2020

Evaluating BERT Embeddings for Text Classification in Bio-Medical Domain to Determine Eligibility of Patients in Clinical Trials

Saurabh Khodake

Technological University Dublin

Evaluating BERT Embeddings for Text Classification in Bio-Medical domain to determine eligibility of patients in Clinical Trials

Saurabh Khodake

D18129553

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Data Analytics)

31-08-2020

DECLARATION

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed:

Date: 31 October 2020

ABSTRACT

Clinical Trials are studies conducted by researchers in order to assess the impact of new medicine in terms of its efficacy and most importantly safety on human health. For any advancement in the field of medicine it is very important that clinical trials are conducted with right ethics supported by scientific evidence. Not all people who volunteer or participate in clinical trials are allowed to undergo the trials. Age, comorbidity and other health issues present in a patient can be a major factor to decide whether the profile is suitable or not for the trial. Profiles selected for clinical trials should be documented and also the profiles which were excluded. This research which took over a long time period conducted trials on 15,000 cancer drugs. Keeping track of so many trials, their outcomes and formulating a standard health guideline is easier said than done.

In this paper, Text classification which is one of the primary assessment tasks in Natural Language Processing (NLP) is discussed. One of the most common problems in NLP, but it becomes complex when it is dealing with a specific domain like bio-medical which finds presence of quite a few jargons pertaining to the medical field. This paper proposes a framework with two major components comprising transformer architecture to produce embedding coupled with a text classifier. In the later section it is proved that pre-trained embeddings generated by BERT (Bidirectional Encoder Representations from Transformers) can perform as efficiently and achieve a better F1-score and accuracy than the current benchmark score which uses embeddings trained from the same dataset. The main contribution of this paper is the framework which can be extended to different bio-medical problems. The design can also be reused for different domains by fine-tuning. The framework also provides support for different optimization techniques like Mixed Precision, Dynamic Padding and Uniform Length Batching which improves performance by up to 3 times in GPU (Graphics Processing Unit) processors and by 60% in TPU (Tensor Processing Unit).

Key words: Clinical Trials, Natural Language Processing, Transformer, BERT, Mixed Precision, Dynamic Padding, Uniform Length Batching, GPU, TPU

ACKNOWLEDGEMENTS

First, I would like to express my sincere thanks to my supervisor Dr. Sarah Jane Delany for guiding me through this journey and giving me the freedom to explore and follow my intuition. Thank you for showing faith in my abilities and showing the path ahead in difficult times.

I want to thank all TU Dublin professors who helped me build a solid foundation so that I could approach the thesis in the right frame of mind. I would also like to reserve special appreciation for the efforts of college staff, admin departments for the efforts they took during the pandemic to ensure the course was not impacted.

I want to express my gratitude to prof. John Gilligan and Dr. Luca Longo for helping me narrow down my area of interest. A special thanks to prof. Robert Ross for the guidance which helped me build techniques that were used in this research.

I want to thank my friends in Dublin who were family away from home.

Last but not least, I want to thank my parents Smita Khodake and Dilip Khodake for having faith in me so that I could pursue higher education, words will never be enough to express my gratitude for their unconditional love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AMP Automatic Mixed Precision

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers.

CNN Convolutional Neural Network

EHR Electronic health record

FP Floating Point

GLEU - General Language Understanding Evaluation

GPU Graphic Processing Unit

ELMO Embeddings from Language Models

KNN K-Nearest Neighbor

LSTM Longh Short Term Memory

MLP Multilayer Perceptron

NLP Natural Language Processing

NER Named Entity Recognition

POS Part Of Speech

QA Question Answering

RNN Recurrent Neural Network

TF-IDF Term Frequency - Inverse Document Frequency

TPU Tensor Processing Unit

USE Universal Sentence Encoder

XML eXtensible Markup Language.

1. INTRODUCTION

1.1 Background

Cancer clinical trials are bio-medical research studies performed on human participants to test out new treatments, interventions that can ensure its effectiveness and safety. To ensure their effectiveness it is very important to be aware of the background medical history and the underlying medical condition of the subject. Many patients in clinical practice don't show the same impact as in trials because of the comorbidity, side effects of any accompanying treatments. In certain cases, age can also act as a deterrent for the drug's effectiveness. A patient in clinical practice whose profile was excluded from the clinical trial cannot be administered with drugs used in trials as they might not show same results. Therefore, it becomes very important to have a demarcation as to which profiles are suitable to be administered with study intervention in accordance with clinical trials. However, when the disease one is dealing with is as varied and complex as cancer it becomes an arduous task to manually review the eligibility criteria (Milian, Ten Teije, Bucur, & Van Harmelen, 2011). In every clinical trial a list of protocols are developed and maintained, these protocols are then expected to be followed in any clinical practice who are trying to reproduce the benefits of these trials. If a model is built which will determine eligibility of a patient, it is essentially building a set of rules following on the lines of these protocols. Reviewing these guidelines manually cost high in terms of time and effort which might keep a potential patient from receiving the benefits of the research. This persuaded a team of researchers from Spain (Bustos & Pertusa, 2018) to devise a text classifier which can take text inputs named 'Study Intervention' and 'Study Condition' to provide eligibility of the patient for the drug in consideration. Study Intervention in the medical domain can be defined as study of impact and effectiveness of a 'prospective drug' on subject as a treatment or preventive measure for a disease. Whereas a Study Condition refers to a patient's health or ailments/disease they are suffering from.

1.2 Research Project

Clinical trials are performed with many constraints and therefore when it comes to extrapolating the merits of such treatments the rule book is not well-defined (Heneghan, Goldacre, & Mahtani, 2017). Not all participants who volunteer for the trials are included in the final research. Exclusion criteria can be defined as a condition that disqualifies a subject from participating in a clinical trial. When such trials are conducted on a massive scale by different personnel, maintaining uniformity becomes a challenge. The problem compounds when such inclusion or exclusion criteria are to be followed in clinical practice by a medical health professional because of the different work environments. The project wants to determine if short clinical text statements consisting of the intervention procedure, study condition, and eligibility criteria be used to identify whether the patient will be eligible or not for the clinical trial. Building a text classifier to analyse and process these text documents can serve the purpose of outlining the eligibility of participants for future clinical trials. This will help a medical professional decide whether to replicate the study intervention followed in trials in his/her clinical practice. In every clinical trial a certain list of protocols are developed and maintained, these protocols are then expected to be followed in any clinical practice while selecting or rejecting a subject for a trial. This research, by helping identify a patient's eligibility is internally developing these protocols.

The traditional approach to any text classification problem is by extracting features from the dataset or by encoding numeric vectors to represent the text. These methods though effective have proven costly (Lai, Xu, Liu, & Zhao, 2015). It also requires huge corpus of labelled data to have good representation of the data. This paper presents a solution that will eliminate the steps required to manually extract features by leveraging pre-trained embeddings generated using BERT to form contextual representation of the clinical statements that preserves the semantics and syntax of the language. The rationale behind this proposal is:

- To reduce cost spent in manually forming representation of text data
- To allow researchers with no medical domain knowledge build equally good models
- To reduce reliance on the size of the dataset to build good text embeddings

Research Question: "To what extent the pre-trained embeddings generated using BERT can provide more accurate results than embeddings generated from the dataset in classifying eligibility of cancer patients for clinical trials using their textual clinical statements?"

1.3 Research Objectives

In order to answer the research question following research objectives are set:

Literature Review Objective:

- To review state-of-the-art text analytics approaches used in the bio-medical domain
- To shortlist text embedding technique to generate vector representation for a sentence which can best represent the semantics and syntax of the language.
- To select classifier which will consume text embeddings and help predict binary label.
- To review common problems in the field of NLP and the measures needed to counter or solve them.
- To shortlist evaluation metric for embedding and text classifier

Primary Objective: To determine if pre-trained embeddings, fine-tuned on the dataset can perform as well or better than the embeddings generated from scratch. This will help researchers train their model with pre-trained embeddings as representation for the text instead of generating them from scratch.

Secondary Objective 1: To determine if character or n-gram embeddings used by fastext (Joulin et al., 2016) are better representation than word embeddings generated from word2vec.

Secondary Objective 2: To determine if drug names in study prescription encoded as categorical feature enhances the accuracy of the model. This will also help us evaluate if the name of the drug is influencing the eligibility rate in clinical trial.

Secondary Objective 3: To determine if study conditions are more important than study intervention in determining the eligibility criteria of a patient. In other words to test if a patient's health is more crucial to determine their eligibility than the drug

whose trial is carried out. This can also help uncover if certain drugs have relaxed criteria as compared to other drugs.

Secondary Objective 4: To determine if sentence embeddings improve accuracy of a text classifier as compared to word embeddings. Sentence embedding is a technique used to represent a sequence in an n-dimensional vector space. Unlike word embedding every word won't have representation, instead a single representation for the entire sentence. Accuracy obtained from both sentence and word embedding is compared to understand if there is a significant difference between the two techniques.

Secondary Objective 5: To determine if classifiers built within the pre-trained framework give more accuracy as compared to independent neural network.

Secondary Objective 6: To determine if embeddings fine-tuned on bio-medical data gives more accuracy as compared to generic embeddings.

Principally, using this technique the author wants to come as close as possible and better the accuracy score of the original research. The major difference in current approach and the previous research are the way the embeddings are generated. This research will be using pre-trained embeddings to represent the data and then fine-tune it on the data while training as opposed to generating embeddings from the same dataset. This method won't have the benefit of learning embeddings from the dataset itself which will be used for final classification. But the advantage of this approach is the independence from relying on the availability of huge volume of labelled data to produce good quality embeddings, apart from being cost effective.

1.4 Research Methodologies

The study began with primary research to understand if text embeddings generated by pre-trained models are better representation for short clinical texts from medical domain as compared to embeddings generated from the scratch. The dataset has been borrowed from previous research on clinical trials. This research will be performing a quantitative and deductive research by analysing numerical data to test this hypothesis.

To complete the literature objective many researches were explored in the field of NLP but specifically about text classification. Traditionally, in any text classification problem huge amounts of resources are spent in understanding data, handcrafting features and most importantly generating embeddings. Embeddings are vector representation for text data. In cases where data is scarce, representation of text data will never achieve a satisfactory level. Recently many pre-trained language models were open sourced with the purpose of open sourcing NLP (Natural Language Processing) research without expensive training pre-requisites. This was achieved by training models with a large corpus of data in an unsupervised manner. Different training techniques were employed for training the models but some popular techniques included predicting a word given the sequence that followed until that word. This helped model understand the structure of the language and form a statistical basis for predicting the probability of a word to appear next. These Language Models are then fine-tuned by training them on specific tasks which has a definite goal.

