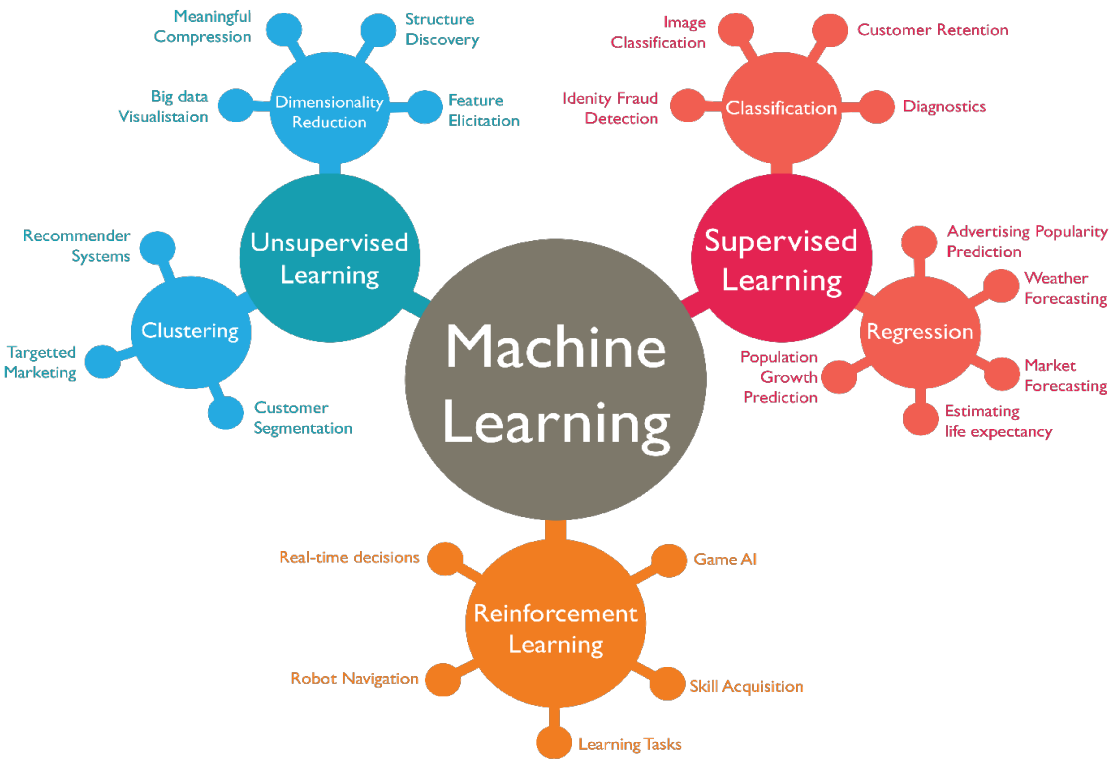


Figure 2.1: Description of Machine Learning types, taken from [65]

## 2.2 Natural Language Processing

NLP [19] is a field in machine learning with the ability of a computer to understand, analyze, manipulate, and potentially generate human language. NLP combines computational linguistics-rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in text or voice data and 'understand' its whole meaning. Some of the applications of Natural Language Processing are the following:

- Speech
  - Speech Recognition
  - Text-to-speech
- Syntax
  - Lemmatization
  - Stemming
  - Parsing
  - Part-of-speech Tagging
- Semantics
  - Lexical Semantics

  - Machine Translation
  - Natural Language Generation – Question Answering
  - Sentiment Analysis
  - Named Entity Recognition

- Discourse
  - **Text Summarization**

### 2.2.1 Text Summarization

Classification of text summarization can be one in two ways extractive and abstractive summarization. Extractive summarization uses statistical and linguistic features to determine the essential features and fuse them into a shorter version. Whereas abstractive summarization understands the whole document and then predicts the summary [20].
A summary can be defined as a document that conveys valuable information with significantly less text than usual [21]. A summary is thus helpful as it saves time as well as retrieves extensive document data. Especially summaries of clinical notes save doctors time. By using text summarization features, they can save more time and more lives. A good summary should be indicative as well as informative [22]. Meaningful summaries point out some crucial parts while an instructive emphasizes the vital information in a document.

As we refer to above, the two main approaches used in summarization are extractive and abstractive. The abstractive approach consists of understanding the source text and outputs the precise and concise summary using linguistic methods to interpret and examine the text and usually requires advanced language generation and compression techniques [23].

Extractive summaries do not focus on the understanding of the text. It extracts the most crucial part based on statistical and linguistic features such as cue words, location, word frequency [24].

### 2.2.2 Methods for Extractive Summarization

Extractive summaries [26] [27] [28] maintains the redundancy by extracting the relevant features from the document. The early methods focus on selecting top sentences based on some greedy algorithms and aim for maximizing coherence and minimizing redundancy [25]. These methods can be further generalized as follows:

- **Cohesion-based Approach:** Cohesion tries to account for relationships among the elements of a text. Lexical chains are used to find a relation between two sentences. (Brin and Page, 1998) [35] proposed a co-reference system that uses cohesion for this purpose in web search.

- **Corpus-based Approach:** It is a frequency-based approach where common words that are often repeated do not carry salient information. It relies on an information retrieval paradigm in which common words are considered as query-words. SumBasic [34] is a similar centroid-based method that uses word probability, and the words in each centroid with higher probabilistic values are selected for the summary.

- **Graph theoretic Approach:** Graph-theoretic representation of passages provides a method of identification of themes. Identification of themes is done [30]. The documents are represented as nodes when preprocessing, stop word removal, and stemming is done. There is a node for every sentence. If two sentences share common information, then they are connected by edges. A few popular algorithms like HITS [31] and Google's PageRank [32] provided base for graph-based summarization. It helps to visualize intra topic similarity where nodes present several topics, and vertices show their cosine similarity with sentences [33].

- **Rhetoric-based Approach:** This method relies on forming text organization in a tree-like representation [36] and [37]. Then, text units are extracted based on their position close to the nucleus.

- **Machine Learning Approach:** Machine learning models are outperforming for nearly all kinds of tasks, including text summarization. Here sentences are classified based on some features as the summary and non-summary sentences. A training data is fed to the system [29] based on probabilities using Baye's. Neural networks better exploit hidden features from the text. Attention mechanism coupled with convolution layers helps to select importing phrases based on their position in the document. Machine learning models for summarization treat this task as a classification task.

## 2.3 Fundamentals of Text-Processing

Text processing is a staging process, which requires filtering text according to an end-to-end pipeline. This pipeline comprises tokenizing raw text, cleaning punctual and other signs, vectorizing text, applying the desired model for training, validating the model, and filtering desired results.

### 2.3.1 Tokenization

Tokenization is a beneficial process before applying NLP methods. It is simply breaking a text chunk into smaller parts. Whether it is breaking the paragraph in sentences, sentences into words, or words in characters, these tokens help understand the context or develop the model for the NLP. Many built-in methods are used for splitting sentences of clinical notes based on spaces, punctuations, medical notations, and symbols. It is also known as linguistic analysis. There are different methods and libraries available to perform tokenization. NLTK, Spacy, Gensim, Keras are some of the libraries that can accomplish the task.

### 2.3.2 Text Cleaning

After the text is split, Some most commonly repeated words, which are known as stopwords, are removed. These tokens do not contain much contextual meaning and are often repeated for grammatical purposes. Stemming and Lemmatization are known as text normalization techniques. Stemming is a rule-based approach, and it strips inflected words based on common prefixes and suffixes. These words are generally inflected in addition to the present form of the verb.

Lemmatization, unlike Stemming, reduces the inflected words properly, ensuring that the root word belongs to the language. So it links words with similar meanings to one word. Text preprocessing includes both Stemming as well as Lemmatization. Many times people find these two terms confusing. Some treat these two as the same. Lemmatization is preferred over Stemming because lemmatization refers to doing things properly using a vocabulary and morphological analysis of words.

### 2.3.3 Vectorizing Text

Computers can only understand and process numbers, so these cleaned tokens need to be converted into vectors of numbers reasonably. This process of converting text into numbers is known as vectorization or embeddings. Word embeddings or word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers, which is used to find word predictions, word similarities. There are various techniques available for this task.

#### Word Embedding

Word embeddings are a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. One of the basic ideas to get a representation is the One-Hot Encoding that is a simple method to vectorize categorical variables like words. During in One-Hot Encoding, each word is independent; the idea of word embedding is to generate distributed representations of the words taking into account their dependency between one another; therefore, words with the same meaning will have a similar representation. We will describe some of those algorithms here, such as Word2vec, GloVe, and FastText.

#### One-Hot Encoding

One Hot encoding is a representation of categorical variables as binary vectors. Each integer value is represented as a binary vector with zero values except the integer's index, which is marked with a 1. Some token may appear very regularly, whereas some may be less common. This creates a need to present words in a one-dimensional vector. This form of vectorization takes categorical data and returns numerical binary data for it. Here (1) shows the presence, and (0) poses the absence.

For Instance:

"Yellow" ===> [1,0,0,0]
"Orange" ===> [0,1,0,0]
"BLACK" ===> [0,0,1,0]
"WHITE" ===> [0,0,0,1]

0ne-hot encoding eliminated the tokens distribution disparity and is most appropriate for shorter documents with fewer repetitive words.

#### Bag-of-Words Model

The bag-of-words model is a simplifying representation used in Natural Language Processing. It predicts the current word based on context. It builds a vocabulary from a corpus of documents and counts

how many times the words appear in each document. To put it another way, each word in the vocabulary becomes a feature, and a vector represents a document with the same length of the vocabulary, a "bag of words."

For Example: "Our the best time. Our the best time was when walking without mask."

Here we have 9 unique words. We will make a frequency vector with one-position for the corresponding word.

"Our" ===>[ 1 0 0 0 0 0 0 0 0 ]
"the" ===>[ 0 1 0 0 0 0 0 0 0 ]
"best" ===>[ 0 0 1 0 0 0 0 0 0 ]
"time" ===>[ 0 0 0 1 0 0 0 0 0 ]
"was" ===>[ 0 0 0 0 1 0 0 0 0 ]
"when" ===>[ 0 0 0 0 0 1 0 0 0 ]
"walking" ===>[ 0 0 0 0 0 0 1 0 0 ]
"without" ===>[ 0 0 0 0 0 0 0 1 0 ]
"masks" ===>[ 0 0 0 0 0 0 0 0 1]

Now to vectorize the sentence, we present the count on the corresponding position in the frequency vector.

"Our time was when walking without masks" ==> [ 1 1 0 0 1 1 1 1 1]

There comes a problem when corpus increases and we have a large vocabulary. This results in a vector with a higher dimension and a lot of zeros for less frequent words. It requires memory resources and makes computation inefficient. A different approach of k-gram can be used in this case to combine multiple common recurring words and use on position for them. [48]

**Term-Frequency (TF-IDF)**

TFIDF, short for term frequency-inverse document frequency, is a numerical statistic reflecting how important a word is to a document in a collection or corpus. A concern with scoring word frequency is that in the text, prevalent words tend to dominate (e.g., more excellent score) but may not contain as much "information content" as rare but perhaps domain-specific words to the model. One solution is TF-IDF takes an approach to rescale the frequency of the common words based on how often they appear in all present documents in a given corpus so that the scores for frequent words like "the," "is" that are also frequent across all the documents are penalized, for which it utilizes the concept of Term-frequency- Inverse document frequency. [51]

TF-IDF considers the relative frequency of tokens in the database in other corpus documents against their size. Term Frequency tf(t,d) is the number of times that term t occurs in document d, but there are many variants with different weights. Generally, term frequency is scaled logarithmically to prevent bias caused by longer documents or terms that appear much more frequently concerning other terms: tf(t;d) = 1 + log(ft;d). Inverse document frequency idf(t, D) is the logarithmically scaled inverse fraction of the documents that contain the word obtained by dividing the total number of documents (N) by the number of documents containing the term ($n_t$), and then taking the logarithm of that quotient $\log(N/n_t)$.

Word2Vec is short for Words To Vector. Word2vec is a technique for Natural Language Processing. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. The TF-IDF is the product of two statistics, term frequency and inverse document frequency. Term Frequency is a scoring of the frequency of the word in the current document. Inverse Document Frequency is a scoring of how rare the word is across all the documents in a corpus. It measures how important a term is. There are various ways of determining the exact values of both statistics.

$$TF - IDF(t,d,D) = tf(t,d).idf(t,D) = \log(N/n_t).(1 + log(ft,d))$$

The result of each term lies between [0,1] where term closer to 1 is more meaningful and vice versa.

For Instance:

The boy and the girl play ===> [0.43 0.00 0.32 0.43 0.00 0.43 0.64]

**Word2Vec Model**

Natural language processing systems treat words as a small unit. NLP systems have constraints to calculate this similarity, especially when there are billions of tokens. Word2Vec was introduced by Mikolov et al. (2013) [48]. It is a deep learning technique where a two-layer neural network takes input and produces a corresponding vector space. Essentially, Word2Vec positions the word in the pre-trained model's space so that its meaning determines its location, i.e., words with similar meanings are grouped together. The distance between two words also has the same meaning, such as word embeddings, which essentially represent words into numerical forms and represent the words with similar meanings or ones that are semantically similar in a common manner. For Example, Words share the same analogy as "Turin is to Italy as Madrid is to Spain." The vector $V_{Turin}$, $V_{Italy}$, $V_{Madrid}$ and $V_{Spain}$ are encoded in a fashion that it reserved their semantic meaning.

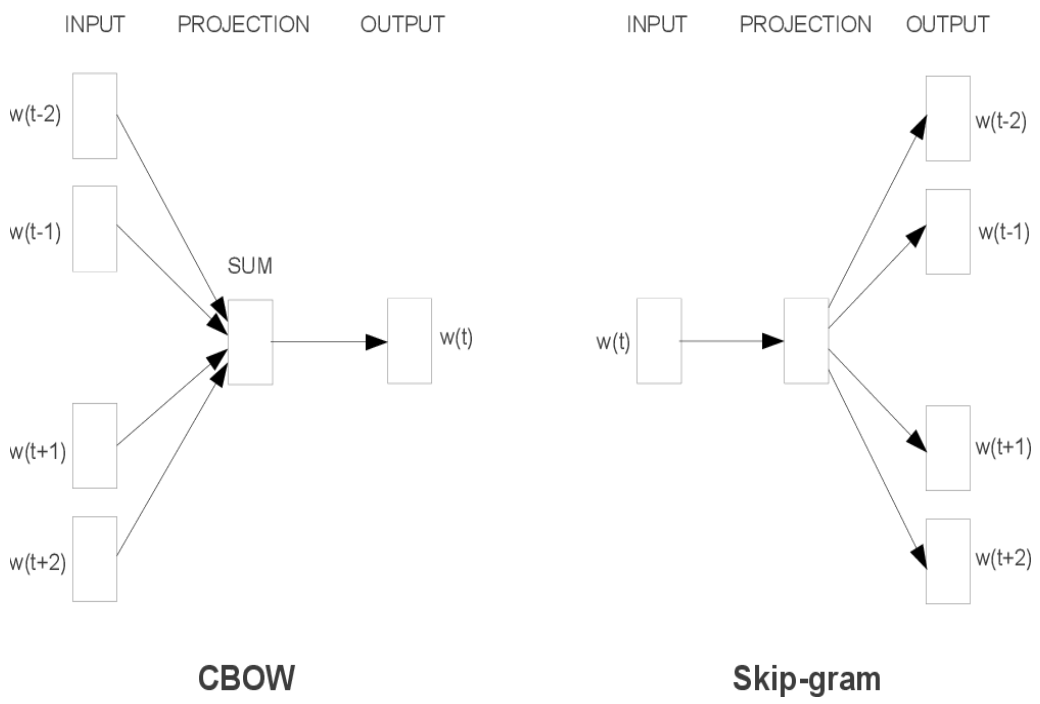


Figure 2.2: Word2Vec models[48]

Word2vec uses two different training methods involving a neural network such as:

- Common Bag Of Words (CBOW)
- Skip-Gram

**Continuous Bag-of-Words (CBOW)**

In this method, we find the probability of focused words given the context. The CBOW model is a method that learns embeddings by predicting a word as a target and taking its context as input. In practice, this becomes very inefficient when working with a large set of words.