The first step in this research will be to form representation for the text documents using both the frameworks of pre-trained embedding models and embeddings generated from scratch. Since results from previous research on embeddings generated from scratch are already published, this research will concentrate on finding the best performance using pre-trained embeddings. The evaluation of best pre-trained embedding will be determined by how well they operate with a classifier to give the best accuracy and F1-score. Text documents or words are mapped into vector space using different embedding techniques. To accomplish the primary objective the researcher tries to understand if pre-trained embedding technique when used to encode bio-medical text documented in clinical trials can represent these texts more accurately. In the process it also tries to replicate and formulate standard clinical guidelines by processing the clinical trials undertaken using pre-trained models. The output of the research should enable clinicians to administer drugs to patients with more confidence.

This study wants to compare character/word/sentence embeddings to determine which type of embedding will work better to determine a patient's eligibility for a clinical trial. This is aligned with the secondary objective 1 and 4. Fastext (Joulin et al., 2016) can encode text in character or n-gram method, while certain pooling

policies can help us extract word or sentence embedding. These different methods of text representation will be compared against each other to find out which method can provide better text representation and more accuracy.

The clinical text or prescription comprises 2 major components i) Study Intervention (treatment) and ii) Study Condition (Patient's health condition). This research wants to understand if certain drugs have more eligibility rate as compared to others. In other words, to understand if the drug name is an influencing factor to determine patient's eligibility as compared to patient's health condition. This study will try to predict eligibility rate using study condition (encoded once as a string and as a categorical feature) and study intervention separately and compare them to find out if any one component is contributing more than the other. This experiment is in line with the secondary objective #2 and #3.

Every language model have their own final layer which can perform variety of NLP tasks such as text classification, Named Entity Recognition, Question Answering etc. Generally the final layers are used for the downstream task as well, but it will be interesting to test a separate classifier in combination with transformer to see if it can provide better accuracy compared to in-built classifier of language model. This experiment is designed in line with the secondary objective #5.

Vanilla language models like BERT are trained on generic English language, however there are some medical domain specific models are available which have been fine-tuned on medical corpus. Whether using these models for embedding generation improve the performance of the model forms an interesting question and hence it is included as the last secondary objective #6.

The dataset was arranged by (Bustos & Pertusa, 2018) in the research "Learning Eligibility in Cancer Clinical Trials Using Deep Neural Networks". In the existing research they trained embeddings from the dataset itself to eventually build a model to determine the eligibility of the patient.

1.5 Scope and Limitations

This research concerns with textual prescriptions comprising ‘Study Intervention’ and ‘Study Condition’ of cancer patients in the medical domain. The area of focus is to improve embedding representation for the dataset. Traditionally (Zhang, Jin, & Zhou, 2010) researchers are acquainted to design their own features which includes numeric presentation of text data. The area of interest for this research is to find out if pre-trained embedding can be a possible alternative for forming representation of the document. Figure 1.1 shows the scope of the research as a intersection of pre-trained embedding techniques used to generate representation of text documents derived from clinical trials to perform text classification.

This research cannot be used for cancer diagnostics. The inferences from this research are limited to cancer clinical trials only. However, the research design can be extended to other sectors. The assumption of this research is that embedding technique of any model is as good as the accuracy it provides while classification. There is a possibility of a scenario where embeddings though richer than other embedding technique performs poorly because of its unsuitability with the classifier. The one time training cost of contextual embeddings is higher compared to word2vec embeddings. But once the model is trained and deployed in production it can serve concurrent request with as much efficiency.

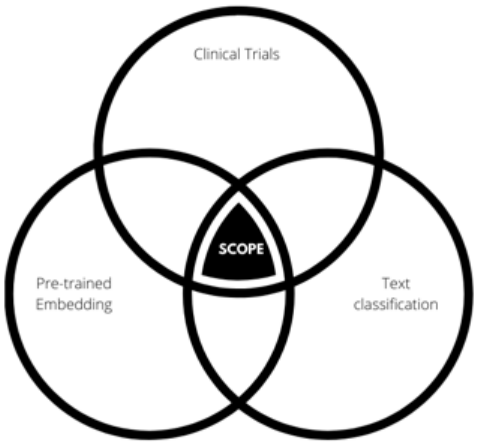

Figure 1.1 Scope of the Research

1.6 Document Outline

The document is organized in 5 chapters

Chapter 1 Introduction - This chapter sets the background of the research problem and lists down the goals which are intended to be accomplished in the study. It lists down the course of action to be taken along with the scope and the limitations of the research.

Chapter 2 Literature Review - This chapter reviews existing research in the field of Natural Language Processing. It takes a detailed view on the two most important component of this research:

i) Classifiers and

ii) Embedding techniques

It concludes with the shortcomings present in existing research.

Chapter 3 Experiment Design and Methodology - This chapter describes the research hypothesis and defines methods to solve the problem. It gives a detailed description on the dataset and the preparation steps needed before it can be used for embedding generation and modelling. The framework to be used is explained in detail and the different methods required for optimization. The evaluation metric required to judge a model are clearly defined. The chapter concludes with the strength and limitations of the design.

Chapter 4 Results and Discussion - This chapter focusses on the implementation of all methods listed in the design chapter and compares the results based on the evaluation metric defined in chapter 3. It links the results with the research objectives and gives a verdict on their success or failure.

Chapter 5 Conclusion - The final chapter summarizes the thesis and brings together all the achievements, contributions to the knowledge body and the impact it makes. The chapter signs off delineating future work in pipeline for the research and recommendations to readers.

2. LITERATURE REVIEW

In this chapter existing researches in Natural Language Processing specifically in the field of text embedding and classification are reviewed. It includes an overview on some popular techniques followed by their advantages and shortcomings.

2.1 Text classifiers

Natural Language Processing in healthcare is area of interest for many researchers. Right from summarizing clinical notes, mapping unstructured data in EHR(Electronic health record) to structured form, extractive QA (Lee et al., 2016) and paraphrase labeling (Bowman, Angeli, Potts, & Manning, 2015) in Natural Language Inference. Textual data are a rich source of information but can need plenty of resources and time to extract insightful information. The traditional approach to text classification relied on extracting representational features (Rogati & Yang, 2002), (Gon¸calves & Quaresma, 2004) from documents and training in supervised mode. Simple feature extraction techniques varied from bag of words (Y. Zhang, Jin, & Zhou, 2010) to Term Frequency - Inverse Document Frequency(TF-IDF) (Ramos et al., 2003), (Dostal & Je˘zek, 2011) model. One of the biggest challenges in text classification is limiting the number of features (Forman, 2004). Some convenient approaches to filter them include selecting top n-words based on frequency of occurrence (Baayen, 1992) or prioritising words depending on TF-IDF score. Forman (2008) used Bi-normal separation (Baillargeon, Lamontagne, & Marceau, 2019) also gained popularity as a technique to be used for feature selection by associating weights for different words. Number of features are normally high in NLP tasks and that can increase the cost of computation, Forman(2008) bypassed curse of dimensionality by using Hidden Markov with apriori information and achieved competent results.

Three major categorization (Minaee et al., 2020) of text classification can be done as follows:

i) Rule Based - Documents are classified into different categories using pre-defined rules. Rich domain knowledge or information on underlying data is required for defining such rules and maintaining them is a costly affair. Medical domain has widely

seen usage of MetaMap (Aronson, 2001) which identifies concepts by finding named entities from the text. Regular expression were used to design rules along with ontologies (Milian et al., 2015) to predict eligibility criteria. However, such techniques require domain expertise and are found to be rigid to adapt change.

ii) Machine Learning - Documents are classified based on the past observations and the features are designed using handcrafted features or extracted using other ML technique. These features are then trained in a supervised manner with a suitable ML algorithm. Naive Bayes, Support Vector Machine(SVM), Decision Tree are few examples of the classifier which can be used.

iii) Mixed - this is a hybrid approach of combining above 2 methods which gives us the advantage of finding hidden patterns and listing the obvious ones.
In this research all the experiments fall in the second category of Machine Learning except one.

Recurrent Neural Network (RNN) (Mikolov & Zweig, 2012) variants became quite common in most NLP researches due to its ability to capture long term dependency among sequences using its internal state memory. The basic RNN model however suffered from vanishing gradient (Hochreiter, 1998) problem arising out of continuous multiplication between matrices of weights and cost correction while backpropogating. The longer these errors have to propagate in layers they show more tendency to shrink leaving the model very small values to learn anything from. LSTM (Long Short Term Memory) did solve this problem to great extent when a 'forget gate' was added to focus only on the important part of the sequences by creating a connection between activation function of forget gate and the computation of the gradients in the network. LSTM's take longer to train and sometimes suffer from overfitting issues. They also have high computational cost and are sensitive to the type of parameter initialization techniques used.

Normally renowned for image classification (X. Zhang, Zhao, & LeCun, 2015) showed CNN(Convolutional Neural Network) can be used effectively even for text classification, achieving better results than traditional NLP techniques to extract features like bag of words or TF-IDF. While RNNs have connected the dots in

sequences better in long range dependencies, CNNs work better in detecting local and positionally invariant pattern (Che, Cheng, Sun, & Liu, 2017). For example a cow appearing in any portion of the image will still be a cow. The textual equivalent of this local invariance property can be explained better in this example. Consider a an example sentence “Study Intervention by drug A for a patient suffering from disease X”. The meaning of the sentence will not change even if the order of phrase changes. CNN takes advantage of this property to learn features on its own by sliding across a window of fixed size over the numerical representation of text and convoluting over the matrix. CNN can also be used to learn their own features (Johnson & Zhang, 2015) by the principle of convolution and pooling, eventually comparing if the features extracted can be associate with the label. CNN can work well with word or character level embedding (Johnson & Zhang, 2016) as shown in this research. Small sample labelled data with pre-trained embeddings and some additional character level features achieved state-of-the-art results (Wang, Wang, Zhang, & Yan, 2017). CNN did suffer from loss of information while performing operations like max pooling which can be compensated by dynamic pooling where the filter window to convolute is adjusted according to the size of its input and CNN sometime struggle to form long term dependencies. This was resolved to some extent by a hybrid model of CNN and LSTM (C. Zhou, Sun, Liu, & Lau, 2015), (P. Zhou et al., 2016) which used CNN to extract the sequence of higher-level phrase representation fed to an LSTM to get sentence representation.

Later in 2017 researchers at google brain used attention mechanism (Vaswani et al., 2017) to replace the complex encoder decoder combinations of recurrent and convolutional networks. Using attention, the model focussed only on certain section of the sequence instead of the entire sentence where maximum information lied while predicting the next word. Both these changes helped reduce the processing time considerably. A separate context vector is maintained for every target word and is used while predicting a word in a decoder. A context vector is bi-directional weighted sums of activation states where weights represent attention required for predicting a word. This change helped the model parallelize operations and tackle the long sequences dependency issues faced in recurrent neural networks.

The landscape of NLP changed with the advent of deep learning and achieved state-of-the-art results in many benchmark tasks. It allowed models to learn hidden patterns among the data and devised a technique to form representation of the text data. Sequential models like RNN suffered from high computational and time costs, CNN who are less sequential in nature suffered in performance when it came to long sentence length. Every next step depends on the output of the previous time step in sequential model which stalls training simultaneously. Transformer reduced the costs by parallelizing computation for every token in a sequence by employing attention mechanism. This allowed training of big models on high volume of corpus. OpenGPT (Radford et al., 2018) and its successor a transformer based model which could work on a wide variety of tasks such as text classification, textual entailment, question answering and semantic similarity assessment. While OpenGPT was trained in a unidirectional model, BERT which came later was a bi-directional model and outperformed in most NLP benchmark tasks and achieved high rankings in General Language Understanding Evaluation (GLEU) (Wang et al., 2019) score.

These Pre-trained Language Models were released publicly for researchers to use. Many models even provide end to end framework which include extracting features from data and downstream NLP tasks. With the development of such frameworks transfer learning (Pan & Yang, 2010) became popular where the learnings of these pre-trained frameworks can be transferred onto other applications. This reduced the need for intensive training with high volume of data. With transfer learning pre-trained models could adapt to new dataset with fine-tuning on relatively small data, which not only made possible for individual researchers to get good results but it also reduced the overall cost to generate embedding for these models.

As mentioned earlier the dataset used for this research was released by (Bustos & Pertusa, 2018) and have also published their research surrounding it. They used word2vec and fastext for representing text data into vector format. Word2Vec used neural network to form representation of words by learning their associations with other words from large corpus of text. Fastext is an extension to word2vec which learns embeddings of n-gram or characters in text instead of the whole word. While predicting eligibility of a patient for clinical trial neural network classifier CNN was the best performing neural network model, whereas KNN(K-Nearest Neighbour)

(Keller, Gray, & Givens, 1985) gave best performance among traditional machine learning model. However, lazy classifier algorithms like KNN are also guilty of time complexity and memory consumption (Soucy & Mineau, 2001) which is also one of the reason why the focus is not on KNN in this research. The embeddings were trained from scratch using only the dataset. They used gensim implementation Word2Vec and Fastext models to train their word embeddings using both skip gram and Continuous bag of words (CBOW) approaches. CBOW is a type of architecture used in word2vec model which tries to predict a word given other words in context. Skip-gram on the other hand tries to predict the context given the target word. Both training methods are available in word2vec to help form representation for these words. After the embedding were generated, they evaluated four classifiers Convolution Neural Network, K-Nearest Neighbour, Support Vector Machine and Fastext internal classifier. Since the area of interest for this research is deep learning, it will be comparing the results with CNN results of the original research. CNN classifier was the 2nd best classifier for (Bustos & Pertusa, 2018). The metrics they used for evaluation were Accuracy, Precision, Recall, F1score and Cohen's kappa. The benchmark scores obtained for CNN model was on an average 0.88 for all metrics and kappa value was 0.76 when 1 million records were used for training and testing.

2.2 Embedding Methods

All machine learning techniques require representation of text in numerical format so that it can be consumed by the neural networks. The representation of the words are learned by training models with enormous text data. The model tries to form representation of the word by taking into account the words adjacent to it. Words which are similar to each other will have their positions nearby to each other in vector space and vice versa for dissimilar words.

Numerical representation which can capture syntactic and semantics of the language in the most effective way possible are the hallmarks of a good embedding method. Traditionally researchers have spent a lot of time in formulating embeddings for a particular dataset before even moving to the actual task. Such methods have

worked well though, as it got desired results but came with an overhead cost. However, once the research was over the reusability of such embeddings are not high. Pre-trained embeddings have been around since some time, but they can struggle in getting good performance given the rigid nature of the embeddings. For eg. A word2vec model will have same representation for the word 'bank' in the phrase 'river bank' and 'bank deposit'. Embeddings trained specifically on the corpus in consideration will however capture the semantics of bank properly because the chances of word bank appearing in both the context is rare in one single dataset. A pre-trained model however doesn't have this liberty as it has been trained on huge corpus which saw 'bank' in both context possibly equal number of times. The final representation of the word is an average of all context the word appeared in.

Fine-tuning can solve the problem to some extent in 2 ways:

i) Update the embeddings of the words during training.
ii) Update the weights of the classifier using backpropagation

Modern word embedding techniques reduces the number of dimension from thousands to few hundreds. One of the important parameter in word embedding is deciding the number of dimension for representation which is best figured out empirically. Higher number of dimensions might bring better accuracy sometimes but it also brings in extra computational cost. A right balance needs to be figured out and is generally dependent on the dataset and the problem which needs to be solved. To generate dense vector representation of words, Word2vec needs a local context window parameter to define the number of words to consider while training embeddings. Context of the word *w* in consideration is learnt by *k* words in the vicinity of *w*. Minimum frequency of word also needs to be decided for model to consider the word for embedding. Such parameters will eventually decide how good the representations are for a model and remain key to the overall accuracy in the classification task. A researcher needs to make an educated guess on deciding the parameters and explore in a specific vector space to find the best possible combination.

Pre-trained embeddings have other challenges to consider such as unawareness of the context, ignorance about the domain. It becomes the role of the researcher to understand these challenges and the background of the pre-trained embeddings such as the corpus it was trained on, training methods used etc. to find a best fit for underlying data.

2.3 Types of Embedding

The success of any good NLP model relies heavily on the quality of text representation in vector space (Ling et al., 2017). Any dataset with more than 1000 unique words in vocabulary requires high volume of training data to form better representation of the language, which is not always possible. The embeddings can be derived from either **generic language** or **domain specific language**. Generic embeddings are trained on day to day English language corpus like news articles, WIKI, journals etc., whereas domain specific embeddings are specifically trained on a corpus related to a particular domain, in this case its bio-medical. The advantage of using domain specific embedding can be the domain knowledge which can be leveraged such as semantic relations and domain specific vocabulary.
Embedding can also be divided in 2 different types such as :

i) Contextual Embeddings – The embedding of the word is reliant on the context of the sentence.
ii) Non-Contextual Embeddings – The embedding of the word is independent of the context of the sentence.

2.3.1 Generalized Embedding

Most of the language models are trained on the general English language corpus consisting of news article, Wikipedia, text gathered by text crawlers, journals etc. The common aspect about all these sources is the proper syntactic and semantic rules being followed. The biggest advantage in training language model is that it can be trained in an un-supervised manner. There is no need for labelled data, instead a large corpus of data split across sentences and documents are enough. The reason why this is possible is the way language models are trained. They can form the input and target output from

within the training data itself. There are a variety of methods involved in training, some of the most simplistic versions are predicting next word given an input stream of words. This can be extended to predicting next sentence given an input sentence. Unidirectional model training flow from left to right while bi-directional training will include training from both ends. Initial representation for input in many language model starts with character embedding, eventually forming representation for the word. The bottom layers of the model will be rich in syntactic information of the model while the top layers will be rich in both syntactic and semantic information. The embeddings can be extracted in various combinations from hidden layers depending on the type of task at disposal.

Fastext embeddings, an n-gram variant of word2vec is another popular technique to be used for representation of the dataset. Fastext will be used as an extension of word2vec to understand if character or n-gram representation of word yields better results. Instead of generating embeddings for words directly, it uses n-gram character or characters to form representation of words. A word like 'idiosyncratic' with n=3 will be represented by fastext as <id, idi, dio, ios, osy, syn, ync, ncr, cra, rat, ati, tic, ic> (Liao, Shi, Bai, Wang, & Liu, 2016). The n-gram words are then trained using skip-gram model to learn embeddings. Fast text can deal with out-of-vocabulary words as any unseen word will be broked down into n-grams or single character to get embedding which is the biggest advantage of using fastext. Gensim package provides methods to load a pre-trained word model for word2vec. Fastext package from facebook provides functions to train your own embeddings, and it expects dataset to be in a certain way in a flat file to be compatible with the package methods.

BERT was trained on general English corpus and fine-tuned on NLP tasks like next sentence prediction and masked language modelling (Liao et.al., 2016). It was completely trained in an unsupervised manner. Models prior to BERT needed labelling or at least part of speech taggers for identifying word types. It's a bi-directional model which trains words from both directions to get a better understanding of the impact on how all words have on the context. It is well known that a word's placement at different position in the sentence changes its POS (Part Of Speech) tagging. Unidirectional flow of words meant that words which are yet to be seen can't be used

to enrich the composition of the embedding of the word under development. This could potentially mean loss of information. Transformers employed by BERT ensures that attention is given on words or phrases which are more important than others, absence of such words or phrases could increase ambiguity of the sentence and is used as a hallmark of the phrase by BERT to determine a word's importance. Transformers can look at target word and understand the context of all the other words which are related to the original word. It tackles issues like co-reference mentioned in section 2.3. The entity or target word can be focussed and the related pronoun or phrases referring back to them can be linked using transformers attention mechanism. This also takes care of the polysemous words by assigning weights to all related words to the target word. Every related word is given a weight as to how much meaning it is adding to the target word which can be captured as representation for the target word. A sentence describing word "bank" will get more weight associated to a term "river" as compared to other words, making it clear for the model to understand it is dealing with nature and not the financial institution.