In figure 2.3 , Input Layer x is an encoded vector of x = $\{x_{1k}, x_{2k}, \ldots, x_{Ck}\}$ where C is the context window and V represents vocabulary size. All input layer is forwarded to a hidden layer with a particular weight matrix $W \in \mathbb{R}^{V \times N}$ Where N is a number of the hidden layers. Afterward, Hidden layers are few to output layer with weight matrix $W' \in \mathbb{R}^{V \times N}$. This procedure takes in the background window the W rows corresponding to the words vocabulary indexes (this is a function of x's one-hot encoding) and averages them. Later used for computation at the output layer.

$$u_j = hv'_w j$$

$$h = \frac{1}{C}(\sum_{i=1}^{C} X_i)W$$

$$y_j = p(w_y j | w_1, ..., w_C) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_j')}$$

The final output $y_j$ is computed by passing $u_j$ through the softmax function. Loss Function here is to minimize the conditional probability.

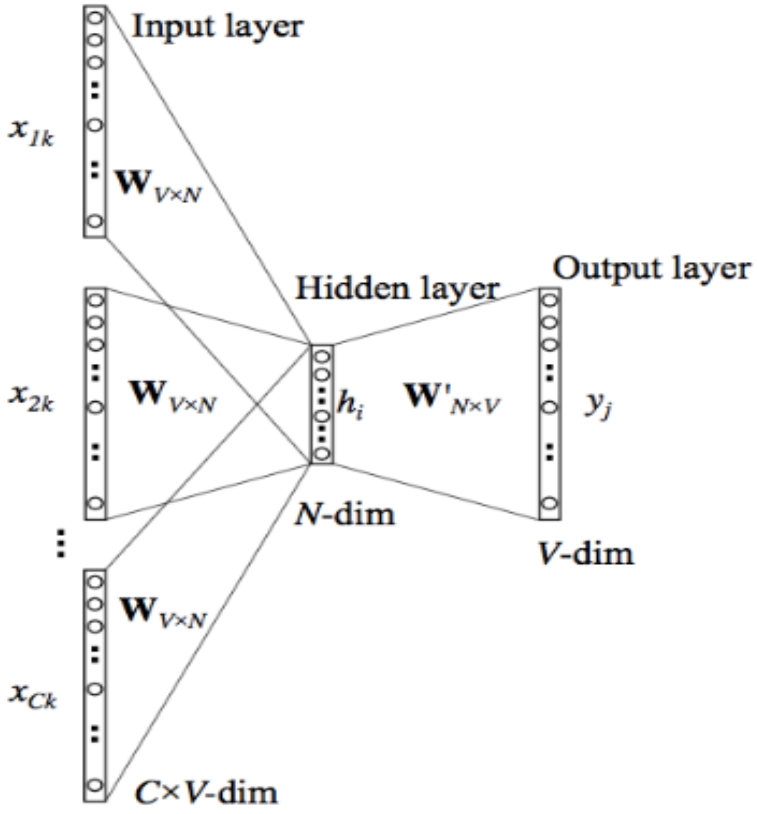


Figure 2.3: CBOW Model from Original Paper[48]

$$\zeta = -\log(p(w_0|w_1))$$
$$\zeta = -u_j * -\log(\sum_{j`=1}^{V} exp(u_j`))$$

**Continuous Skip-gram Model**

Here we find the probability of the corpus-based on focus words. We use negative sampling here to avoid softmax computation. Spearman correlation is used to compare the ranked list with cosine similarities. The Skip-Gram model is suitable for larger datasets.

Figure 2.4 shows that this is opposite to CBOW as defined before. Here we predict the output from hidden layers. Hidden Layers compute h = x.W, and output hidden layer is forwarded as

$$u_{c,j} = u_c = hv`_{wj} \forall j \in \{1,2,...,C\}$$

Loss function changes here because of C nominal distribution.

$$\zeta = -\log(p(w_{0,1}, w_{0,2}, ..., w_{0,C}|w_I))$$
$$\zeta = -\sum_{c=1}^{C} u_{c,j} * + C\log(\sum_{j`=1}^{V}) exp(u_j`)$$

Output Layer results:

$$y_{c,j} = p(w_c j = w_{0,\mathrm{c}}|w_I) = \frac{exp(u_{\mathrm{c,j}})}{\sum_{j`=1}^{V} exp(u_{\mathrm{j}`})}$$

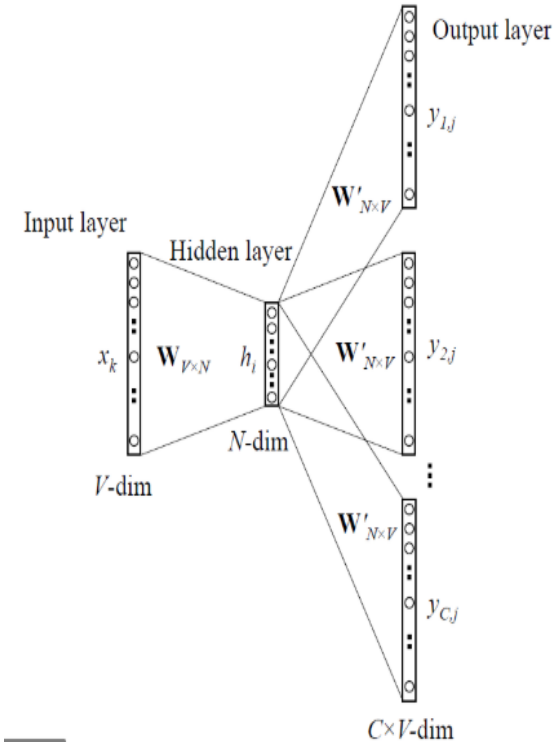


Figure 2.4: Skip-gram Model from Paper [48]

Both algorithms are implemented using a 3-layer neural network in the Tensorflow library, which includes a hidden layer. This makes it faster to train the models.

**FastText**

FastText is a vector representation technique developed by Facebook AI research, released in 2016 [49] [52]. As its name suggests, it's a fast and efficient method to perform the same task, and because of the nature of its training method, it ends up learning morphological details. FastText splits words into several n-grams (sub-words) instead of feeding individual words into the Neural Network. For example, the tri-grams for the word apple are "app," "ppl," and "pl" (ignoring the beginning and end of the word

boundary). The term enclosing vector for apple will be the sum of all these n-grams. Figure 2.5 shows the structural form of FastText.

$X_i$ is the n-gram feature that will be converted to (one-hot encoded representation) embedded and averaged from the hidden variable. This design is similar to the paper [48]. CBOW model (2013) with a label that is replacing the middle word. A softmax function calculates probability distribution over several classes but when the number of classes is large, computing the linear classifier is computationally expensive. [49]

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n))$$

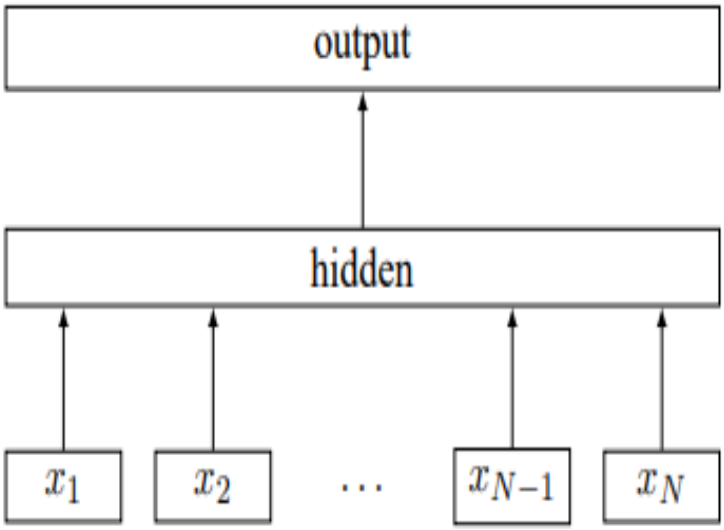


Figure 2.5: FastText Architecture from paper [52]

A is lookup Matrix, B is Output transformation, and hierarchical softmax is applied to their product in the equation above. Often, when looking for the most likely class, the hierarchical softmax is beneficial at test time. The likelihood that the path from the root to this node is associated with each node. So a node is always less probable than the parent's probability. The trade-off of the FastText model is in technical requirements, its memory consumption grows with the number of n-grams, and it usually takes a long way to train in comparison to GloVe and Word2vec.

**GloVe (Global Vectors)**

Word2vec was mainly based on preserving semantic analogies on basic algorithms using Neural Networks. It was a window-based approach and had the demerit of ignoring global statistics. The glove is based on the matrix factorization technique on the word context matrix. It first constructs a large matrix of co-occurrence information, i.e., for each word, you count how frequently we see those words in some context in a large corpus. This presence of globals GloVe ideally works better. GloVe, 2014 introduced by

Pennington et al. [50] at Standford was a log-bilinear model combing local context- matrix factorization such as Latent Semantic Analysis (LSA).

The count matrix in the case of GloVe is preprocessed by standardizing numbers and smoothing them. The GloVe makes the parallel implementation compared to word2vec, making it much easier to train more data. This produces the word vectors, which operate well on word comparison and similarity tasks and identify entities. Matrix factorization methods for making low-dimensional representations of words have origins that stretch back to Latent Semantic Analysis.

The first step is to create a matrix for co-occurrence. Calculation of the matrix with a fixed window dimension (words are jointly valid when they appear together in the same window) considers the local context. The GloVe is a count-based model, whereas count-based models learn vectors by making dimensionality reduction on a co-occurrence counts matrix. On the other hand, Word2vec is a predictive model. The principle of GloVe is that the relationships of co-occurrence between two words in a sense are strongly related to their meaning. Instead of the probabilities of themselves, the correct starting point should be the term vector learning with the coexistence likelihood ratios. The most popular model takes the form of the $P_{ik}$ / $P_{jk}$ ratio, which relies on three terms i,j,& k.

Here, w $\in \mathbb{R}$ and the information present the ratio $P_{ik}/P_{jk}$ in the word vector space. $P_{ij}$ indicates the probability of the term j to appear in context with i and can be calculated as

$$F(w_i, w_j, \hat{w_k}) = \frac{P_{ij}}{P_{jk}}$$

$$P_{ij} = \frac{X_{ij}}{X_i}$$

In the equation above X represents co-occurrence matrix and $X_i$ which is the total number of words that appeared in context. $X_{ij}$ denotes i,j th term in the matrix.

F is a function that takes the embedding of the words i,k,j as input. The number of possibilities for F is vast, but by enforcing a few desiderata, we can select a unique choice. Since one of GloVe's goals is to create word vectors that express meaning using simple arithmetic (vector addition and subtraction), a function F must be chosen to match this property with the resulting vectors. Considering that vector spaces are essentially linear structures, with vector differences, the most natural way of doing so is to restrict our consideration to those F functions which only depend on the difference between the two target terms. The simplest way to do this is to calculate the input to F as the disparity between the compared vectors.

$$F(w_i - w_j, \hat{w_k}) = \frac{P_{ij}}{P_{jk}}$$

In the case of word-word co-occurrence matrices, the distinction between the word and the meaning word is arbitrary, and we are free to swap the two functions. To do so reliably, we will swap X $\implies$ $X_T \& w \implies \hat{w}$.

$$F((w_i - w_j)^T, \hat{w_k}) = \frac{F(W_i^T \hat{w_k})}{F(W_j^T \hat{w_k}}$$

The ratio can be transformed into a subtraction of probabilities by taking the logarithm of likelihood ratios. A bias term is for each word to consider that some terms occur more often than others. These operations' products can later be converted into an equation over a single entry in the co-occurrence matrix.

$$w_i * \hat{w_k} + b_i = \log(P_i k) = \log(X_i k) - \log(X_i)$$

$$w_i^T \hat{w_k} = \log(P_i k) = \log(X_i k) - \log(X_i)$$

this term is independent of k so it can be absorbed into a bias $b_i$ for $w_i$. Finally, adding an additional bias $\widetilde{b}_k$ for $\widetilde{w_k}$ restores the symmetry.

$$w_i^T \hat{w_k} + b_i + \hat{b_k} = log(X_i k)$$

A major drawback to this model is that all co-occurrences are measured equally, even those that rarely or never occur. To avoid this a weighted least squares regression model by introducing a weight function into Objective Function J. Training is aimed at minimizing J.

$$J = \sum_{i,j=1}^{V} f(X_i j)(w_i^T \hat{w_j} + b_i + \hat{b_j} - \log X_i j)^2$$

Here V is the Vocabulary and Weighted function must obey three rules:

- If function f is viewed continuous then it should vanish faster as square-log limit tends to infinity f(0)=0.
- F (x) should be non-decreasing in order to prevent over weighting with unusual co-occurrences.
- For large values of x, f(x) should be relatively small, so that repeated co-occurrences are not over-weight.

$$f(X_i k) = min(1, (\frac{X_{ik}}{x_{max}})^{\alpha})$$

This function slashes the output of exceedingly common word pairs (where $X_{ij} > x_{max}$) and simply returns one. Since this number is always smaller than the total number of matrix entries, the model scales no worse than O( $||V||^2$)[50].

## 2.4 Recurrent Neural Networks

This section will go through Recurrent Neural Networks and their features and limitation for our work. There are three significant kinds of Neural Networks.

1. Feedforward Neural Network
2. Convolution Neural Network
3. **Recurrent Neural Network**

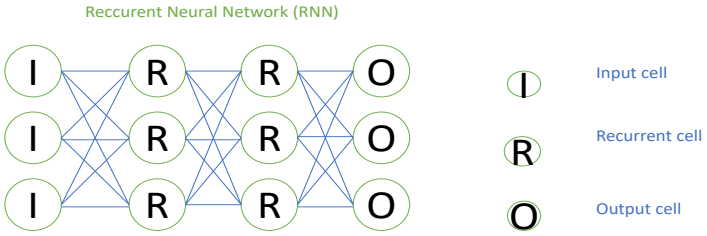


Figure 2.6: Recurrent Neural Network representation

Recurrent neural networks are a variant of feedforward networks. As shown in Figure 2.10, there are hidden layers; each neuron in hidden layers receives an input with a specific delay in time by using recurrent Neural networks to access previous information in current iterations. With these features, our model can train more and learn more. We can say while processing RNN remembers everything of earlier elements. RNN is used to work with sequential data, and in sequential data, there is a lot of information present in the data and the sequence of the data. So in RNN, we have these loops that allow information to persist through time. RNN uses a backpropagation algorithm for training data. The Backpropagation algorithm is a powerful algorithm to train a neural network, and we have discussed it in the next section. RNNs suffer from the problem of vanishing gradients, which hampers the learning of long data sequences. Furthermore, why do RNNs suffer from vanishing gradients? And we have explained Long Short-Term Memory (LSTM), which solves the vanishing gradient problem in the following section.