Until now the entire focus has been on the target word but to ensure that rest of the words are not ignored and that an imbalance is not created. BERT uses Masked Language Modelling which will randomly as the name suggests mask a word and try to predict the hidden word. Textual entailment or next sentence prediction is a training process which involves pairing of sentences. The pairs can be a right or wrong and the training of the model is accomplished by letting them identify if the pairing are right or wrong by giving prediction score. This exercise helps BERT understand context at sentence level and is beneficial for Natural Language Inferencing (Devlin, Chang, Lee, & Toutanova, 2019). BERT stores information about a sentence in a special token represented as [CLS] and it called as a classification token.

BERT released large and base (small) models with 24 and 12 transformer layers respectively. The large BERT variant consisted of 1024 layers with 16 attention head. The whole setup costs up to 340 million parameters. The base or light variant of BERT consisted of 768 layers with 12 multi-attention head and 110 million parameters. There are minor improvements in BERT large model but the cost of computation are considerably high as compared to the benefits.

2.4.1 Domain Specific Embedding

BERT has been trained on generic English corpus it is expected that medical terminologies like disease names and drugs will not be something BERT might have encountered in training. The word distribution will shift from generic English corpora to biomedical domain when it comes to training on dataset related to clinical trials. BioBERT a BERT variant fine-tuned on medical domain has significantly outperformed BERT in biomedical text mining tasks (Wada et al., 2020). It has the same architecture and vocabulary as the BERT model. This is important to preserve the pre-learning achieved by BERT model and not compromise the weights associated with the vocabulary. Bio-medical text mining tasks include biomedical relation extraction (2.80% F1 score improvement) and 12.24% Mean reciprocal rank improvement in biomedical question answering (Wada et al., 2020).

2.4 Gaps in Research

Referring back to the original research in regards to this dataset, some challenges arises when dealing with a non-contextual embedding technique like word2vec. **Polysemy** is the ability of a word to have multiple meanings completely depending on the surrounding semantics. Embedding techniques like Word2Vec used in the original research have one representation for unique token which is an average of all the different manifest of the occurrence of the word. It will get dominated by the maximum frequency of the polysemic version of the word. It will be closer to cluster of words which occurred with the maximum frequency. **Coreference resolution** (Ng & Cardie, 2002) is the activity to find all expressions which are referring to the same entity and is an important aspect of a good language model. In clinical trial dataset health condition is often referred to a patient and as a result it becomes important to dissect which ailments and diseases are being referred to a patient. Coreference resolution can be further be split into anaphora and cataphora (Moradshahi, Palangi, Lam, Smolensky, & Gao, 2019) resolution. Situations where it becomes difficult to identify entity or person or item referred in the later part of the text by pronoun or noun phrase is called anaphora.

Example: Adam read the bible, he was impressed by the depth in the scriptures.

While a pronoun or noun phrase used even before the entity is mentioned is called cataphora.

Example: He regretted lying to me, Mike said in a hushed tone.

In this example the pronoun ‘He’ is referring to a third person and ‘me’ is referring to person Mike, both coming before the entity being referred to. Coreference issues are important to this research because when a bio-medical named entity is referenced by a pronoun or noun phrase in any part of the sentence the embeddings should be able to capture these references.

Lexical ambiguity is the property of word to have different meaning at different situation. Also known as semantic ambiguity, it is found at sentence level as words with multiple meaning form a sentence which makes it difficult for the model to understand. **Multi-sential ambiguity** is the challenge to separate reference being made across different entities. But there are some interesting aspects of natural language which help solve such conundrums such as co-occurrence. Co-occurrence states that words similar or related to each other tend to be in proximity in a sentence. Relatedness of the actions associated with an entity can be a good indicator of what the entity is in the context. A car could be turned on, off, driven, parked but so can be some other vehicle. It still however narrows down the search for the entity to automobiles.

3. EXPERIMENT DESIGN AND METHODOLOGY

The purpose of this chapter to design a framework which is end to end capable to perform computations to transform text into numerical representation that can be used for training a text classifier. The framework encompasses techniques which the author believes is a good fit to be in line with the research aim and literature review. Figure 3.1 gives a brief overview on the topics to be discussed:

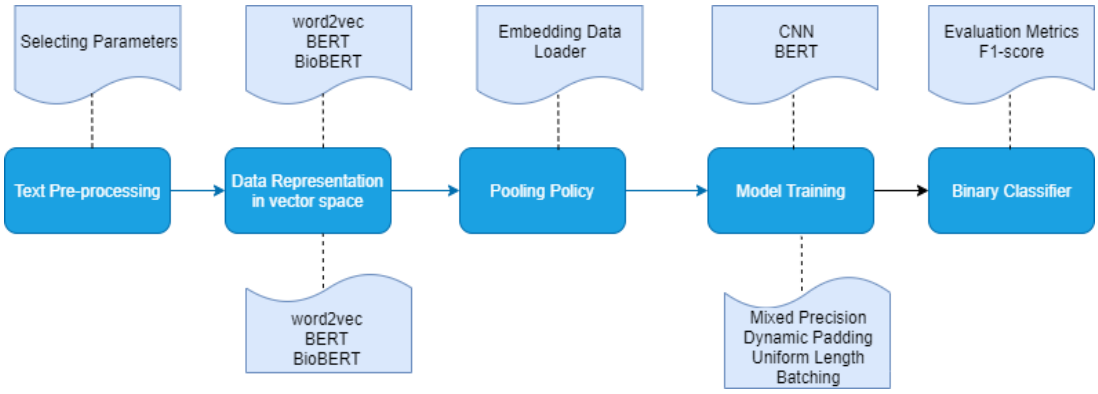

Figure 3.1 Design Flow

Research Hypothesis

This research aims to investigate the effect of using pre-trained embeddings instead of training them from dataset required for text classification. This study wants to compare design proposed in this research against the best neural network model from the original research (Bustos & Pertusa, 2018) that trained word2vec embeddings on CNN classifier.

Hence, the alternate hypothesis of the research states:

H_1: If pre-trained language model BERT is employed instead of convolutional neural network built on word2vec embeddings to predict cancer patient's eligibility for clinical trial using their textual medical prescription then the prediction F1-score, accuracy increases.

3.1 Data Understanding

The dataset was compiled by (Bustos & Pertusa, 2018) for their research on interventional clinical trials protocol on cancer. This dataset was originally sourced from https://clinicaltrials.gov in XML format. Each XML file represented details of a single clinical trial. Study Intervention, Study Condition and Eligibility were extracted from every clinical trial file. Text was in unstructured format with polysemy and synonymy problems (Bustos & Pertusa, 2018). Study intervention and condition entities were merged into short texts of clinical statements, followed by the eligibility criteria as a dependent column to be predicted. A subject is either eligible or ineligible for the drug in consideration for the clinical trial. Table 3.1 shows some sample records from the dataset.

Eligibility Criteria	Clinical Statement
Eligible	Study interventions are fludarabine phosphate . Hodgkin lymphoma diagnosis and induction failure minimal residual disease greater than or equal to one marrow blasts by morphology after induction persistent or recurrent cytogenetic or molecular evidence of disease during therapy requiring additional therapy after induction to achieve remission
Not Eligible	Study interventions are questionnaire administration . Stage iii squamous cell carcinoma of the hypopharynx diagnosis and although there are no known adverse effects of black
Not Eligible	Study interventions are bortezomib . Ovarian mucinous cystadenocarcinoma diagnosis and concomitant medications known to inhibit or induce cytochrome pfour hundred and fifty family three subfamily polypeptide four threeafour are to be avoided
Eligible	Study interventions are KW-2450 . Solid tumor diagnosis and subjects with inflammatory diseases of the gastrointestinal tract or malabsorption syndrome

Table 3.1 Sample dataset

3.2 Data Description

Total number of records in the dataset are 1 million. Study intervention can be defined as the evaluation of treatment to cure a patient. Approximately 15,000 different drugs are being examined in the clinical trials. Top 100 drugs by frequency consisted 70% of the dataset. Study conditions represent underlying health condition of the patient. There are around half million unique patient conditions. The unique combination of study intervention and condition obviate any rule based solution to be effective without any domain expertise, even. The disease this dataset was primarily concerned with was cancer and its different types. Many terms which are strictly found in the medical domain only are expected, which makes it difficult for the pre-trained model usually trained only on the general English language corpus to understand the semantics.

Clinical statement which is a combination of study intervention and condition had average length of 24 words with the shortest statement being 6 words and longest being 439 words. Distribution of statements had variance of 170, skewness of 3.1 and kurtosis of 21.

Study Intervention are shorter as compared to study condition, with average word count for study intervention is 4 and 16 for study condition. The vocabulary count or unique words in the dataset is 33,599.

Description	Min	Max	Average	Variance	Skewness	kurtosis
Length of character in Clinical Statements	25	2742	152	5991	2.41	15.87
Number of words in Clinical Statements	4	439	21	133	2.56	18.48

Table 3.2 Summary Statistics on Clinical Statements

Distribution of the dependent variable ‘Eligibility’ was perfectly balanced with 50% representation for both values indicating the binary nature of the problem.

3.3 Data Preparation

The clinical trials have been gathered over a long time period, resulting into different reporting format and styles by different individuals. Sentences were split at period but not at period found at abbreviations or medical notations. Punctuations, single character words, non-alphanumeric and whitespaces were removed from the text. Stop words were retained to preserve the semantics of the sentences.

There were no null values in study intervention but around 40 thousand records had null values in study conditions. Retaining these records didn't make sense as the patient's health condition seemed crucial to decide if he or she was eligible or not for the trial. The vocabulary consisted of around 33,000 words approximately. Some records with less than 4 characters in study conditions were also removed. The first aim behind minimal data pre-processing was to reduce reliance on cleaning data for better accuracy. The second aim was to preserve the information and flow of the texts as far as possible. This is why stopwords like [and, or] weren't removed to conserve the semantics of the text documents. The third aim was to have a framework which will require minimum human intervention to build a good model so that any researcher with minimum NLP or medical domain knowledge can achieve decent results.

3.4 Pre-trained Embedding Generation

Pre-trained embeddings are trained on massive amount of data to form representation for language and are saved. They can be re-used on other tasks and therefore can also be considered as a form of transfer learning. There are different options available for pre-trained embeddings. In this research 2 pre-trained embedding techniques will be looked at:

i) Non-Contextual Embeddings - Word2Vec

ii) Contextual Embeddings - Bidirectional Encoder Representations from Transformers

3.4.1 Word2Vec Embeddings

Word2Vec embeddings for words can be generated by loading a pre-trained model in memory. There are different pre-trained versions of word2vec model available with different dimensions of embedding they generate. The word2vec version used is

trained on Wikipedia 2017, UMBC web base corpus and statmt.org news dataset encompassing 1 million word vectors, generating 300 dimensions per word token.