### 2.4.1 Backpropagation algorithm

The backpropagation algorithm is probably the most fundamental building block in a neural network. Backpropagation is an algorithm of training neural networks to perform tasks more accurately. The algorithm use gradient descent to minimize error in the predictions of Machine Learning models. This will calculate the gradient of the error function of any given error function and an artificial neural network while considering the different weights within that neural network. The Gradient Descent serves to find the minimum of the cost function. After the RNN outputs the prediction vector, we compute the prediction error and use the backpropagation through time algorithm to compute the gradient. [3] Nonetheless, RNNs suffer from the problem of vanishing gradients, which hampers the learning of long data sequences. The gradients carry information used in the RNN parameter update; when the gradient becomes smaller and smaller, the parameter updates become insignificant, which means no real learning is done. In the following section, we discussed the details of the Vanishing Gradient Problem.

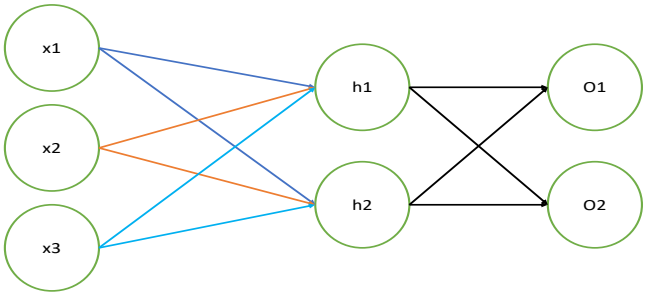


Figure 2.7: Backpropagation example

**Summary: the equations of backpropagation**

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \quad \text{(BP3)}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \quad \text{(BP4)}$$

Figure 2.8: The equations of backpropagation [67]

### 2.4.2 Vanishing Gradient Problem

Plain recurrent networks greatly suffer from the vanishing gradient problem. This problem makes it hard to learn and tune the earlier layers' parameters in the network. This problem becomes worse as the number of layers in the architecture increases. As more layers using certain activation functions are added to neural networks, the loss function gradients approach zero, making the network hard to train. One solution to the Vanishing gradient problem is LSTM (Long Short Term Memory) [47]. Long short-term memory (LSTM) neural networks solve the vanishing gradient problem by allowing network cells to forget part of their previously stored memory.

## 2.5 Long Short-Term Memory

Long-Short term memory (LSTM) network is a variant of recurrent neural networks (RNN), capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) [40]. LSTM is efficient here because of a long diagnosis text document. Hidden representation for each input is obtained with character level LSTM and Word Level LSTM. Character level encoding is expected to catch better features because of their many medical terms with the same suffix. The input is the sequence of sentence matrices in the word-level LSTM, each obtained by concatenating the word representations derived from the previous layer. The last hidden state is the interpretation of each sentence.

In this paper [41] they were proposed an upper bound on extractive summarization of discharge notes (MIMIC-III) and developed an LSTM model.

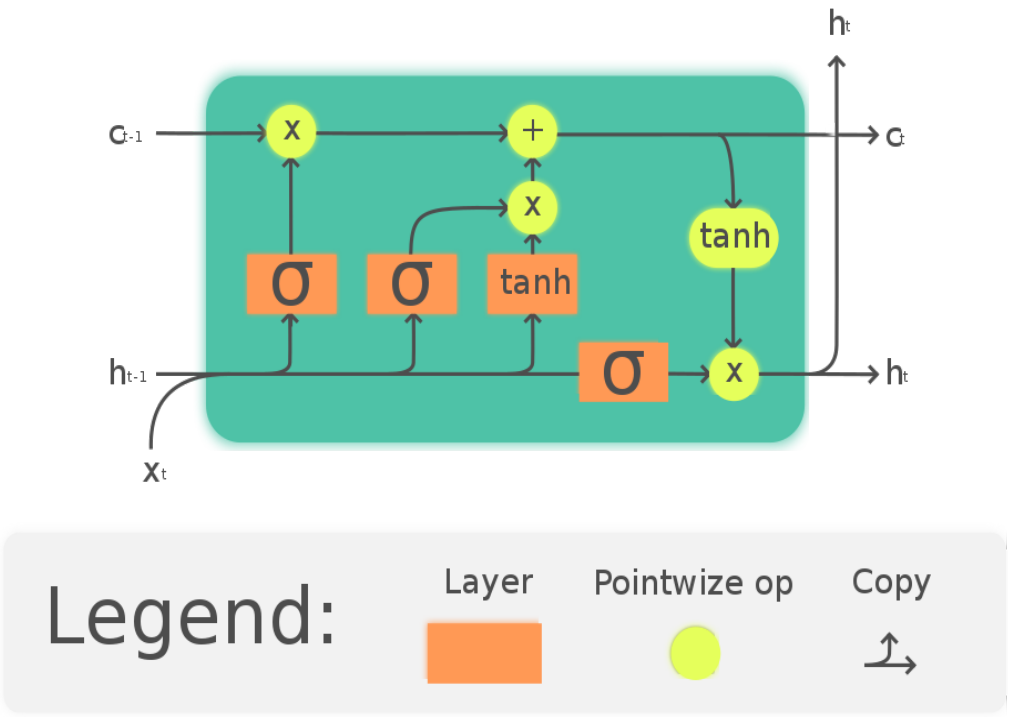


Figure 2.9: The Long Short-Term Memory (LSTM) cell can process data sequentially and keep its hidden state through time. [42]

## 2.6 Transformer model

In this section, we discuss the Transformer model and its preference. The paper Attention Is All You Need [10] introduces a novel architecture called Transformer. As the title indicates, it uses the attention-mechanism we will see later in the next section. The transformer is an architecture for transforming one sequence into another one with the help of two parts, encoder and decoder, the strength of which is founded upon its self-attention mechanism. Compared to RNN and LSTM sequence to sequence models, as we described in previous sections, it can capture deeper, finer-grained features and superior parallelism. It has been widely used as a powerful baseline in neural machine translation and has also been applied to text summarization. [39]

The Encoder is on the left, and the Decoder is on the right. Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times, described by Nx in the shown Figure 2.10. We see that the modules consist mainly of Multi-Head Attention and Feed Forward layers. The inputs and outputs are first embedded into an n-dimensional space since we cannot use strings directly. One minor but important part of the model is the positional encoding of the different words. Since we

have no recurrent networks that can remember how sequences are fed into a model, we need to give every word in our sequence somehow a relative position since a sequence depends on its elements' order. These positions are added to the embedded representation n-dimensional vector of each word.

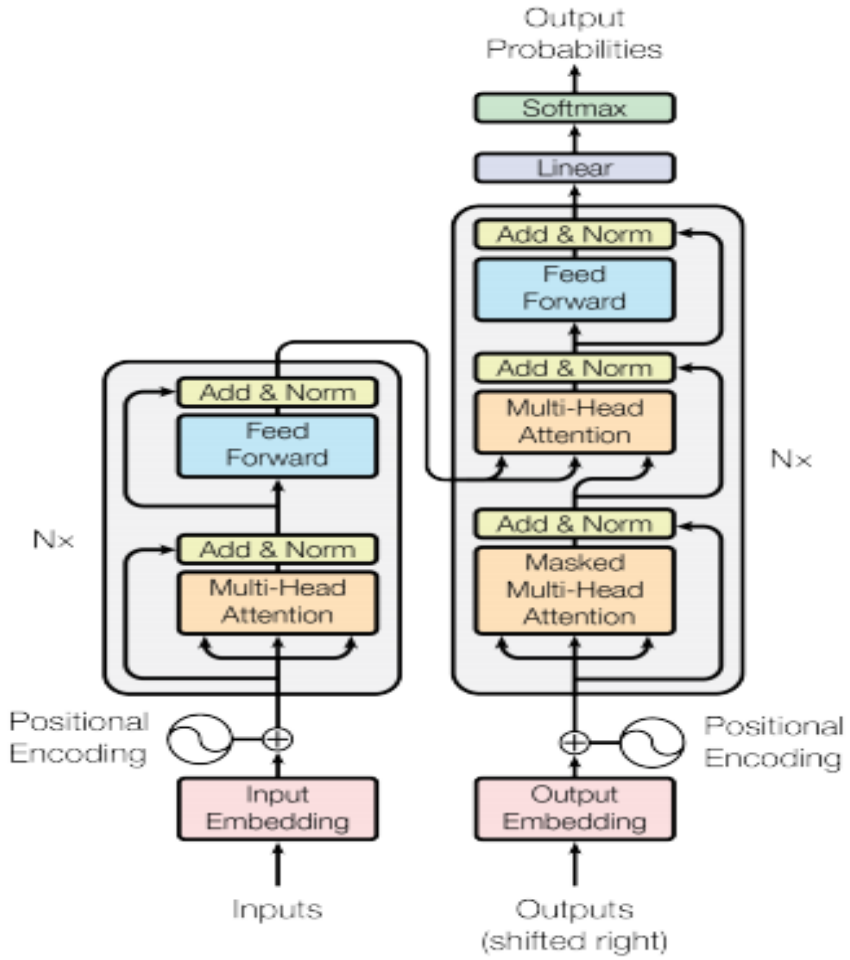


Figure 2.10: Transformer Model - model architecture [39]

## 2.7 Bidirectional Encoder Representations from Transformers model

In this section, we will discuss BERT's features, especially how it is powerful for NLP. Bidirectional Encoder Representations from Transformers (BERT) is a Transformer-based Machine Learning technique for Natural Language Processing (NLP) pre-training developed by Google. BERT is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers. BERT is the transformer architecture trained to learn language representations and conceived to be used as the main architecture to NLP tasks. [8]

BERT is conceptually simple and empirically powerful. Previous to BERT, many standard language models were unidirectional architectures where every token can only attend to the subsequent one in a single direction (right-to-left or left-to-right).

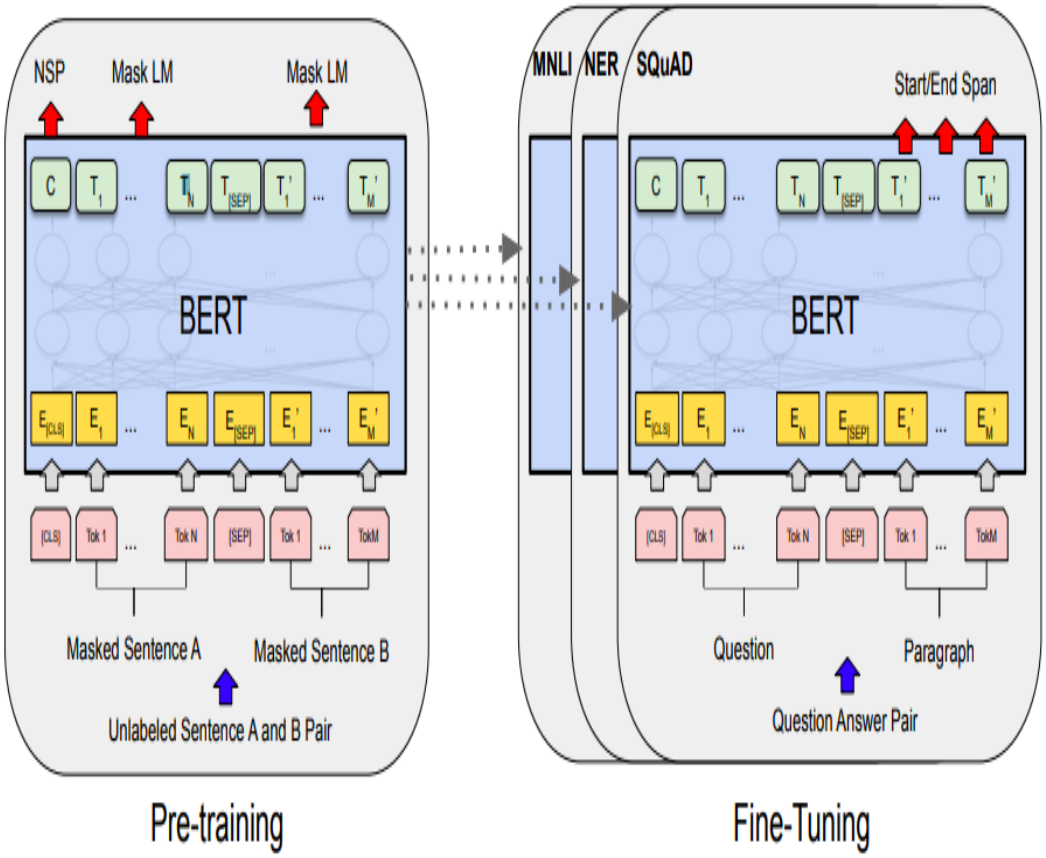


Figure 2.11: Overall pre-training and fine-tuning procedures for BERT. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token. [9]

BERT is trained using two objectives; one of them is a masked language model pre-training objective where some tokens from the input sequence are randomly masked, and the model learns to predict these words. Masking means that the model looks in both directions, and it uses the full context of the sentence, both left and right surroundings, to predict the masked word. Unlike the previous language models, it takes both the previous and next tokens into account simultaneously. Besides, the existing combined left-to-right and right-to-left LSTM based models were missing this same-time part. The second one is to understand the relation between two sentences during pre-train; the BERT training process also uses Next Sentence Prediction (NSP). During training, the model gets as input pairs of sentences, and it learns to predict if the second sentence is the following sentence in the original text as well.

These general-purpose pre-trained models can then be fine-tuned on smaller task-specific datasets. This approach results in excellent accuracy improvements compared to training on the smaller task-specific datasets from scratch. BERT model is first initialized with the pre-trained parameters, and all of

the parameters are fine-tuned using labeled data from the downstream tasks. Fine-tuning has a significant effect on performance for supervised tasks. [11]

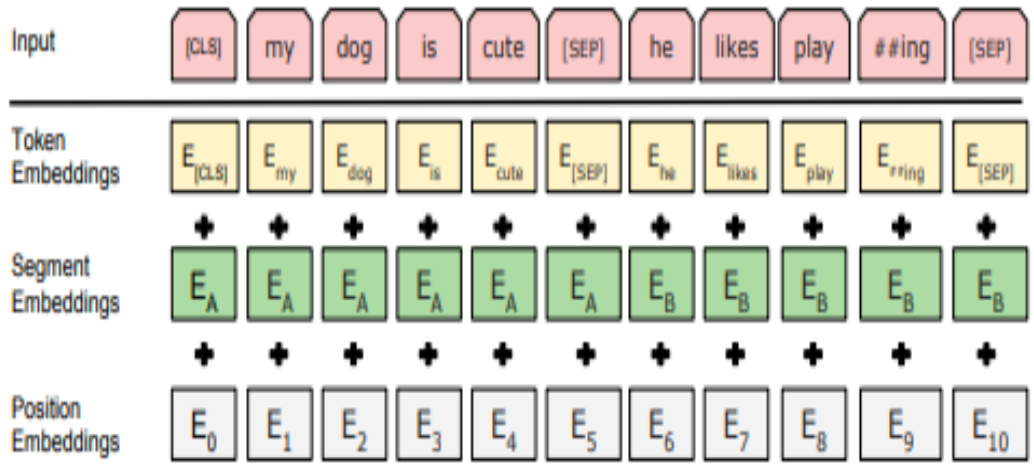


Figure 2.12: BERT input format [9]

As Figure 2.15 describes, BERT's input sequence is composed of two sentences with a [SEP] token in between and the initial classification token [CLS] that will later be used for prediction. Each token has a corresponding embedding, a segment embedding that identifies each sentence, and a position embedding to distinguish each token's position. All these embeddings are then summed up for each token. During BERT fine-tuning [58], the pre-trained transformer serves as an encoder and is added on top of it a randomly initialized classifier. BERT relies on a Transformer, the attention mechanism that learns contextual relationships between words in a text. As we mentioned in the previous section, a basic Transformer consists of an encoder to read the text input and a decoder to produce a prediction for the task. Since BERT's goal is to generate a language representation model, it only needs the encoder part. The input to the encoder for BERT is a sequence of tokens, which are first converted into vectors and then processed in the neural network.