3.4.2 BERT Embeddings

There are different ways how embeddings can be generated using BERT. Some of the important techniques are listed and evaluated as follows.

3.4.2.1 BERT as a service

i) BERT pre-trained model of choice needs to be downloaded.
ii) Start BERT service using command line interface from the directory where BERT is downloaded. Number of workers define how many threads will be handling requests for generating embeddings. Number of workers need to specified, more than one in case one wants to parallelize operations. If the number of worker threads is 2, it means that it can handle 2 concurrent requests from client. If more than 2 requests are made they will be handled by a load balancer. Currently, BERT as a service is proof checked with tensorflow version 1 only.
iii) Once the server is up and started, client service can be started at the same machine or other machine using IP address of the server. An instance of BERT client can be instantiated to be used for making requests for BERT embeddings. The request can be sent using client api with one or multiple sequence of text. In response BERT embeddings of the dimension (batch_size, max_len, hidden_layer_length) are received. A small snippet of command line interface can be seen in Appendix A Figure 7.1.
Features: Good for small datasets. The embeddings generation and storage takes lot of computational resources and memory. The process of request and response hinders faster embedding generation.

3.4.2.2 BERT API

It is a pytorch implementation of transformer with a wrapper provided by sklearn. In this case one doesn't have to run a server client pair, instead one can instantiate class BertClassifier provided by sklearn. It also provides the ease of downloading model from internal call with support to almost 16 models. The embeddings generated will be

used for classification internally. Parameters that are tunable include number of layers, max_seq_len, learning rate etc.

Features: Good for big datasets and training. It provides an option to add a static Multilayer Perceptron (MLP) layer and the number of neurons in it. However, no provision to attach a custom deep learning network like CNN etc.

3.4.2.3 Pytorch HuggingFace Trainer

Above approaches were good for small dataset but however it presented lack of flexibility in terms of designing your own network. In this section will list down two approaches to generate BERT embeddings using transformer API's provided by Huggingface and train them on either a custom neural network classifier or BERT internal classifier.

3.4.2.3.1 BERT embeddings with Customized Neural Network classifier

A general purpose transformer architecture provided by huggingface is initialized. The immediate benefits of choosing this is the flexibility to define custom pooling policy. Pooling policy can be defined as extracting embeddings from one or more combination of hidden layers from a transformer architecture. Class PoolPolicy which will have methods to provide complete customization over number of layers to extract the output from a pre-trained model, and the weights associated with each layer. In the end it also provides an option to squeeze layers to obtain sentence embeddings.

A utility called Embeding Data Loader is designed which will take care of the whole embedding process right from tokenizing to generate embeddings. It provides flexibility to define maximum number of tokens to be taken from the sequence from both ends. The embeddings will be pooled from the layers specified by utitlity PoolPolicy. A loader will be defined which will iteratively invoke a new batch. Once a batch is invoked, the transformer will generate embeddings for the whole batch train it and discard it, so that memory is freed for the next batch. This is the most crucial step which bypasses performance issues like memory overflow. Words which are out of vocabulary will be broken down into tokens and looked up from the standard BERT vocabulary for n-gram tokens or further broken to be a character embedding. Now for

the requirement to comply with one vector per word, it needs to combine the embeddings generated from disintegrated tokens called as sub-tokens. The 2 methods they can be used is by either summing embedding values of all the sub-tokens or take an average of all sub-tokens to represent the original token.

Pre-trained model will be loaded from AutoTokenizer API provided by huggingface. Lastly the research defines customized model and instantiate it. The training will start with data loaded from EmbdedingDataLoader which will internally call the PoolPolicy class to build with the configurations set. Readers interested in understanding Pooling Policy and data embedding loader can find it in Appendix A Figure 7.2.

Features: Supports training for large dataset and can also be extended to train on a custom neural network architecture.

3.4.2.3.2 BERT embeddings with BERT classifier

The last framework designed uses a huggingface API's is an all BERT embedding generator and classifier. The challenges with previous implementation of all BERT model was memory limitation and flexibility in defining a custom network as a classifier. This limits the volume of the data it wants to train. It supports distributed training across GPU and support for an important optimization technique called mixed precision which is going to be discussed in next section. It also provides data collator support which can help us design batching process using the optimization techniques like dynamic padding discussed in detail in next section.

Features: Supports training for large dataset and can be integrated with optimization techniques like mixed precision and dynamic padding.

3.5 Methods to optimize embedding

Optimization technique called as **Mixed Precision** Training (Micikevicius et al., 2018) is used in this research. Typically, high number of layers and parameters see improvement in the performance metric but also results in increase in consumption of memory and computational costs for training of the model. Most of the deep learning models use float32 dtype (FP32) occupying 32 bits in memory, but there are cheaper options of float16 (FP16) which takes half the memory as compared to former. Float16

improves performance by 3 times in GPU and by 60% in TPU (Le Grand, Götz, & Walker, 2013). Although this offers performance benefits it suffers from stability issues when it comes to training quality. This is where mixed precision offers a compromise at cases where it uses float16 for faster step time. However, some variables and few computations are still kept in float32 (Floating Point 32) to prevent any drop in quality. Latest GPUs have specialized hardware to perform faster operations for float16 (Floating Point 32) and therefore these low precision dtypes will be used wherever possible. In the research (Le Grand et al., 2013) proved that mixed precision provides faster computations without any degradation in accuracy or hype-parameter tuning. NVIDIA Apex provides API for Automatic Mixed Precision to enable Tensor Core-accelerated training. Mixed Precision can be enabled by setting the optimization level and model as parameters.

Static vs Dynamic Padding

Static Padding: A major constraint for neural network is that batch size for all sequences should be of the same length so that a proper batch matrix representation can be built. But in real word scenarios text sequences have varied length. Therefore, it becomes necessary to have a fixed length cut-off for sequences who are longer than the cut-off mark. Sequences shorter than the cut-off need to be padded with dummy values generally zeros so that shapes of all sequences in a batch are equal.

Dynamic Padding is another method that can be evaluated to address padding issue. Truncating sentences is loss of information while padding them with zeros is unnecessary information being tagged which will still undergo computation and loss will be calculated for a vector space which is in reality devoid of information. Dynamic padding combats both the issues by considering one batch at a time and the max sequence length will be set at the maximum length of the sequence in that particular batch. Therefore, no loss of information will happen for sequences and padding will be restricted to minimum as compared to the original strategy. Normally dynamic padding is popular for text generation and language modelling but it can be extended for classification tasks as well. Trainer class from huggingface provides support for dynamic padding. This strategy will be further optimized by using an approach called as **Uniform length Batching** where batches of similar length sequences are grouped, so the extreme cases of small sequences in a mini batch which

also accidentally has one longer sequence will be avoided. This will help combat long unnecessary padding requirement.

3.6 Parameter Selection

Selecting the parameters such that it does justice to the task at hand is one of the most important task at hand for any NLP researcher. One such critical parameter will be the maximum length of sequence to be allow in the model. In ideal scenario one would like to keep the whole sequence, but that will result in performance and quality issues. Figure 3.2 shows distribution of words in a sentence across whole dataset, x-axis represents the length of words in each sentence and y-axis represents the count.

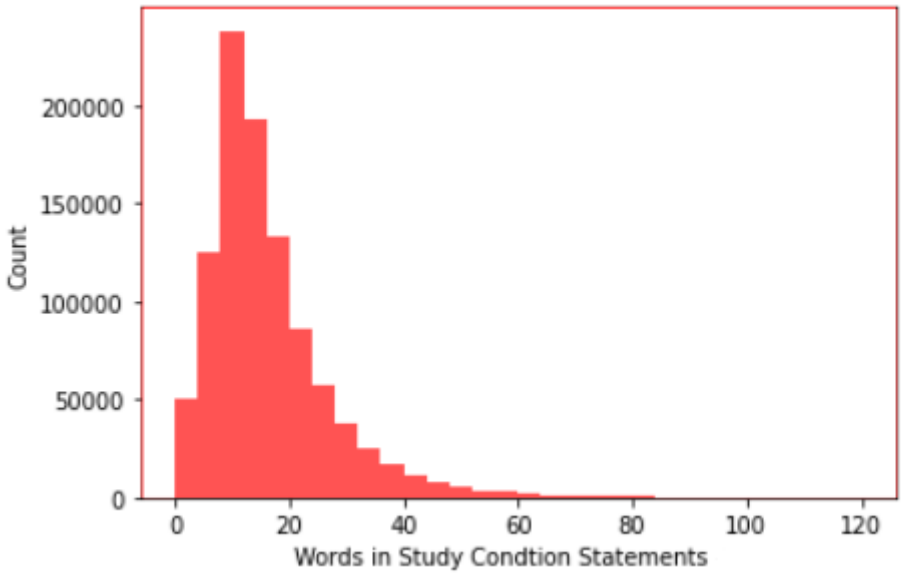

Figure 3.2 Word distribution across Clinical Statements

It can be seen in the figure 3.2 there aren't many sequences whose length is more than 70 words, empirically it is decided to set max sequence length as 75 words which turns out approximately 350 characters for all models where static padding have been used. Using this limit 97% of the clinical trials are covered. The original research used 1000 characters as maximum sequence length. In cases where dynamic padding is not used,

350 characters will be made as the hard cut-off limit. While using dynamic padding the researcher doesn't need to bother about the sequence length.

Word2Vec parameters - There are not enough tuning parameters available for pre-trained word2vec. The dimensional representation for each word is fixed and depends on the property of the pre-trained model. In this case every word will be represented with 300-D vector dimension. Words which are out of vocabulary will be represented by a 300 dimensional vector of zeros. Static padding will be used for word2vec with maximum length set at 350.

BERT parameters - In BERT maximum sequence length was kept at 350 when static padding was used, whereas for dynamic padding no limit was required. Batch size will be kept at 16 as anything above runs the risk of memory clogging. Ideally would have kept it higher as there were enough training example to reduce number of weight updates, but for a system with 8 GB RAM this was the best that can be managed. No pooling strategy will be used when word embeddings were to be generated and 'reduced mean' when sentence embeddings are needed. Reduced Mean pools average of all the hidden state mentioned in pooling layer along the time axis.