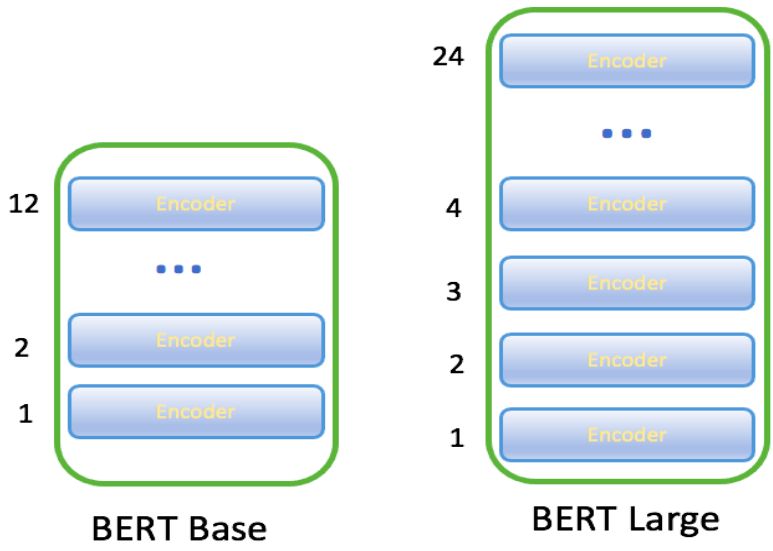


Figure 2.13: BERT model architecture

Both BERT model sizes have many encoder layers, twelve for the Base version and twenty-four for the Large version. These also have larger feedforward-networks 768 and 1024 hidden units, respectively, and more attention heads 12 and 16, respectively. We have utilized a base variant with 12 layers (transformer blocks), 768 hidden sizes, 12 attention heads, and 110 million parameters in our work. [9]

## 2.8 Attention mechanism

Attention described for the first time in this short paper Itti et al. (1998) [12]. Attention is a concept that helped improve the performance of neural machine translation applications. Attention allows the model to focus on the relevant parts of the input sequence as needed. It made it challenging for the models to deal with long sentences. An approach to solve the problem of losing relevant information in long sentences is to use the attention mechanism. A solution was proposed in Bahdanau et al.[13], 2014 and Luong et al.[14], 2015. These papers introduced and refined a technique called Attention, which significantly improved machine translation systems' quality.

Self-attention, also called intra-attention, is a process of attention that relates different positions of the individual sequence to determine the representation's same sequence. Self-attention has been used successfully in a variety of tasks, including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations. [15] [16] [17]
Mathematically, three separate vectors, namely query vector, key vector, and value vector, should be used per embedding of the word; such vectors can easily be generated by multiplying three matrices (K, V, Q) that we trained during the training process. First, a function f is used to compute the similarity between each key $K_i$ and query Q to obtain weight W. Many different kinds of functions are listed in table 2.1.

A softmax is used to normalize the weights and then finally attention weights are used to compute a

| Name | Function |
|---|---|
| Content-based Attention | $f(Q_T, K_i) = cosine[Q_T, K_i]$ |
| Additive | $f(Q_T, K_i) = V_a^T * \tanh(W[Q_T, K_i])$ |
| Location-Based | $\alpha_{l,i} = Softmax(WQ^T)$ |
| General | $f(Q_T, K_i) = Q^T W K_i$ |
| Dot-Product | $f(Q_T, K_i) = Q^T K_i$ |
| Scaled Dot-Product | $f(Q_T, K_i) = \frac{Q^T K_i}{\sqrt{n}}$ |

Table 2.1: Functions for Various Attentions

final context representation

$$a_i = Softmax(f(Q, K_i)) = \frac{exp(f(Q, K_i))}{\sum_i exp(f(Q, K_i))}$$

$$Attention(Q, K, V) = \sum_i a_i * V_i$$

The transformer model is one of the efficient encoder-decoder architectures. It presented a lot of improvements to the soft attention and made it possible to do transduction modeling without recurrent network units [10]. The Multihead Mechanism goes through the scaled dot Product attention several times in parallel, rather than only measuring the attention. Simply connected and linearly translate independent attention outputs into the necessary dimensions.

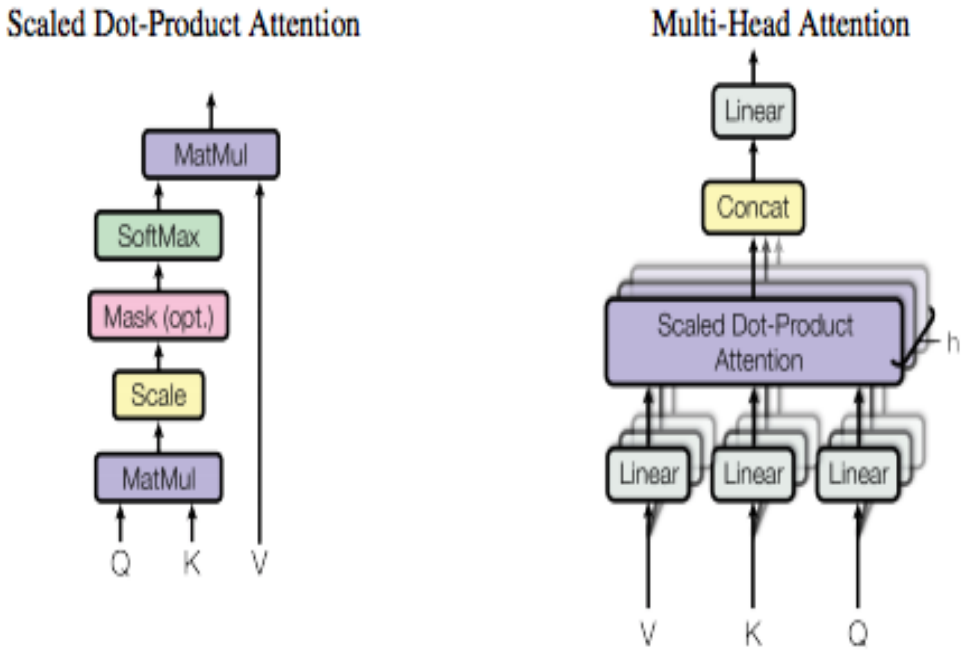


Figure 2.14: Scaled Dot-Product Attention (Left). Multi-Head Attention consists of several attention layers running in parallel (Right). [10]

The attention-head view visualizes the attention patterns produced by one or more attention heads in a given transformer layer, as shown in Figure 2.8. The representing Attention-head picture for BERT indicates that for inputs, the cat sat on the mat Sentence A and the cat lay on the rug Sentence B. The left and center figures represent different layers/attention heads. The right figure depicts the same layer/head as the central figure, but with Sentence A → Sentence B filter selected [18].

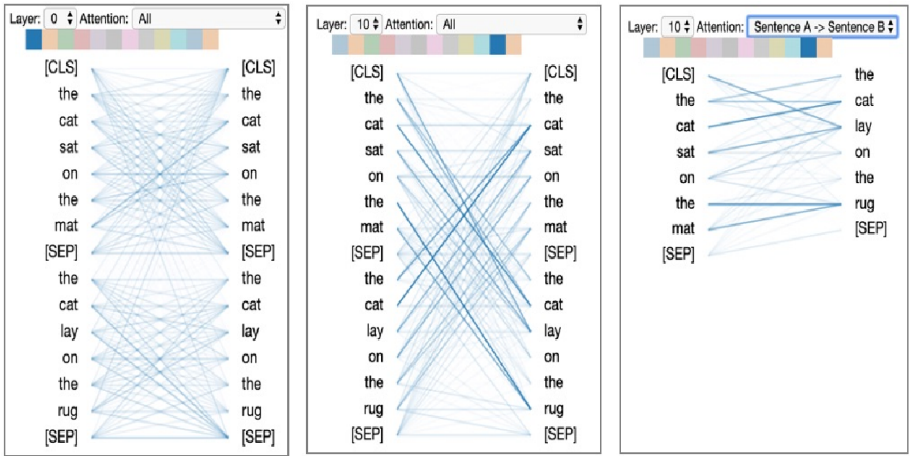


Figure 2.15: Attention-head view for BERT [18]

The model view provides a birds-eye view of attention across all of the model's layers and heads, as shown in Figure 2.9. BERT learns multiple attention mechanisms, called heads, which operate parallel to one another. From the model view, we can see that BERT produces a rich array of attention patterns.

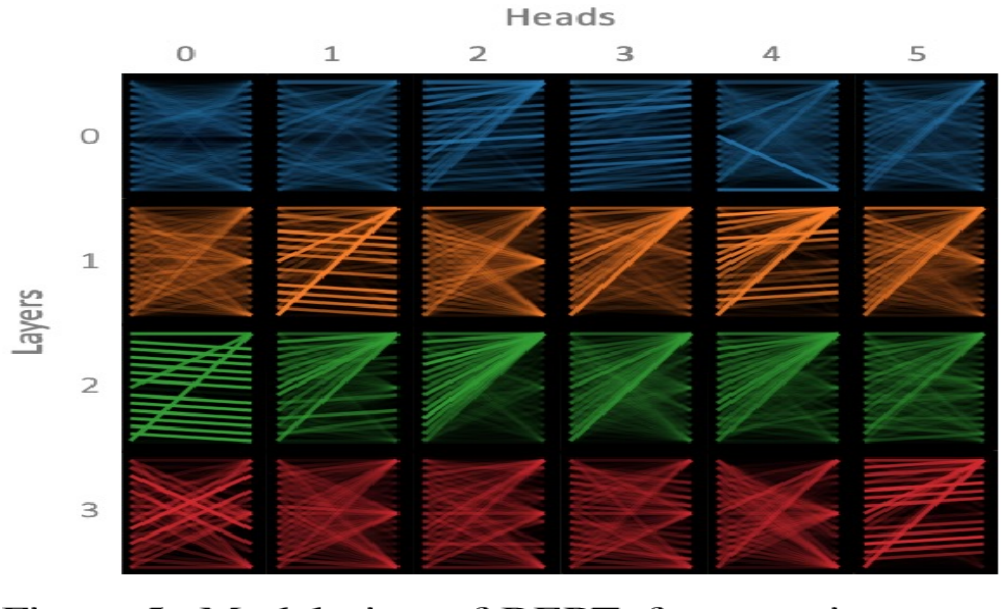


Figure 2.16: Model view of BERT, for the same inputs as in Figure 2.8 [18]

The representing neuron view of BERT, as shown in Figure 2.10, indicates for layer 0, head 0. The neuron view visualizes the individual neurons in the query and key vectors and shows how they are used

to compute attention. Positive values are colored blue and negative values orange, with color intensity representing magnitude. Like the attention-head view presented earlier, the connecting lines indicate the strength of attention between the related words. The model view and the attention-head view how what attention patterns the model learns; therefore, the neuron view indicates how the model forms these patterns.

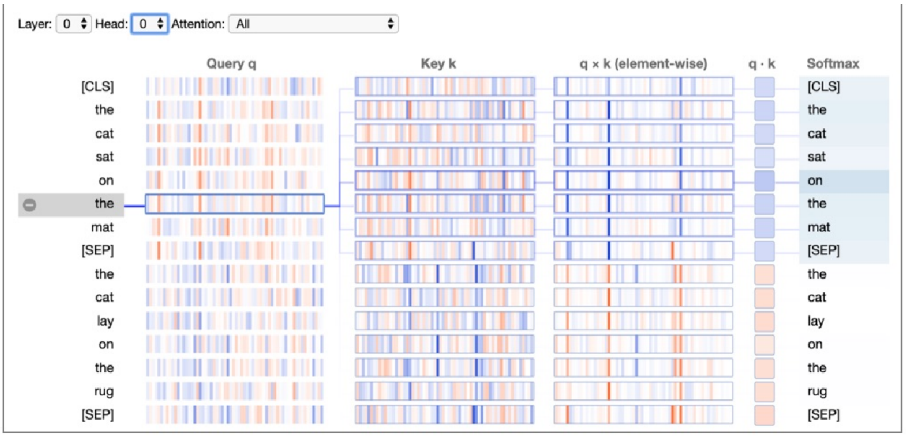


Figure 6: Neuron view of BERT for layer 0, head 0 (same one depicted in Figure 2, left). Positive and negative

Figure 2.17: Neuron view of BERT for the same one depicted in Figure 2.8 [18]

# CHAPTER 3

# Methodology

We describe a research method that started from the identification of a suitable and relevant dataset to execute a controlled experiment of the model proposed in this thesis work, the presentation of the proposed model, the selection of statistical-based solution as baseline used for comparison, and finally, a tool for visual inspection of the output results. Despite this progress, the digital system‘s interoperability continues to be an open problem that poses challenges in data integration, processing, and evaluation for a model. At the exact moment, there are increasing concerns from the scientific community that experiments are not quantifiable. Based on the previous MIMIC (MIMIC-II), published in 2010 MIMIC-III, we predict that MIMIC-III will be commonly used worldwide in scientific and industrial science, quality improvement programs, and higher education.

The Computational Physiology Laboratory is an interdisciplinary unit of data scientists and clinicians at the Massachusetts Institute of Technology. MIMIC-III is the third MIMIC critical care database version and helps us build on previous data processing and development expertise. The MIMIC-III database was filled with data collected during routine hospital care, so patients were not burdened, and their workflow was not interrupted. The information has been downloaded from various sources, including the Social Security Administration's death master file, electronic health record databases, critical care information system documents. [55]

## 3.1 Information Flow

For the prediction of a discharge summary from clinical notes, we are using the table of NOTEEVENTS from MIMIC-III database, which are free text notes providing progress notes and hospital discharge summaries. Many previous studies like Baumel et al. (2017) and Mullenbach et al. (2018) used discharge summaries or discharge procedures[56][59]. Only a few others, like Kavuluru et al. (2015), the used patient stays from other databases of UKY [62]. The data is obtained in a large tertiary care hospital. Specifically, data of patients admitted to critical care units. The MIMIC-III database is widely used for academic and industrial research purposes.

In this work, we are using diagnosis and procedure codes to combine for localizing patient summaries. We are using a complete dataset with 9017 codes. Data processing of clinical notes with modern NLP features gives better results than all previous Mullenbach et al. (2018) [59] and Shi et al. (2017) [57]. We initialized embedding from pre-trained distributed models. Training contains the choice to either evaluated embedding using Word2Vec or GloVE 42B with 300 dimensions. There is an additional option to create stack embedding because many relevant embeddings cannot any of these models. So, we construct random embeddings for out-of-vocabulary terms. Stack embeddings are supposed to increase coverage; external embeddings from gloves are Common English corpus instead of any specialized medical corpus. Protecting the health information and input tokens; these embeddings are kept trainable to divert according to the relevant context [57] [59].

We are experimenting with the model used by Stefano et al. (2019) [60] with a variant of computing embeddings with word level. The model computes by changing filter size from lower to a higher level to compute attention in a hierarchal way similar to Yang et al. (2016) [64]. The model formed by Mullenbech et al. (2018) was tuned for the top 50 codes, and parameters were optimized for the top 8 occurring codes, whereas we propose a model for total codes in MIMIC-III. Attention is scored based on embedding descriptions at the word level.

## 3.2 Pre-processing

In textual data science tasks, this means that any raw text needs to be carefully preprocessed before the algorithm can digest it. In the most general terms, we take some predetermined body of text and perform upon it some fundamental analysis and transformations, to be left with artifacts that will be much more useful for a more meaningful analytic task afterward. The creation of the MIMIC dataset involved balancing interpretation simplicity against proximity to ground-based facts. As such, information represents the underlying data sources, updated in response to user feedback through the MIMIC database iterations. The underlying text's problem is its unusual nature: medical drug names, vital signs, and abbreviations. Applying simple lemmatization and stemming tools may lose the semantic meaning of diagnosis notes. [63]

Additionally, these notes are de-identified based on regulations of (H.I.P.P.A) to protect health information. Information like lastname, first name, location, dates is removed and is replaced with unknown generic tokens. For Instance [** Known Lastname 025046**], the purpose of putting this number is to keep human identification hidden from the researcher. MIMIC presents subject_ID to interpret the related history to every person. Various text-preprocessing approaches were tried on MIMIC; Baumel et al. (2017) replaced non-alphabetical characters with pseudo tokens [56].

Shi et al. (2017) [57] transformed the messy and inconsistent raw note texts in these parts into tidy diagnosis details. He used several regular text pre-processing techniques, such as standard expression matching and tokenization. That resultant mark is a short sentence or a term that articulates a disorder or illness. To evaluate the model commonly, he visited patients that do not include details of the condition are rejected, and worked on top 50 codes.

The strategy of removing tokens that do not contain alphabetical characters (e.g., remove "500" but keep "250 mg"), lower all tokens, and replace tokens that appear with a ' UNK ' token in less than three training documents[59]. He cropped documents at a maximum of 2500 tokens in each summary.

The text processing data consumed by this work is similar to how we are using SpaCy to create custom tokenization with the rule-based approach [60]. The prime reason behind this approach is the nature of MIMIC notes described in section 4.1. To treat the most common medical abbreviations accurately, as the tokenizer sometimes breaks them incorrectly. Therefore we delete tokens with no alphabetic characters: this law excludes numbers and bogus tokens made entirely of symbols, but it preserves words containing numbers like (from 60 Capsules, removing 60).

This mapping retains the text structure, is easily interpretable when interpreting the written annotations, and allows such words to be accurately tokenized. We have checked the numerical attributes associated with the anonymized tokens: just as [**Unknown 5252**] is set to be preserved as "unknown_5252". We found that this strategy resulted in a significant increase in the number of distinct tokens in the body, leading to yet more complex outcomes. Also, dates are ignored; just the name of months is taken as a string from discharge, and admission dates; for example, 2008/8/24 is left as "august."

Similar to Tal Baumel et al. (2017) [56] approach of dealing with out-of-vocabulary words, We took all those tokens that repeated at least three times in the training corpus. Unique UNK token is introduced for OOV words. The approach of shortest Levenshtein distance was implemented to map unknown words to known words, but it does not show any significant development in evaluation metrics [61].

Here is an example that illustrates some of these tokenization excerpts: anonymized dates and names, organized alphabetical lists, information on drug delivery, vital signs, etc. This demonstrates a sample of clinical notes before or after our preprocessing and tokenization. The note processed with a one-sentence-per-line format is generally more straightforward, and we can observe how most problem expressions are treated correctly. The main problem is detecting the sentence boundaries, with many phrases incorrectly separated into two or more phrases.

## 3.3 Model Overview

The model is based on three components, embedding layer, BERT layers, and attention layer. Data is fed to embedding layers which then initializes matrices for text present and forwards to the BERT layers. The embedding layer provides a dense representation of words and their relative meanings. These embeddings can be computed in various ways with a selective option to choose Word2Vec, Fast text English typical crawl of GloVe with forty-two billion tokens. Although this corpus is not medical specific, it yields a good coverage for text present in discharge summaries. BERT layers create an overview of sentence-level understanding to generate attention at sentence vectors.

BERT is multi-layer neural architecture implemented over the transformer. It has two major variants, and we have utilized a base variant with 12 layers (transformer blocks), 768 hidden sizes, 12 attention heads, and 110 million parameters.

Attention has been described previously in section 2.8, and It is crucial to infer the critical information from the text. For words-level attention representation, word representation hit goes through a fully connected layer and a non-linear activation function. The attention distribution for tokens on the last layer has an irregular pattern. Attention focuses on the last layers (12 layers) according to BERT layers' learning strategy, starting the first layer up to twelve layers. The essential, meaningful words are in the last layer.

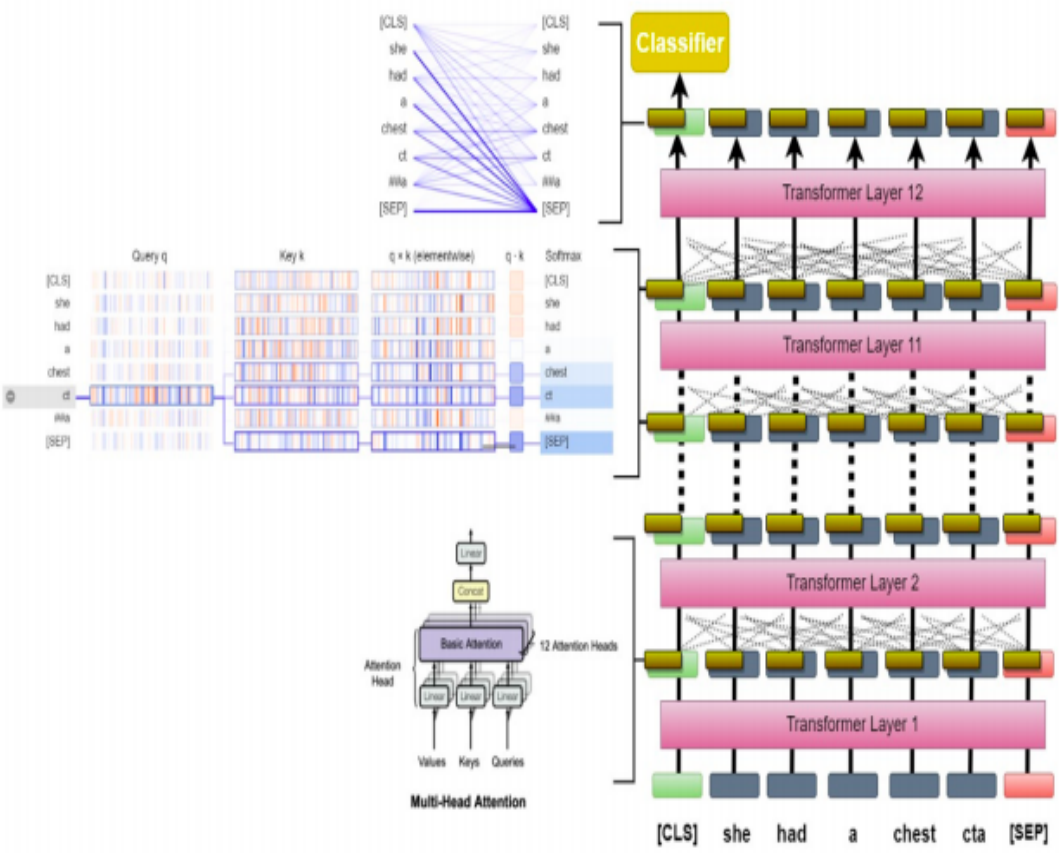


Figure 3.1: BERT base architecture with 12 transformer layers. Every layer conducts embeddings and attention scores for corresponding tokens. These multi-head attentions correlate every to every other word, as seen in the images at left.

Admission Date: [**2118-6-2**] Discharge Date: [**2118-6-14**]

HISTORY OF PRESENT ILLNESS: This is an 81-year-old female with a history of emphysema (not on home O2), who presents with three days of shortness of breath thought by her primary care doctor to be a COPD flare. Two days prior to admission, she was started on a prednisone taper and one day prior to admission she required oxygen at home in order to maintain oxygen saturation greater than 90%. She has also been on levofloxacin and nebulizers, and was not getting better, and presented to the [**Hospital1 18**] Emergency Room. In the [**Hospital3 **] Emergency Room, her oxygen saturation was 100% on CPAP. She was not able to be weaned off of this despite nebulizer treatment and Solu-Medrol 125 mg IV x2.

Review of systems is negative for the following:

Fevers, chills, nausea, vomiting, night sweats, change in weight, gastrointestinal complaints, neurologic changes, rashes, palpitations, orthopnea. Is positive for the following: Chest pressure occasionally with shortness of breath with exertion, some shortness of breath that is positionally related, but is improved with nebulizer treatment.

PAST MEDICAL HISTORY:

COPD. Last pulmonary function tests in [**2117-11-3**] demonstrated a FVC of 52% of predicted, a FEV1 of 54% of predicted, a MMF of 23% of predicted, and a FEV1:FVC ratio of 67% of predicted, that does not improve with bronchodilator treatment. The FVC, however, does significantly improve with bronchodilator treatment consistent with her known reversible air flow obstruction in addition to an underlying restrictive ventilatory defect. The patient has never been on home oxygen prior to this recent episode. She has never been on steroid taper or been intubated in the past.

DISCHARGE MEDICATIONS:

1. Levothyroxine 75 mcg p.o. q.d.

2. Citalopram 10 mg p.o. q.d.

Figure 3.2: A sample from MIMIC Discharge Summary

admission date june discharge date june

history of present illness this is an year old female with a history of emphysema not on home o2 who presents with three days of shortness of breath thought by her primary care doctor to be a copd flare two days prior to admission she was started on a prednisone taper and one day prior to admission she required oxygen at home in order to maintain oxygen saturation greater than she has also been on levofloxacin and nebulizers and was not getting better and presented to the hospitaltoken emergency room in the hospitaltoken emergency room her oxygen saturation was on cpap she was not able to be weaned off of this despite nebulizer treatment and solu medrol mg iv x2

review of systems is negative for the following fevers chills nausea vomiting night sweats change in weight gastrointestinal complaints neurologic changes rashes palpitations orthopnea is positive for the following chest pressure occasionally with shortness of breath with exertion some shortness of breath that is positionally related but is improved with nebulizer treatment

past medical history copd last pulmonary function tests in november demonstrated a fvc of of predicted a fev1 of of predicted a mmf of of predicted and a fev1 fvc ratio of of predicted that does not improve with bronchodilator treatment the fvc however does significantly improve with bronchodilator treatment consistent with her known reversible air flow obstruction in addition to an underlying restrictive ventilatory defect the patient has never been on home oxygen prior to this recent episode she has never been on steroid taper or been intubated in the past

discharge medications

levothyroxine mcg p.o. q.d.

citalopram mg p.o. q.d.

aspirin mg p.o. q.d.

Figure 3.3: Sample of Preprocessed discharge summary

# CHAPTER 4

# Dataset

In this section, we have presented the dataset that we have used. We used discharge summaries clinical notes in the NOTEEVENTS table from MIMIC-III. Each discharge summary is identified with a unique label for the linked disease. In total, we have 47,724 clinical notes. We have used 100 discharge summaries for training our models, and it is allowed to perform a qualitative evaluation as described in Sections 5 & 6.

## 4.1 The MIMIC-III Dataset

MIMIC-III (Medical Information Mart for Intensive Care III) is an extensive and freely-available clinical database. It contains information on patients confined to critical care facilities in a major tertiary hospital. The data includes vital information, medication, laboratory tests, care provider findings and notes, fluid balance, procedure codes, diagnostical codes, photo reports, stay-length hospital data, and more. The database serves projects such as scientific and market research, programs to improve quality, and higher education. [55]

The first release is MIMIC- II (Multiparameter Intelligent Monitoring Intensive Care), which contained patients' data at Beth Israel Deaconess Medical Center between 2001 and 2008. Since the MIMIC was one of the few first available databases, many research publications are concluded based on MIMIC-II. To relate it with the current data website `https://mimic.physionet.org/mimicdata/whatsnew/` describes the relationship between tables to understand how it was upgraded. MIMIC-III is an extension of old MIMIC-II, which was later incorporated with further data from 2008–12. [55]

In this work, we have used the version is MIMIC-III (v1.4), released in 2016. MIMIC-III includes data associated with 53,432 adults and 8100 new-born children admitted to critical care units between 2001 and 2012. Also, the median age of adult patients is 65.8 years. The median length of an ICU stay is 2.1 days, and the median length of a hospital stay is 6.9 days. A mean of 4579 charted observations ('chart events') and 380 laboratory measurements ('labevents') are available for each hospital admission. Table 4.1 provides a breakdown of the adult population by the care unit.

MIMIC-III is presented as a compilation of comma-separated value (CSV ) files. A free demo version32 4.1. THE MIMIC-III DATASET of 100 records can be downloaded directly from Physionet, but access to all 26 tables, a particular MIT-based course is required. The course “Data or Specimens Only Research” course ensures defining data regulation laws for research purposes. Here is a table presenting details of the data. The tables are associated with identifiers that usually have the suffix ‘ID.’ For example, SUBJECT_ID describes a unique patient, HADM_ID points to an extraordinary admission to the hospital, and ICUSTAY_ID refers to single access to an intensive care unit.