3.7 Classifiers

The main two classifiers being evaluated are the Convolutional Neural Network and BERT classifier along with the different embedding techniques mentioned earlier. CNN will be tested with both word2vec and BERT embeddings. Not many researchers to the author's knowledge have worked on training a separate classifier with BERT embeddings at such a high volume of data. The design of the CNN based architecture will have 3 layers of 1 dimensional convolutional layer as many researches for short texts have previously shown multi-layered 1-D convolutional layer provide good results.

The first CNN layer will have an embedding layer with an embedding matrix as a vocabulary dictionary with word2vec embeddings. For BERT embeddings however the embeddings are fed directly to the network. Fine-tuning of the network will be done empirically and given the embedding dimension for word2vec are 300,

the next CNN layer will have number of neurons to the tune of 500 approximately. A maxpooling layer will be used with pool size set at different values like [3,4,5] and with a stride of [1,2]. Relu activation function will be used for each CNN layer. Dropout layer will be added after each CNN layer for regularization purposes. A fully connected dense layer will succeed the CNN layers, followed by an output layer with softmax activation function with 2 nodes for binary classification.

3.8 Model Evaluation method

Metrics used for evaluation will be Accuracy, Precision, Recall, F1-score and Cohen's Kappa value.

Precision also known as positive predictive value is the ratio of relevant instances among the retrieved ones.

Precision = (True Positive)/(True Positive + False Positive)

While Recall also known as sensitivity is the ratio of total relevant instances that were retrieved.

Recall = (True Positive)/(True Positive + False Negative)

F1 score gives us the harmonic mean of precision and recall calculated as:

F1 score = 2 * (Precision * Recall)/ (Precision + Recall)

Cohen's kappa gives us a score which measures inter and intra rater ability for categorical values. It gives us the confidence on the results as it compares with how better the results are compared to agreement occurring by chance.

cohen's kappa = $(p_o - p_e) / (1-p_e)$

where p_o is relative observed agreement and p_e is agreement by random chance

3.9 Strength and Limitation of Approach

Strengths: The biggest takeaway from this design is the framework to show compatibility between transformers and Keras deep learning network (CNN in this case, but it can be replaced with any neural network architecture supported by Keras). Not many researches have been done to the author's knowledge that designed a framework which gives accessibility and flexibility to tune features derived from 2 different architectures. In other words, one has complete control over the process of embedding generation from the transformers and the fine-tuning of downstream network. This approach gives us the liberty to tune one parameter or hyper-parameter at a time and makes it easier to understand which component of the framework is influential. The path to the best possible combination of parameters becomes faster because of the ease of access in tuning.

Steps for the preparation of data has been kept simple, any researcher who wants to replicate this design will get up to speed to training model quickly without spending a lot of resources in pre-processing the data. Another advantage of lightweight pre-processing is that the performance of the model won't be reliant on the quality of pre-processing of text data and therefore it becomes easier to replicate the results on different bio-medical datasets as long as the text data follows basic language syntax and semantics.

The design ensures that important aspects regarding dataset and training parameter which are historically known to be influential factors in the area of embedding generation are tested like domain specific embeddings, sentence vs word vs character embeddings, encoding string as a categorical feature etc. Along with these experiments, one also wants to find out the best possible combination of hyper-parameters.

Limitations: Although the aim of the research is to find pre-trained embeddings' ability to match or better performance as compared to embeddings generated from scratch this research is considering only BERT pre-trained embeddings in detail and word2vec as a reference point. However, many pre-trained embedding techniques are available like Embeddings from Language Models (Peters et al., 2018), Universal

Sentence Encoder (Cer et al., 2018) etc. The choice to opt for BERT was driven by the fact that BERT and its variants have been able to achieve top position in GLUE leaderboard at the time when this research was conducted. This study is limited to data from bio-medical domain itself, however the design can be extended to test data from other domains as well but no guarantee over results can be made. A researcher will have to experiment and fine-tune certain aspects to get the best results. This study is specifically looking at classification task and there are no assurances over how it will perform over other NLP tasks such as NER, Question Answering etc.

4. RESULTS AND DISCUSSION

In this chapter the results of the experiments are discussed in liaise with the primary aim of the research which was to understand if embeddings generated from pre-trained models can be used as a representation of clinical texts for building a text classifier. The other objectives are supplementary to build the best possible model for the primary objective.

4.1 Model Comparison

In this section the outcomes of different modelling approaches and any significant traits are reported. In summary chapter, performance in accordance to evaluation metric is made and the best model is selected.

4.1.1 Baseline Model

The baseline model provides a base or starting point against which to compare and assess the performance of future progress. A simple model of predicting every label which has higher frequency count compared to the other variable is predicted for every case to get a baseline accuracy. Since the label is equally distributed in dependent variable the baseline accuracy for prediction is 50%.

However, the benchmark performance or the score to beat is F1-score of 0.88 achieved by CNN model with embeddings generated from scratch in the original research. 1 million datapoints were used by (Bustos & Pertusa, 2018) for this research. Although the best performance was obtained by KNN (K-Nearest Neighbour) in the original research, training 1 million or even 100k records on KNN is practically not possible with the infrastructure at disposal and it is not in the scope of this research. To conclude the baseline or benchmark score is set at 0.88 F1-score.

4.1.2 Word2Vec Pre-trained embeddings

10% or 100k records were randomly selected from the entire dataset (1 million). 80% of the dataset was reserved for training and 20% for testing. A word2vec pre-trained model trained on WIKI, news journals etc. was used to generate word embeddings for

each token after pre-processing text data as mentioned in the design section 3.3. The vocabulary count in the reduced training dataset was 16,082 which constitutes almost 50% of the total vocabulary. No minimum frequency was set on words, all words were retained. An embedding matrix was initialized that maintained words and their embeddings which were found in the vocabulary of pre-trained model. Out of the total corpus, 70% words were found in word2vec vocabulary while the rest 30% were represented by vector of zeros.

4.1.2.1 Word2Vec Embeddings with CNN as classifier

Aligning to the primary objective to determine if pre-trained embeddings can provide better representation and accuracy in text classification, clinical statement is used as text input. The embeddings are provided by pre-trained word2vec model. Model architecture can be seen in figure 4.1. The best parameters were obtained by comparing evaluation metrics between successive experiments. The average precision and recall score obtained was 0.82 and 0.83 respectively using 100k records (80k for training). Kappa's coefficient obtained was 0.71 which will be considered as substantial agreement. F1 score obtained was 0.82 which is substantially less than the benchmark score of 0.88 that needs to be improved upon.

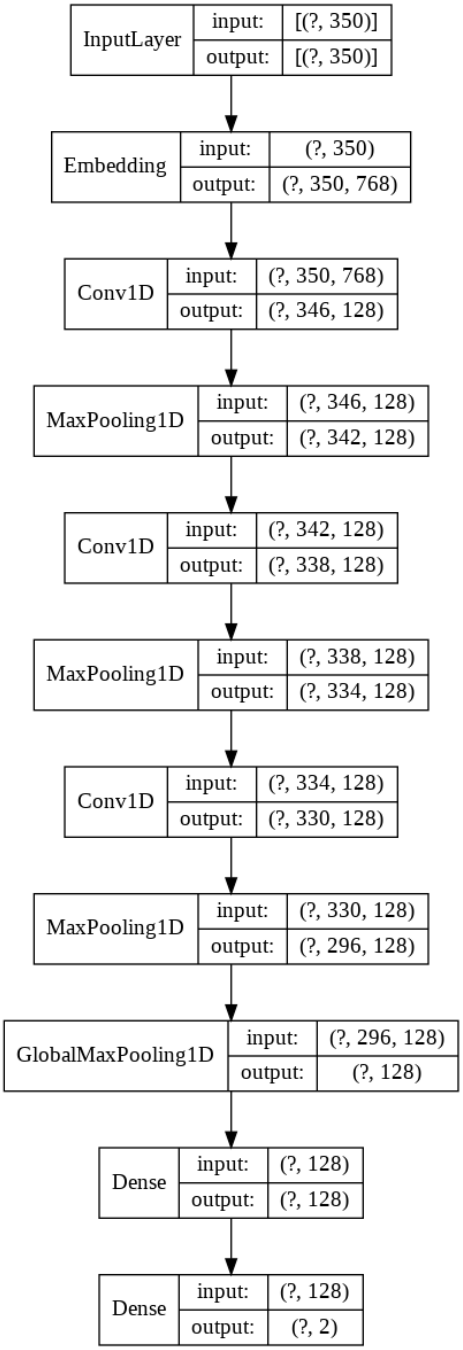

Figure 4.1 CNN Classifier

4.1.2.2 Study Intervention as categorical feature with Word2Vec-CNN

In this experiment 'Study Intervention' which is a part of clinical statement is encoded as a categorical feature to check if it can contribute more as a categorical feature than text. This experiment is aligned to the secondary objective #2. Study Intervention

combined with Study Condition forms Clinical Statement. Study Intervention is encoded as a single feature. The input comprises of Study Condition as a string and Study Intervention as a categorical feature. The structure of the network was similar to section 4.2.1 apart from an additional feature in parallel concatenated with a CNN output as shown in Figure 4.2. The results were not significantly different from results obtained in section 4.2.1 with average F1-score as 0.815. On further exploration it was found that removing the categorical feature of Study Intervention and using only Study Condition resulted in drop in performance of 4-6% in F1-score and accuracy. When a word2vec model was built using word2vec embeddings of Study Intervention only, the model accuracy was 58% which is far less than 74% accuracy obtained by using Study Condition as a standalone text input. This conforms to the belief that the patient's health is playing a far important role than the drug in consideration to decide their eligibility and was in liaise with the secondary objective #3.