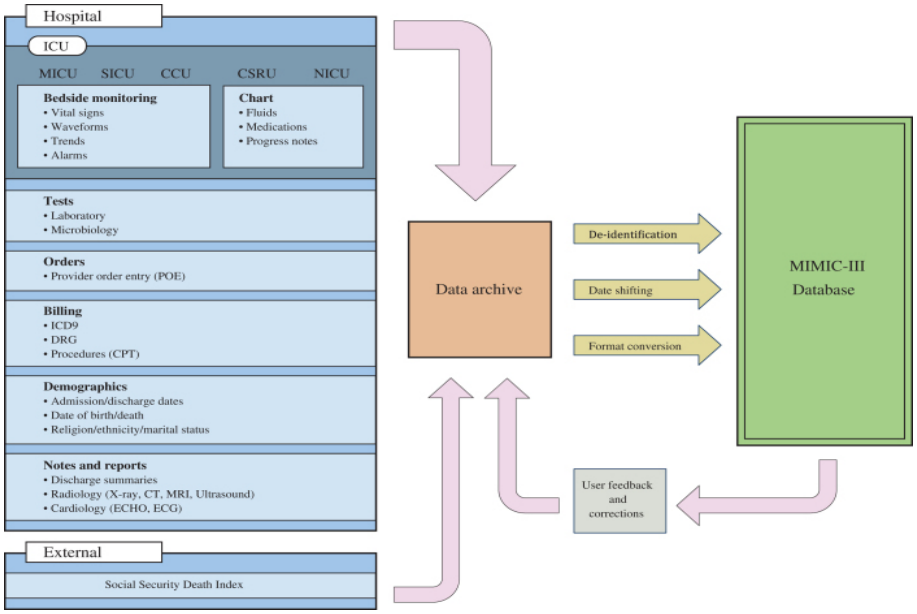


Figure 4.1: Overview of MIMIC-III critical care database , taken from [66]

| **Table Name** | **Description** |
|---|---|
| ADMISSIONS | Every unique hospitalization for each patient in the database (defines HADM_ID). |
| CALLOUT | Information regarding when a patient was cleared for ICU discharge and when the patient was actually discharged. |
| CAREGIVERS | Every caregiver who has recorded data in the database (defines CGID). |
| CHARTEVENTS | All charted observations for patients. |
| CPTEVENTS | Procedures recorded as Current Procedural Terminology (CPT) codes. |
| D_CPT | High-level dictionary of Current Procedural Terminology (CPT) codes. |
| D_ICD_DIAGNOSES | Dictionary of International Statistical Classification of Diseases and Related Health Problems (ICD-9) codes relating to diagnoses. |
| D_ICD_PROCEDURES | Dictionary of International Statistical Classification of Diseases and Related Health Problems (ICD-9) codes relating to procedures. |
| D_ITEMS | Dictionary of local codes ('ITEMIDs') appearing in the MIMIC database, except those that relate to laboratory tests. |
| D_LABITEMS | Dictionary of local codes ('ITEMIDs') appearing in the MIMIC database that relates to laboratory tests. |
| DATETIMEEVENTS | All recorded observations are dates, for example, time of dialysis or insertion of lines. |
| DIAGNOSES_ICD | Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system. |
| DRGCODES | Diagnosis Related Groups (DRG), which are used by the hospital for billing purposes. |
| ICUSTAYS | Every unique ICU stays in the database (defines ICUSTAY_ID). |
| INPUTEVENTS_CV | Intake for patients monitored using the Philips CareVue system while in the ICU, e.g., intravenous medications, enteral feeding, etc. |
| INPUTEVENTS_MV | Intake for patients monitored using the iMDSoft MetaVision system while in the ICU, e.g., intravenous medications, enteral feeding, etc. |
| OUTPUTEVENTS | Output information for patients while in the ICU. |
| LABEVENTS | Laboratory measurements for patients both within the hospital and in outpatient clinics. |

| | |
|---|---|
| MICROBIOLOGYEVENTS | Microbiology culture results and antibiotic sensitivities from the hospital database. |
| NOTEEVENTS | Deidentified notes, including nursing and physician notes, ECG reports, radiology reports, and discharge summaries. |
| PATIENTS | Every unique patient in the database (defines SUBJECT_ID). |
| PRESCRIPTIONS | Medications ordered for a given patient. |
| PROCEDUREEVENTS_MV | Patient procedures for the subset of patients who were monitored in the ICU using the iMDSoft MetaVision system. |
| PROCEDURES_ICD | Patient procedures, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system. |
| SERVICES | The clinical service under which a patient is registered. |
| TRANSFERS | Patient movement from bed to bed within the hospital, including ICU admission and discharge. |

Table 4.1: Tables from MIMIC-III Dataset

We hold only the summaries of discharge and their addenda providing the most detailed diagnostic information. The timestamp of the documents gives us the correct reading order for patient admission. Some hospital stays do not have discharge summaries: following previous studies, we only consider those that do (Perotte et al., 2013; Baumel et al., 2017; Mullenbach et al., 2018). The data set comprises 8,929 unique ICD codes for patients with discharge summaries (6,918 diagnoses & 2,011 procedures). We ignore five of them, however, and use 8,924 codes. The total number of header codes is 1168.

To get access to the dataset, the instructions provided by MIMIC are here (https://mimic.physionet.org/gettingstarted/access/). You should follow the following components. The four main components are

- Complete CITI training course
- Create a PhysioNet account
- Request access to MIMIC III
- Accessing MIMIC III

Since you will be working with patient data, you are required to complete the training course “Data or Specimens Only Research” provided by CITI. After completing the CITI training course, you should create a PhsioNet account. After activating your PhysioNet account, you need to request access to MIMIC III. After receiving access to the MIMIC III database, you can use the dataset for your research.

## 4.2 Experimental Setup

In this section, we discuss the tools we used for the implementation of our work.

### 4.2.1 Framework and libraries

We chose Python programming language for our experiments. Python is an interpreted, high-level and general-purpose programming language. It made our work much easier by providing its excellent libraries such as Pandas, PyTorch, NLTK, Spacy, Tensorflow, etc.

**Pandas**

We used Pandas to manipulate our data processing. Since our dataset was extensive and Pandas made our work much more manageable. Pandas is a software library written for the Python programming language for data manipulation and analysis in computer programming. In particular, it offers data structures and operations for manipulating numerical tables and time series Pandas is very similar; you feed it some data that fits into rows and columns, you make some operations, and you get a result; you do all of this through commands that manipulate data frames and series instead of spreadsheets, how you represent the information at the end is up to you.

**PyTorch**

PyTorch [53] is one of the most used machine learning libraries for applications such as NLP and computer vision. It is an open-source library based on Torch library. At the beginning developed by facebook's AI Research lab. It is free and open-source software released under the Modified BSD license.

**NLTK**

The Natural Language Toolkit (NLTK) is a suite of libraries and programs for symbolic and statistical NLP for English written in the Python programming language. The NLTK module is a massive toolkit designed to help you with the entire NLP approach. NLTK will provide you with everything from splitting paragraphs to sentences, splitting words, identifying the part of speech, highlighting themes, and even helping your machine understand what the text is about.

**Spacy**

SpaCy is a free, open-source library for NLP in Python. It's written in Cython and distributed under MIT license. It is developed for performing simple to advanced Natural Language Processing tasks such as tokenization, part-of-speech tagging, named entity recognition, text classification, calculating semantic similarities between text, lemmatization, and dependency parsing, among others. Unlike libraries like NLTK, another trendy NLP library in Python, SpaCy focuses on providing NLP software for use in production environments. A compelling feature of SpaCy, which makes it different from all other NLP tools, is its support for deep learning workflows, which allows models trained using other machine learning libraries from the Python eco-system such as Tensorflow, Keras, Scikit-learn, and PyTorch, to be readily integrated with it.

**Huggingface's Transformers**

The Huggingface transformers [54] package is a trendy python library providing pre-trained models that are extraordinarily useful for various NLP tasks. It previously supported only PyTorch, but as of late 2019, TensorFlow 2 is supported as well. Hugging-face Transformers is used many well-known transformer architectures, such as BERT, RoBERTa, GPT-2, or DistilBERT, that obtain state-of-the-art results on a variety of NLP tasks like text classification, information extraction, question answering, and text generation.

### 4.2.2 Google Colaboratory

Collaboratory or Colab, for the short name, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education. Colab is based on the open-source project Jupyter; the code is executed on a remote virtual machine connected with your account. Colab notebooks are stored in Google Drive, or you can load from GitHub. Google Colab provides NVIDIA Tesla K80 GPU that can be used for up to 12 hours continuously. Also, Colab is offering free TPU. Using the resources provided by this tool, we could carry out all the experiments without any problems and with fast execution times.

In addition, we have utilized the Jupyter notebook. The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. You can use Anocando to set Jupyter notebook.

### 4.2.3 Pycharm

PyCharm is an integrated development environment (IDE) specially designed for Python development, and we worked with Python. One of the most extraordinary things from PyCharm is that it is possible to use the virtual env tool to create a project-specific isolated virtual environment. The main idea of virtual environments is to manage a particular project's settings and dependencies regardless of other Python projects.

### 4.2.4 Github

GitHub is a service that allows you to host your Git repositories online and collaborate with others on them. You can use GitHub through their web portal and the GitHub desktop GUI and the Git Shell. We worked in a private Github repository that only belongs to the company. I have pushed most of the work to my individual public Github is shown in chapter 6.

### 4.2.5 Doccano (Docker)

Doccano [2] is an open-source annotation tool for machine learning practitioners. It provides annotation features for text classification, sequence labeling, and sequence to sequence tasks. To run Doccano, we need Docker. Docker is an open-source project and platform as service products that use OS-level virtualization to deliver software packages called containers. Docker Container is a standardized unit created on the fly to deploy a particular application or environment. To export and import the to Doccano the data, we have utilized JSON, and JSONL data file format, making it easier for human-readable text to store and transmit data objects consisting of attribute-value pairs array types.

# CHAPTER 5

# Evaluation Metrics

In this section, we describe evaluation metrics for summary content, in which case we do not have gold-standard as a reference summary. Evaluating a summary [38] is an essential task for text summarization. Any project requires to be evaluated at the end of work since we do not have a gold standard to compare either reasonable summary or not the trustworthy summary. For our case, we have taken the original document as a reference summary and used four different evaluation metrics which do not require gold-standard human summaries for the evaluation. Good summaries would expect that the more similar a summary is to the original document, the better it's content.

For the task of understanding the summary, it is crucial to understand which words are vital in the original document. We used Cosine similarity metrics as one of the distribution similarities. Cosine Similarity helps overcome the count of the common words in the other word Cosine similarity demonstrates vector similarity, and it also avoids word count frequency. [7]

$$cos\theta = \frac{\overrightarrow{A.B}}{||A||.||B||}[4] \tag{5.1}$$

Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The resulting similarity ranges from $-1$, meaning exactly opposite directions; there is no similarity found, to 1 representing the same direction; there are similarities between the summary. With 0 indicating orthogonality, and there is some similarity. In conclusion, a smaller angle means higher similarity.

The next metric we used is Jaccard similarity. Jaccard similarity measures the similarity between two sets. Thus, a higher percentage demonstrates the higher similarity between the two documents.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}[6] \tag{5.2}$$

Size of the intersection of set A and set B over the size of the union of set A and set B. Here we take the number of unique elements as the intersection of set A and set B. For denominator has taken the number of common elements as the union of set A and set B. By showing a short example, we can demystify the main idea of Jaccard similarity. For instance, we have two sentences, such as below.

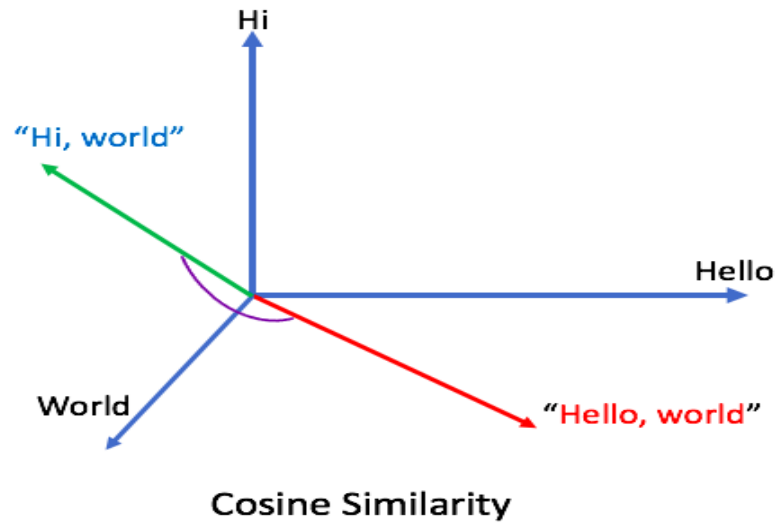


Figure 5.1: Cosine Similarity

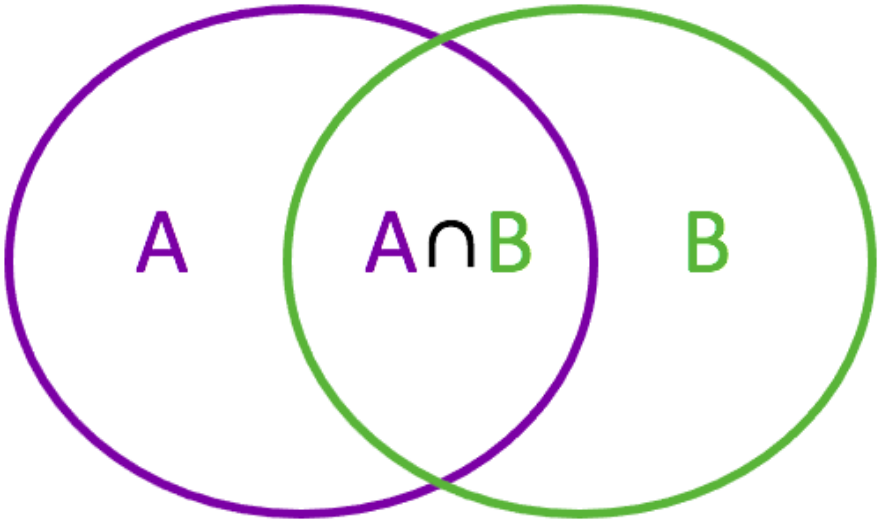


Figure 5.2: Jaccard Similarity

"I will go back"=[I, will, go, back]
"I will go there soon"=[I, will, go, there, soon]
**Jaccard Similarity = 3/6 = 0.5**

By applying Jaccard similarity, we could find similarities between those sentences.

We experimented for evaluation two other metrics: Kullback Leibler (KL) divergence, and Jensen Shannon (JS) divergence. Good summary to be characterized by the low divergence between probability distributions of words in the original document and summary, and higher similarity with the original

form.

The KL divergence formula between two probability distributions, P and Q, is given below.

$$D(P||Q) = \sum_{x \in X} P(x) log \frac{P(x)}{Q(x)} \tag{5.3}$$

KL divergence is not symmetric; both input-summary and summary-input divergences will give us different results; however, both results are helpful. In our case, the two distributions of word probabilities are measured from the input and summary, respectively.

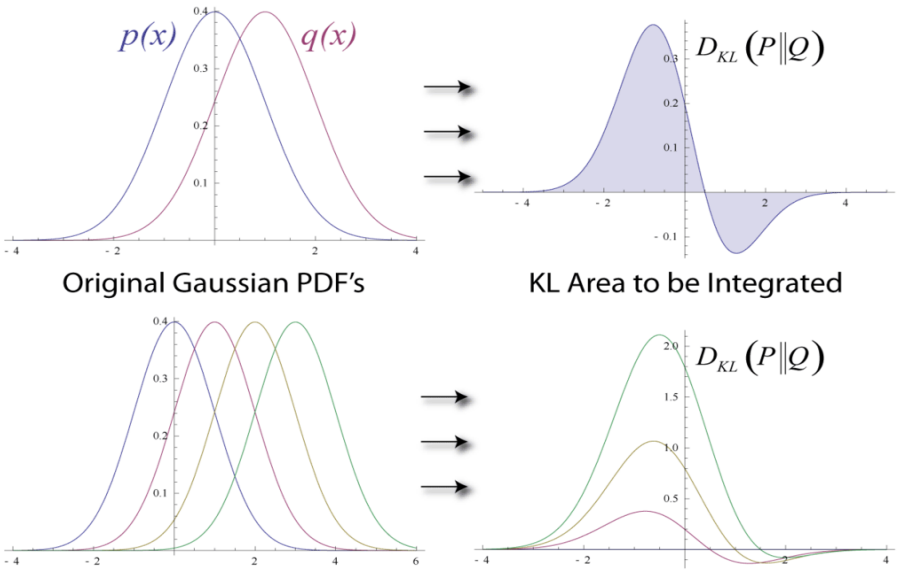


Figure 5.3: Illustration of the relative entropy for two normal distributions [68]

The JS divergence formula is shown below.

$$J(P||Q) = 0.5KLD(P||M) + 0.5KLD(Q||M)M = 0.5(P+Q) \tag{5.4}$$

$$M = 0.5(P+Q) \tag{5.5}$$

The JS divergence is symmetric, and this makes the difference between KL and JS and in computation makes it easier than KL. The idea of JS is that the distance between two distributions cannot be very different from the average of distances from their mean distribution.

## 5.1 Web-based Text Annotation Tool

We selected to use doccano [2] as an annotation tool. Doccano is a modern-looking open source and user-friendly web-based annotation tool. We know it might be hard to make gold-standard data by domain experts. So to reduce the struggle for domain experts, doccano is a valuable tool. They are written mainly in Python. NodeJs and Vue it deployed as a docker container running on the remote server.

Doccano allows you to import and export data easily.

We uploaded our patients' notes to doccano as a JSONL (JavaScript Object Notation Line) file format. And we chose to work with one label as it is named summary. Once you finish annotation, you can export your annotated data in either JSON (JavaScript Object Notation) or JSONL (JavaScript Object Notation Line) file format. The difference between JSON and JSONL is the JSON format is structured as an array of elements; nonetheless, the JSONL contains a component for each line.

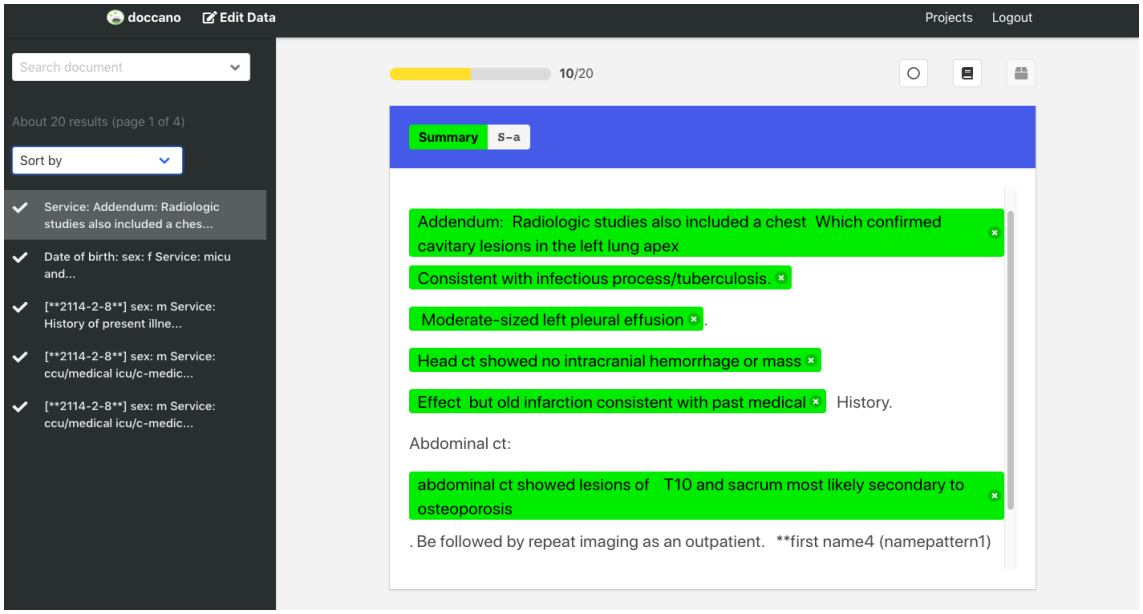


Figure 5.4: Web-based annotation tool doccano

# CHAPTER 6

# Results

This chapter presents the results by performing evaluation metrics that we have discussed chapter about it previously. We have achieved a comparison of our method with the baseline for the extractive summarization method. First, Parts-of-Speech-based sentence tagging is established on the empirical frequency selection method (Edmundson, 1969).

The Transformed-based approach demonstrates significant improvement compared with this baseline. Transformed-based results show that the summary is more informative than a statistical-based approach based on all Cosine similarity, Jaccard Similarity, Kullback–Leibler (KL) Divergence Jensen-Shannon (JS) Divergence scores. Frequency-based approach outcomes highest divergence it means more divergence makes less similarity. Overall, the attention-based mechanism poses great abstraction power for the summarization task.

We developed CUI for both our solutions as well. Users can see results with experimented metrics by choosing either a Transformed-based solution or a Statistical-based solution. Codes for these results and work we have done can be found at `https://github.com/rizvansaatov/Attention_based_Summarization_Approach_of_Clinical_Notes` [5].

We worked in a private GitHub repository provided by LINKS Foundation. For that reason, I did not publish not allowed sources such as dataset, etc. However, in my GitHub repository, you can find most of the work in the provided link above. We chose to work with the Colab notebook since Colab supports GPU, and as an IDE, we used PyCharm. All required guidelines are provided in the readme section.

The results are shown in table 6.1 with chosen baseline model. The table demonstrates the different results after applied various metrics. It shows in table 6.1 experimental results on 100 patients' clinical notes from MIMIC-III. We have calculated the mean value on 100 patients' clinical notes and compare them with the baseline. As the results are shown, the BERT model is better than the chosen baseline. KL Divergence and JS Divergence where values closer to 0 mean less distance means higher similarity. Cosine Similarity and Jaccard Similarity, where a higher value indicates higher similarity.

Figure 6.1 & 6.2 shows the results' performance on 100 patients' clinical notes values with Cosine Similarity, Jaccard Similarity, KLD, and JSD, respectively, as discussed in table 6.1.

Figure 6.3 & 6.4 shows the experimental results on 100 patients' clinical notes for each patient. Figure 6.5 & 6.6 shows the experimental results on 100 patients' clinical notes with Cosine Similarity and Jaccard Similarity, respectively. Figure 6.7 & 6.8 shows the experimental results on 100 patients' clinical notes with KLD and JSD, respectively.

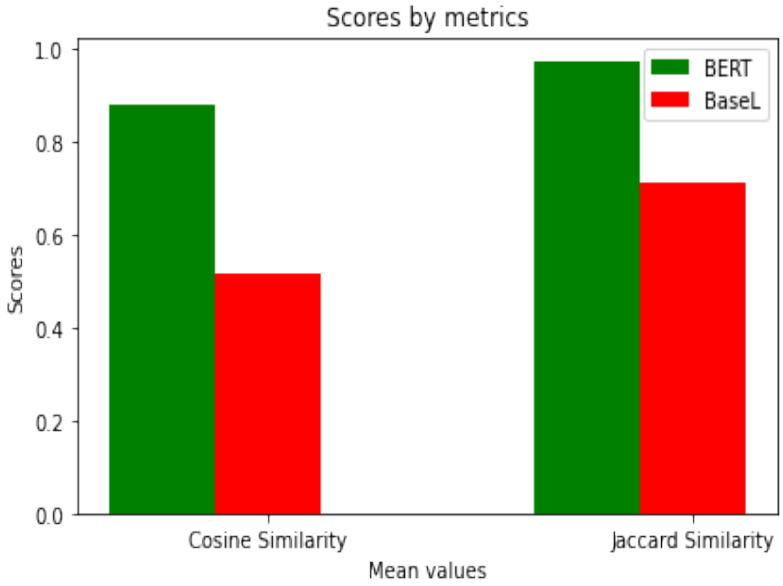


Figure 6.1: Experimental results on 100 patients clinical notes with Cosine Similarity and Jaccard values


Figure 6.2: Experimental results on 100 patients clinical notes with KLD and JSD values

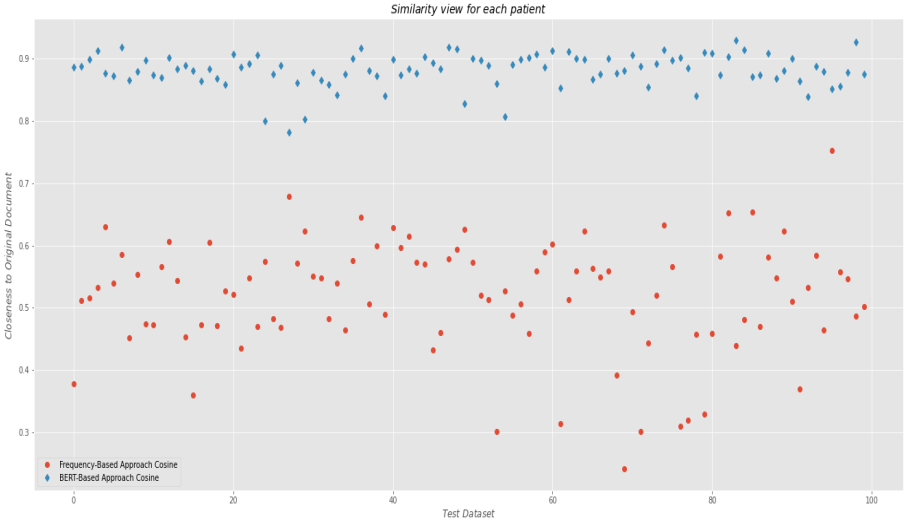


Figure 6.3: Experimental results on 100 patients clinical notes with Cosine Similarity for each patient

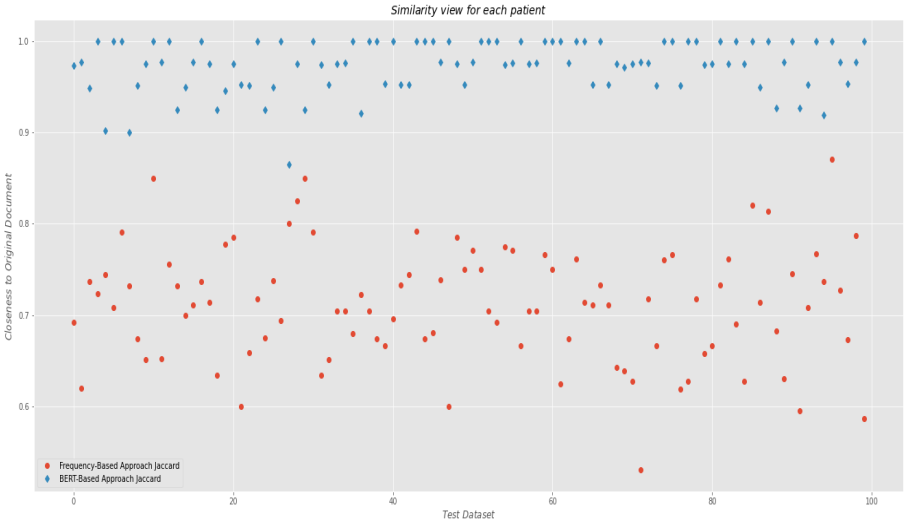


Figure 6.4: Experimental results on 100 patients clinical notes with Jaccard Similarity for each patient

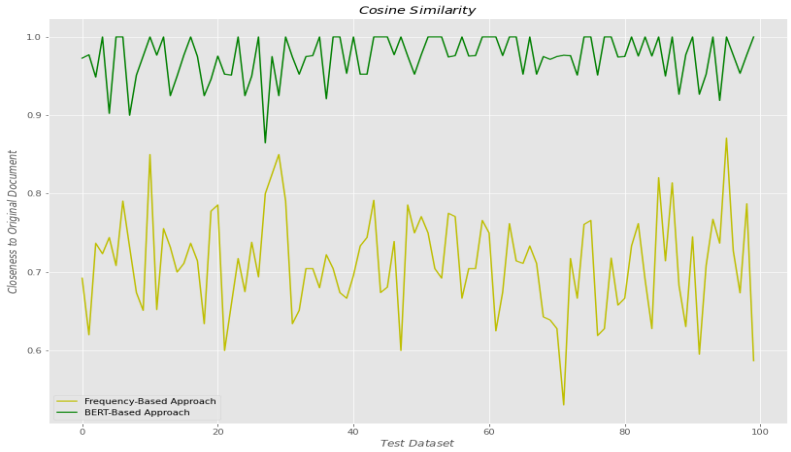


Figure 6.5: Experimental results on 100 patients clinical notes with Cosine Similarity

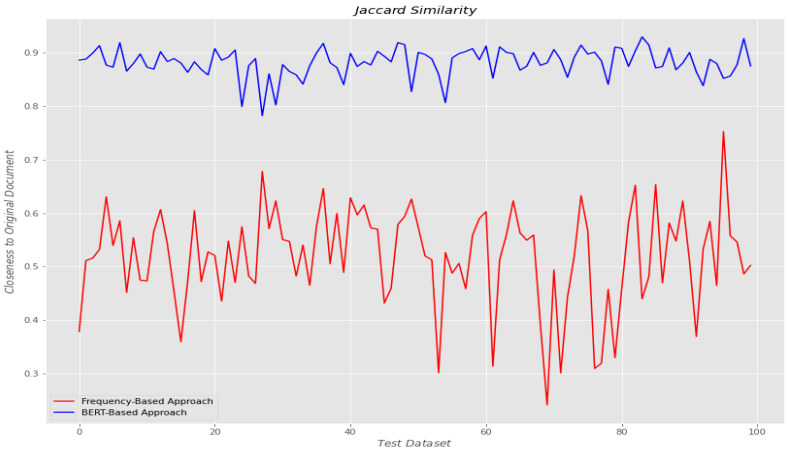


Figure 6.6: Experimental results on 100 patients clinical notes with Jaccard Similarity

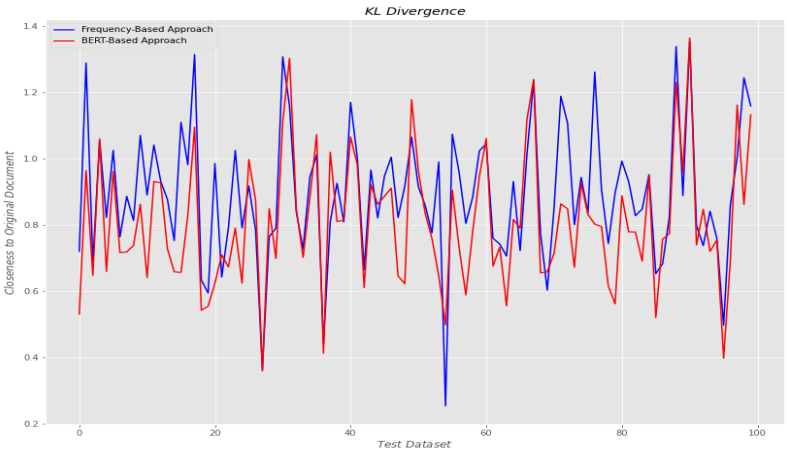


Figure 6.7: Experimental results on 100 patients clinical notes with KLD

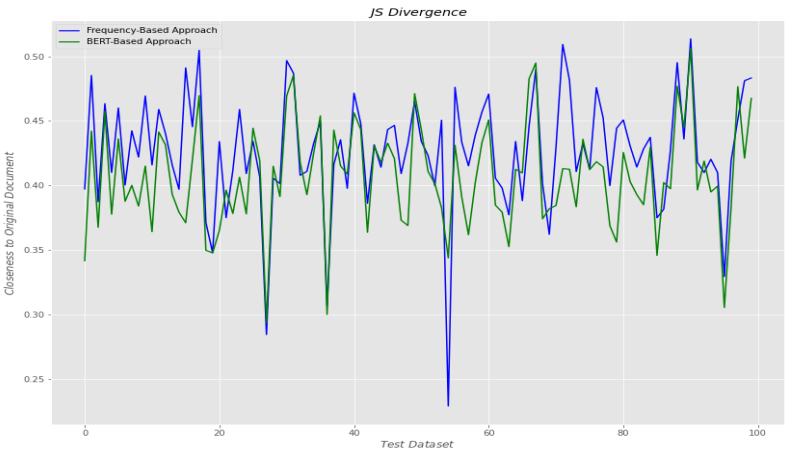


Figure 6.8: Experimental results on 100 patients clinical notes with JSD

| **Models** | **100 patients** | **Cosine similarity** | **Jaccard similarity** | **KL divergence** | **JS divergence** |
|---|---|---|---|---|---|
| Baseline | Mean Value | 0.51835 | 0.7110 | 0.8916 | 0.4258 |
| BERT | Mean Value | 0.8818 | 0.9738 | 0.7947 | 0.4054 |

Table 6.1: Experimental Results on 100 patients clinical notes compared to baseline

# CHAPTER 7

# Conclusion

In this thesis work, we developed a CUI-based summarization Machine Learning model and its evaluation. All our methods are applied based on Extractive summarization techniques. The Extractive text summarization process extracts the main points of a text without any change to those points. It generates them word for word producing a subset of the original text's sentences. Abstractive summarization reproduces essential material in a new way after interpreting and examining the text using advanced Natural Language Processing techniques. Evaluation in medical summarization is at the most challenging range compared to others. A physician who authored a discharge note may find the obtained summary helpful.