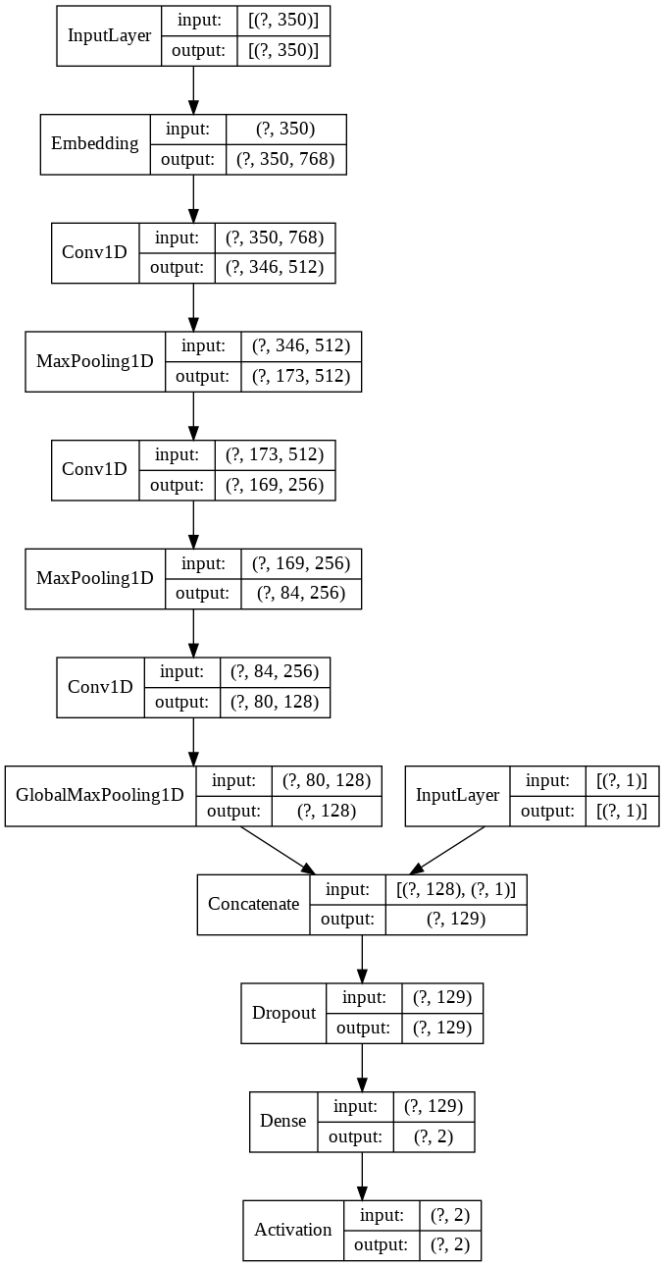

Figure 4.2 CNN classifier with Study Intervention as Categorical Feature

4.1.3 Fastext Pre-trained embeddings

In previous experiment 30% of the vocabulary didn't have any valid embeddings to represent them instead they were represented by zero vectors. Therefore, a test was conducted if having embeddings at character or n-gram level for text increased its accuracy. The total train-test samples and the split was kept exactly similar to previous experiments. But the performance dropped by 2-3% as compared to word embeddings used in word2vec model with F1-score reported as 0.80. Therefore, the result of

secondary objective #1 is that character embeddings used in fastext doesn't provide better results than word2vec.

4.1.4 BERT pre-trained embeddings

As mentioned in design chapter 3.6, two separate approaches were planned with BERT:

i) To generate BERT embeddings for clinical texts and train them on a separate network for classification

ii) To generate BERT embeddings for clinical texts and train them on BERT classifier

While the advantage with first approach was the flexibility in designing your own network. The second approach was rigid but is more efficient in utilizing resources and compatible with BERT embeddings. Both the approaches are discussed in following sections.

4.1.4.1 BERT embedding trained on CNN classifier

BERT embeddings were generated using base model producing 768 dimensional vector per word. The framework used to generate BERT embeddings was implemented according to section 3.4.2.3. Uncased version of the BERT is used to generate embeddings which is insensitive to uppercase or lowercase characters. Number of epochs were kept at 3, each epoch took around an hour to train. The batch size was kept at 16. Equal weight was given for all layers while pooling the embedding. Embeddings were drawn from the 12th layer. Out of vocabulary words which get broken into sub-tokens are embedded as sub-tokens and finally single vector was generated by averaging embeddings of n-grams, the other method tested was by adding sub-token embeddings. These embeddings are then trained on downstream CNN network architecture as defined in the section 3.6. The weights and biases were tuned for the downstream classifier network, while the pre-trained weights and embeddings of the BERT layers were untouched. Performance improved for the model with loss decreasing to 0.38, precision and recall increased to 0.86 and 0.87 respectively. F1-score reported was 0.87 and accuracy of the model was 88%. Cohen's kappa was 0.76 indicating substantial agreement. This experiment was in alignment with the primary

objective to determine if pre-trained embeddings can perform better than embeddings generated from scratch. However, the F1-score is still short of the benchmark accuracy.

4.1.4.2 BERT embedding trained on internal BERT classifier

Building on previous approach where only parameters of the classifier's were trained, in this experiment the pre-trained weights of the embedding generator are allowed to be updated as well. This method tunes the embedding generation in accordance to classifying task. F1-score obtained was 0.91 with precision and recall in the same range. The final accuracy was 91% seen in the figure 4.4. Cohen's Kappa was 0.81 indicating almost perfect agreement. Loss was very low just after 1st epoch as can be seen in the figure 4.3 with loss after 3rd epoch was reported as 0.25, which took anywhere between 50 minutes to 1 hour for one epoch to train on 80k records. The batch size impacted the training time as expected, with higher batch size the training time reduced. When the training size was increased from 80,000 to 400,000 F1-score obtained was 92.5 with both precision and recall score as 0.92. Because of the computational cost, training couldn't be allowed to run beyond 3 epochs which took more than 6 hours to finish. Noticeable improvements were observed in loss and accuracy metrics, indicating more improvement if further optimization could be achieved in the training process and the number of epochs could be increased. The loss curve showed slight decreasing trend even after 3rd epoch but the validation loss started increasing.

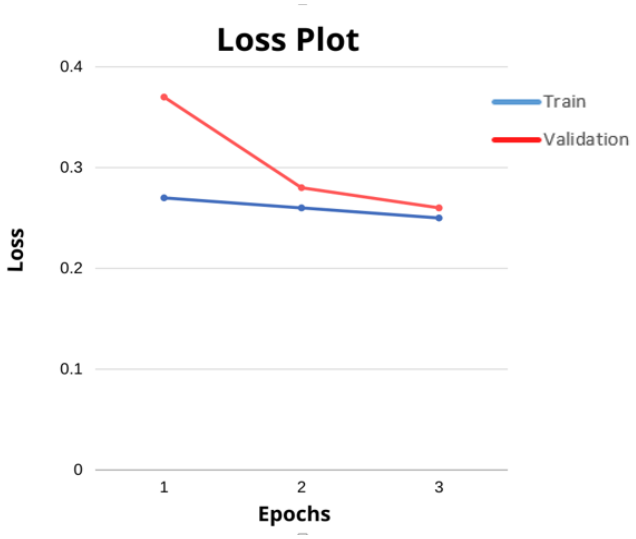

Figure 4.3 Loss plot for BERT embedding and Classifier

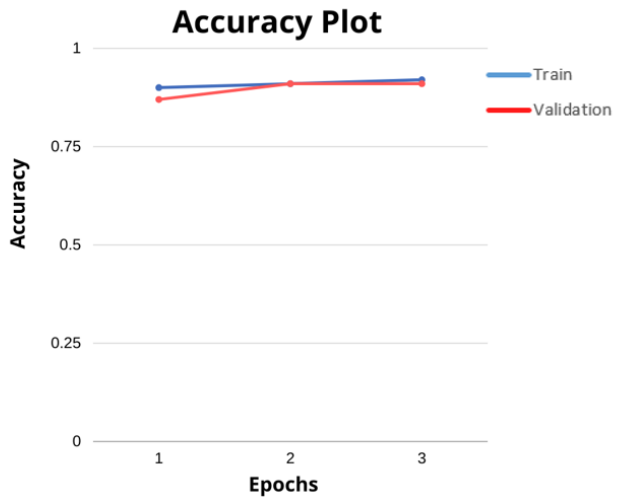

Figure 4.4 Accuracy plot for BERT embedding and classifier

4.1.5 Domain Specific Embeddings

When a language model is trained on a corpus which belongs to a specialised field of work, the model has learnt key traits which is very specific to that particular field. Such models are called as domain trained model and the embeddings generated by such models will be domain specific.

4.1.5.1 Domain specific embedding trained on CNN classifier

Extending the framework used for section 4.1.4.1, bio-medically fine-tuned BERT model called as BioBert (Wada et al., 2020) is employed to understand if bio-medical fine-tuned BERT embedding can help improve the results. Since BioBert is initialized

on BERT pre-trained weights and fine-tuned on biomedical text mining tasks, the training parameters were kept as same as BERT. The idea was to understand if embeddings generated using a bio-medical fine-tuned model will improve results for in classification task, this is in accordance to secondary objective #6. Unfortunately, there was no significant improvement seen as compared to earlier results with BERT base, negating the theory of performance improvement by using domain specific embeddings.

4.1.5.2 Domain specific embedding trained on BERT classifier

Similarly, BERT classifier is used with BioBERT embeddings, no improvement are seen in final results as compared to the best model. F1-score reported was 0.90. One of the challenges in bio-medical dataset is the interpretations of domain specific terminologies which BioBert is equally unequipped with as they use the same vocabulary dictionary as BERT. The reason for them to not upgrade the vocabulary is the risk to compromise the pre-trained weights and other parameters of BERT base model. Instead, they fine-tune the weights of the model in accordance with the original vocabulary which also includes the sub-tokens fine-tuned in accordance to bio-medical corpus. But it fails to improve performance above BERT classifier, and secondary objective #6 can be concluded that classifier accuracy doesn't improve by using domain specific language model. However, this should not discourage using domain trained BERT models for other NLP tasks such as Named Entity Recognition, Relation Extraction and Question answering etc.

4.1.6 Sentence Embeddings Vs Word Embeddings in BERT

Every sentence in BERT starts with a [CLS] token, [CLS] stands for classifier. It was originally trained on 'Next Sentence Prediction' task using [CLS] token as an approximate representation for the entire sequence. To produce sentence representation, vector representation of entire [CLS] token is used and its classification accuracy is compared with normal word embeddings. This experiment is designed to test the aim of the secondary objective #4. This was tested with sample size of 10,000 for both type of embeddings (sentence and word) and found that sentence embeddings gave slightly better average F1-score of 0.85 whereas word embeddings gave an average F1-score of 0.83. This is in compliance with the previous approach on BERT

pre-trained model and classifier in section 4.1.4.2 which used sentence embedding as well to form representation of the sequences. Another way to extract sentence embeddings is by pooling average of the embeddings generated from the last layers instead of using [CLS] token. Comparison between both the methods to extract sentence embeddings

i) [CLS] token as sentence embedding and

ii) Average pooling of embedding from hidden layers

didn't show any significant differences.

4.2 Discussion and Conclusion on Results

The aim behind the varied experiments was not only to find best performing setup for classification but also to understand which aspect of pre-trained embeddings can have an impact on the classification accuracy of a dataset. A peculiar dataset as it has plenty of domain specific terminology. A benchmark score was assumed in the original research done on Clinical Trials (Bustos & Pertusa, 2018).

The experiment started out with a simple pre-trained model 'word2vec' which generates word embedding for every word and the classifier used was CNN with 3 hidden layers. The performance of word2vec was going to be a reference score for all BERT pre-trained embedding models. The average F1-score reported with word2vec was 82% way below the benchmark. Before proceeding to BERT embeddings it was important to understand if encoding study interventions/drug name as a categorical variable will in any way boost to the results. However, no improvement was found and study intervention was used as raw string in future experiments.

With pre-trained embeddings one major challenge was the shortage of embeddings found for words which are not in the vocabulary of the model. To identify if proportion of words which are out of vocabulary are going to play a huge role in the representation of the sentences, **fastext** was used. Word2vec variants like fastext uses character embeddings and can have representation for unseen words too. The precision, recall score obtained using fastext were 79% and 80% respectively which is

3-4% less than word2vec word embeddings. CNN used as a classifier with Word2Vec and BERT embeddings saw improvement in performance by 5% to get F1-score as 0.87 with BERT embeddings. This result is still not an improvement over the benchmark score.

BERT embeddings with BERT classifier achieved the best performance in this research with 0.91 F1-score. T-test is performed to ensure if the improvement is significant enough. Ten readings of F1-score was noted. The results of t-test was reported as below:

*An independent-samples **t-test** was conducted between F1 scores of embeddings generated from scratch and pre-trained embeddings. **Significant difference** in the F1-score was found (M=0.88, SD= 0.00016 for embeddings from scratch, M= 0.91, SD= 0.000238 for pre-trained embeddings), (t(10)= -7.98, p <= 0.01).*

The improvement in the results by pre-trained embeddings is more significant as it uses only $1/10^{th}$ of the data for training compared to what the original research used.

The next experiments are concerned with different nuances used in text embedding and classification this study wants to test out in pursuit of understanding text embeddings and classification. It tries to investigate if bio-medical domain specific embeddings can improve upon the results obtained using generic English embeddings. However, no improvement is seen in the performance and BERT generic embeddings will be used for future experiments. The final test performed was to understand if sentence embedding can provide better representation of the text sequence instead of contextual word embeddings. It is found that using sentence embeddings provide minor improvements but significant reduction in vector size.

The summary of all significant results are listed in table below with the best performing model highlighted.

Summary of important results				
Description	**Embeddings Type**	**Embeddings Generation method**	**Classifier**	**F1-Score**
Benchmark Score - Word2vec embeddings generated from dataset and CNN was used as a classifier	Embeddings generated from dataset	word2vec	CNN	0.88
Word2Vec with CNN - Embeddings generated from pre-trained word2vec model and CNN was used as a classifier	Pre-trained	word2vec	CNN	0.82
BERT with CNN - Embeddings generated from pre-trained BERT model and CNN was used as a classifier	Pre-trained	BERT	CNN	0.87
BERT embedding with BERT classifier - Embeddings generated from pre-trained BERT model and BERT internal classifier was used	Pre-trained	BERT	BERT	0.91
BioBERT with CNN - Embeddings generated from fine-tuned BioBERT model and CNN was used as a classifier	Pre-trained	BioBERT	CNN	0.87
BioBERT with BERT- Embeddings generated from fine-tuned BioBERT model and BERT internal classifier used as a classifier	Pre-trained	BioBERT	BERT	0.89

Table 4.1 Summary of important results

Below table shows the analysis/ conclusion of all objectives listed before.

Objective Summary		
Objective	**Description**	**Analysis/Conclusion**
Primary Objective	To determine if pre-trained embeddings, fine-tuned on the dataset can perform as well as or better than the embeddings generated from scratch. This will help researchers to train models with pre-trained embeddings as representation for the text instead of generating them from scratch.	Pre-trained embeddings performed significantly better than benchmark results
Secondary Objective #1	To determine if character or n-gram embeddings used by fastext are better representation than word embeddings generated from word2vec.	Fastext embedding didn't perform better than word embeddings
Secondary Objective #2	To determine if drug names in study prescription encoded as categorical feature enhances the accuracy of the model. This will also help us evaluate if the name of the drug is influencing the eligibility rate in clinical trial.	Encoding study intervention as categorical feature didn't improve results
Secondary Objective #3	To determine if study conditions are more important than study intervention in determining the eligibility criteria of a patient. In other words a patient's health is more crucial to determine their eligibility than the drug whose trial is carried out.	Study conditions were more important than study interventions
Secondary Objective #4	To determine if sentence embeddings improve accuracy of a text classifier as compared to word embeddings. Sentence embedding is a technique used to represent a sequence in an n-dimensional vector space. Unlike word embedding we don't have a representation for each word, instead we will have a single representation for the entire sentence. We compare accuracy obtained from both sentence and word embedding to understand if there is a significant difference between the two techniques.	Sentence embeddings improved accuracy as compared to word embeddings
Secondary Objective #5	To determine if classifiers built within the pre-trained framework give more accuracy as compared to independent neural network.	In-built classifiers gave better results as compared to independent neural network
Secondary Objective #6	To determine if embeddings fine-tuned on bio-medical data gives more accuracy as compared to generic embeddings.	Domain specific embedding didn't improve accuracy as compared to generic embeddings

Table 4.2 Objective Summary

4.3 Strengths and Weaknesses of Findings

Strengths: The primary benefit of the results is the evidence that pre-trained embeddings can be a good alternative for researchers instead of designing their own embeddings. The metrics used for analysing the results are straightforward and widely acceptable in the field of classification, which establishes the good quality of the model. The model has been able to achieve better results than the original research in $1/10^{th}$ of the sample size. This research can be used as a proof for using pre-trained embedding as a representation of bio-medical texts even when the sample size of the dataset is not big. Using pre-trained embedding will also give head start to a researcher and focus more on quintessence of classification.

The secondary advantage is exploring which factors may or may not have had influence over the results that were aligned to the secondary objectives. These results can be extrapolated while doing research on clinical trials and start experimenting with only those techniques which brought improvement in the results.

Weakness: Although this implementation has been able to give flexibility in adjusting the operation of components working in a transformer, it doesn't have support to interpret embeddings. This research has been an effort to shift away from black box approaches, which BERT is normally associated with to something which gives more interpretability in the functionality of the transformer. There is not enough understanding over which phrase or token has had an impact on predicting the label. The model can be reused for binary task but will need to be fine-tuned even for bio-medical data. For tasks which have more than 2 labels will need to attach a separate layer at the end specifying number of labels. The model has been tested only for balance datasets, the results will suffer in cases where class imbalance is heavily prevalent.

5. CONCLUSION

This chapter summarizes the research and highlights few key aspects uncovered. It also tries to build a roadmap for the next steps on the work that remains to be done in the field of text embeddings and classification.

5.1 Research Overview

This thesis started with an aim to leverage existing state-of-the-art architecture for generating embeddings instead of reinventing the wheel for text representation. The dataset in question was from bio-medical background containing information of from clinical trials conducted over the years. A research was performed in 2018 on the same dataset with word2vec and fastext embeddings, both being non-contextual approaches which means the embeddings generated for each word won't be considering the impact of neighbouring words while transforming the words into numerical vectors. The embeddings used in the research was generated from the dataset. Now to form good representation for any language, substantial amount of corpus is required which is not feasible in every approach, but this particular dataset had access to it. Therefore, the researchers were able to attain excellent results while performing the classification. The approach this research demanded needed to be equally good or better to be in reckoning for future researches and therefore this became the success criteria for the research.

Since the research is more focussed on the quality of text representation instead of the classification task, more emphasis on methods to generate embeddings that will serve as the best representation of the clinical text. One of the reason why network architecture for classifier remains fairly constant other than the adjustment of parameters in hidden layers in accordance with the dimensions of the embedding output generated. Classification accuracy will serve as a method to judge embeddings, but the aim was not to build the best classifier.

After performing literature review on different language models which come with pre-trained weights BERT was zeroed to produce representation for the text as it has outperformed several models in various NLP tasks. There were a host of classifiers

which have done exceptionally well in text classification, eventually CNN classifier was chosen as it is known for impressive performances in text classification. Several gaps were identified in the existing research such as polysemy, co-reference issues, lexical and multi-sential ambiguity that could be resolved using BERT as per previous researches (Wiedemann, Remus, Chawla, & Biemann, 2019).

While formulating the design for the research different methods to generate BERT embeddings were discovered. Embedding generation techniques like BERT as a service and sklearn's BERT API provided ease of use, they came with several computational limitations. This led the hunt for different method to generate embeddings dynamically and process them on the classifier in real time. Classification network was designed using huggingface transformer API which serves as the hallmark of the research because of the flexibility and compatibility it provides with external frameworks. Taking advantage of the flexibility this framework provides that could optimize model training process by dropping unnecessary overhead using techniques like mixed precision, dynamic padding and uniform length batching. Parameter tuning was done successively and the combination with the best possible result was chosen.

Model which scored best average F1-score and kappa score will be chosen as the best model.

5.2 Problem Definition

The research problem can be divided in two parts, one which concerns with data, while other being methodological.

Data specific problem - Can a text classifier be built which will determine the eligibility of a patient for a clinical trial, given his health conditions and the drug in consideration. Clinical trials is a restrictive process where many protocols need to be followed to determine the eligibility of a patient. This is an arduous process where the possibility of mistakes are high. The scale of such trials is such that many personnel are involved in conducting these trials and therefore maintaining consistency in such process becomes difficult. Compounding on this problem is the fact that when these

drugs are released in the market, a clinical practitioner should be aware of the profiles which were excluded or included during clinical trials if he/she wants to reproduce the results of clinical trial. This involves a lot of overhead which takes time to process and could prevent a patient from receiving the benefit of the drug in consideration. Having a model which will determine a patient's eligibility for the drug will make it easy for the doctor to administer the drug with confidence. In case of future clinical trials the same model can be reused for different study/subjects. The model can represent a guideline of rules internally which will eventually decide if a patient with certain health condition can or cannot undertake the treatment.

Methodological problem - In a text classification problem the