On the other hand, a physician who is not acquainted with patient history may find it tedious in explanation. We applied two new evaluation metrics and showed that our attention-based summarization model using the BERT model is better than our chosen baseline. By applying as evaluation metrics, Cosine similarity and Jaccard similarity show the architecture achieves better results on a set of MIMIC-III clinical notes. Overall, we used four evaluation metrics shown in Figure 6.1 and Figure 6.2, and as a result, we can emphasize the advantage of the attention-based mechanism on text summarization.

# CHAPTER 8

# Limitations and Future Work

Recently in Natural Language Processing has a significant impact on ML algorithms' deployment in the healthcare sector. Particularly for medical summaries, which can now be better processed than ever before. As discussed in the second section, the transformer model's advantage is that a significant trust level can be achieved by improving the model to further accurate automated understanding. Distributed models have given a strong foundation for next-generation Biomedical Natural Language Processing. So our summarization model can be developed more by using a new feature of NLP. Several propositions can be present in future work.

Furthermore, as a limitation, we can mention our work's evaluation procedure since we did not have a gold standard for comparison. Can be applied new method or metrics to show our the model works quite well. It is a crucial issue for doctors to review the important information of patients. This is a challenge to show that our summary consists of vital information and our model works with high accuracy. This is part of ongoing research experimentation. For that reason, the research about this topic can go further according to cutting-edge technology innovations day by day. Besides, the utilized clinical attention-based model is fine-tuned on the MIMIC-III dataset, and our model may not work very well for other kinds of structured clinical notes dataset. Also, this part can be added to future work.

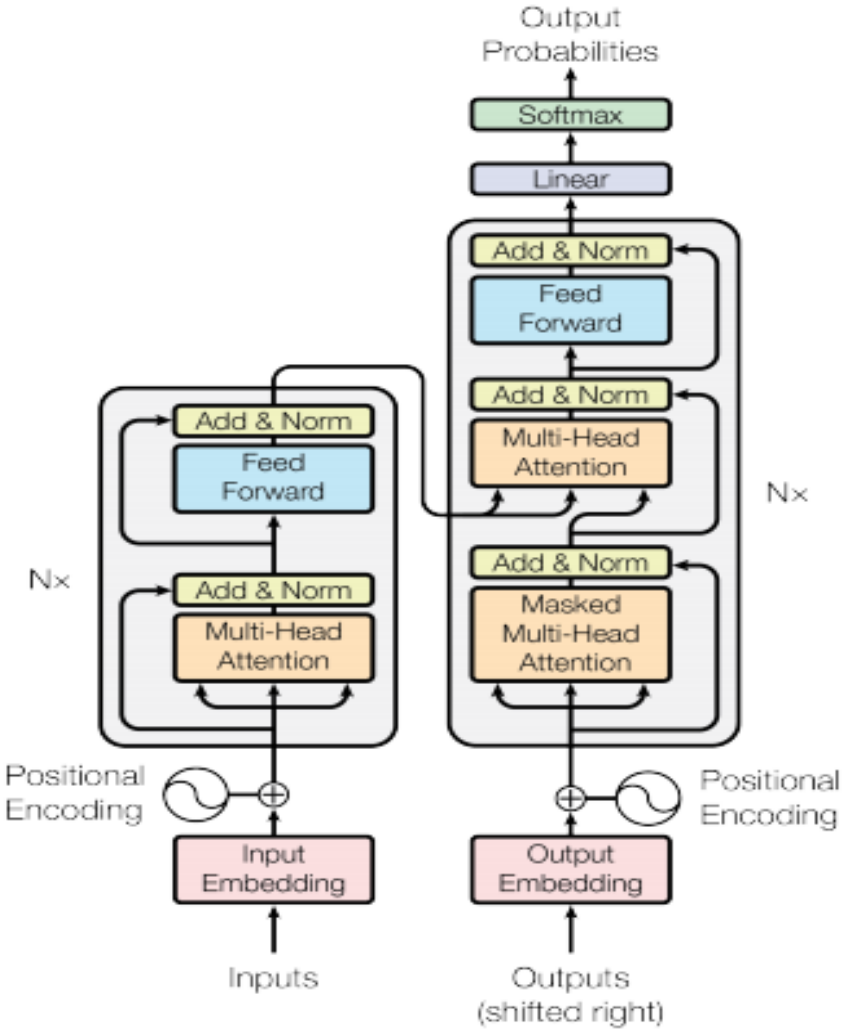


Figure 8.1: Transformer Model - model architecture [39]

# Bibliography


[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. ArXiv, abs/1706.03762

[2] Hiroki Nakayama et al. doccano: Text Annotation Tool for Human. Software available from https://github.com/doccano/doccano. 2018. url: https://github.com/doccano/doccano

[3] http://neuralnetworksanddeeplearning.com/chap2.html

[4] Wikipedia contributors. (2021, February 18). Cosine similarity. Wikipedia. url: https://en.wikipedia.org/wiki/Cosine_similarity

[5] Rizvan Saatov. (2021). rizvansaatov/Attention_based_Summarization_Approach_of_Clinical_Notes. GitHub. https://github.com/rizvansaatov/Attention_based_Summarization_Approach_of_Clinical_Notes

[6] Wikipedia contributors. (2021b, February 18). Jaccard index. Wikipedia. https://en.wikipedia.org/wiki/Jaccard_index

[7] Prabhakaran, S. (2020, October 11). Cosine Similarity - Understanding the math and how it works? (with python). ML+. https://www.machinelearningplus.com/nlp/cosine-similarity/

[8] Vaswani, A. (2017, June 12). Attention Is All You Need. ArXiv.Org. https://arxiv.org/abs/1706.03762

[9] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. ArXiv, abs/1706.03762.

[11] Kovaleva, O., Romanov, A., Rogers, A., Rumshisky, A. (2019). Revealing the Dark Secrets of BERT. EMNLP/IJCNLP.

[12] Itti, Laurent, Christof Koch, and Ernst Niebur. "A model of saliency-based visual attention for rapid scene analysis. "IEEE Transactions on Pattern Analysis & Machine Intelligence 11 (1998): 1254–1259.

[13] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. CoRR, abs/1409.0473.

[14] Luong, T., Pham, H., & Manning, C.D. (2015). Effective Approaches to Attention-based Neural Machine Translation. ArXiv, abs/1508.04025.

[15] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130, 2017.

[16] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In Empirical Methods in Natural Language Processing, 2016.

[17] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017

[18] Vig, J. (2019). A Multiscale Visualization of Attention in the Transformer Model. ArXiv, abs/1906.05714.

[19] Wikipedia contributors. (2021c, February 26). Natural language processing. Wikipedia. https://en.wikipedia.org/wiki/Natural_language_processing

[20] Bhatia, N., & Jaiswal, A. (2015). Trends in extractive and abstractive techniques in text summarization. International Journal of Computer Applications, 117(6).

[21] Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. Comput. Linguist., 28(4):399–408.

[22] Weiguo Fan, Linda Wallace, Stephanie Rich, and Zhongju Zhang, “Tapping into the Power of Text Mining”, Journal of ACM, Blacksburg, 2005.

[23] D. Das and A. F. Martins, "A survey on automatic text summarization," Literature Survey for the Language and Statistics II course at CMU, vol. 4, pp. 192-195, 2007.

[24] Farshad Kyoomarsi, Hamid Khosravi, Esfandiar Eslami and Pooya Khosravyan Dehkordy, “Optimizing Text Summarization Based on Fuzzy Logic”, In proceedings of Seventh IEEE/ACIS International Conference on Computer and Information Science, IEEE, University of Shahid Bahonar Kerman, UK, 347-352, 2008.

[25] Allahyari, M., Pouriyeh, S.A., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., & Kochut, K. (2017). Text Summarization Techniques: A Brief Survey. ArXiv, abs/1707.02268.

[26] Madhavi K. Ganapathiraju, “Overview of summarization methods”, 11-742: Self-paced lab in Information Retrieval, November 26, 2002.

[27] Klaus Zechner, “A Literature Survey on Information Extraction and Text Summarization”, Computational Linguistics Program, Carnegie Mellon University, April 14, 1997

[28] Berry Michael W., “Automatic Discovery of Similar Words”, in “Survey of Text Mining: Clustering, Classification and Retrieval”, Springer Verlag, New York, LLC, 24-43, 2004.

[29] Joel larocca Neto, Alex A. Freitas and Celso A.A.Kaestner, "Automatic Text Summarization using a Machine Learning Approach”, Book: Advances in Artificial Intelligence: Lecture Notes in computer science, Springer Berlin / Heidelberg, Vol 2507/2002, 205-215, 2002.

[30] Canasai Kruengkari and Chuleerat Jaruskulchai, "Generic Text Summarization Using Local and Global Properties of Sentences", Proceedings of the IEEE/WIC international Conference on Web Intelligence (WI’03) , 2003.

[31] Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, page 703–710. AAAI Press.

[32] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

[33] Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence, 2(3):258–268.

[34] Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In IJCAI, volume 7, pages 1776–1782.

[35] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst., 30(1–7):107–117.

[36] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, page 121–128, New York, NY, USA. Association for Computing Machinery.

[37] Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '95, page 68–73, New York, NY, USA. Association for Computing Machinery.

[38] Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto, "Automated Summarization Evaluation with Basic Elements", In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), 2006.

[39] J. Sun, Y. Wang and Z. Li, "An Improved Template Representation-based Transformer for Abstractive Text Summarization," 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207609.

[40] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9, 1735-1780.

[41] Alsentzer, E., & Kim, A. (2018). Extractive Summarization of EHR Discharge Notes. ArXiv, abs/1810.12085.

[42] Wikipedia contributors. (2021a, February 5). Long short-term memory. Wikipedia. https://en.wikipedia.org/wiki/Long_short-term_memory

[43] Miotto R, Li L, Kidd BA, Dudley JT. Deep patient: an unsupervised representation to predict the future of patients from electronic health records. Scientific reports. 2016 May 17;6:26094.

[44] Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, andMark RG, MIMIC-III, a freely accessible critical care database, 2016,DOI: 10.1038/sdata.2016.35.

[45] Nils J. Nilsson, INTRODUCTION TO MACHINE LEARNING,1998,Robotics Laboratory Stanford University

[46] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement Learning: A Survey, 1996,Journal of Artificial Intelligence Research, Vol 4, (1996), 237-285, arXiv:cs/9605103

[47] Hu, Y., Huber, A.E., Anumula, J., & Liu, S. (2018). Overcoming the vanishing gradient problem in plain recurrent networks. ArXiv, abs/1801.06105.

[48] Tomas Mikolov et al, 2013, Efficient Estimation of Word Representations in Vector Space, In Proceedings of the International Conference on Learning Representations (ICLR 2013)

[49] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, Enriching Word Vectors withSubword Information, 2016,arXiv,arXiv Preprint: 1607.04606

[50] Jeffrey Pennington, Richard Socher, Christopher D. Manning, 2014, GloVe: Global Vectors for WordRepresentation, Proceedings of the 2014 Conference on Empirical Methods in Natural Language-Processing (EMNLP), Pages: 1532–1543

[51] Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17.doi:10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2

[52] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. EACL.

[53] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. NeurIPS.

[54] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. ArXiv, abs/1910.03771.

[55] Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, andMark RG, MIMIC-III, a freely accessible critical care database, 2016,DOI: 10.1038/sdata.2016.35.

[56] Tal Baumel et al, 2017, Multi-Label Classification of Patient Notes a Case Study on ICD Code Assignment, In AAAI Workshop on Health Intelligence, arXiv Preprint, arXiv:1709.09587

[57] Shi et al,2017,Towards automated ICD coding using deep learning. arXiv preprintarXiv:1711.04075.

[58] Kanwal, N., & Rizzo, G. (2021, October 01). Attention-based Clinical Note Summarization. Retrieved from https://arxiv.org/abs/2104.08942v2

[59] James Mullenbach et al, 2018, Explainable Prediction of Medical Codes from Clinical Text, Computational Linguistics: Human Language Technologies, Volume 1, Pages: 1101–1111

[60] Stefano Macrino, Carmelo Velardo, Giuseppe Rizzo, "A Dilated CNN Approach for Coding of Clinical Notes", Submitted to Transactions on Computing for Healthcare, 2020

[61] Lipsky Gorman et al, 2010, Section classification in clinical notes using supervised hidden Markovmodel, In Proceedings of the 1st ACM International Health Informatics Symposium, 744–750. ACM

[62] Ramakanth Kavuluru et al, 2017, An empirical evaluation of supervised learning approachesin assigning diagnosis codes to electronic medical records. Artificial intelligence in medicine65(2):155–166.

[63] Dilated Convolution Networks for Classification of ICD-9 based Clinical Summaries - Webthesis. (2020a). Https://Webthesis.Biblio.Polito.It/14400/. https://webthesis.biblio.polito.it/14400/

[64] Yang et al, 2016, Hierarchical attention networks for document classification. In Proceedings ofNAACL-HLT, 1480–1489

[65] CleanPNG - HD png images and illustrations. Free unlimited download. - CleanPNG / KissPNG. (n.d.). Retrieved from https://www.cleanpng.com/

[66] Pubmed. (n.d.). Retrieved March 14, 2021, from https://www.pubmed.ncbi.nlm.nih.gov/

[67] Nielsen, M. (1970, January 01). Neural networks and deep learning. Retrieved March 14, 2021, from http://neuralnetworksanddeeplearning.com/chap2.html

[68] Wikipedia contributors. (2021f, March 8). Kullback–Leibler divergence. Wikipedia. https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence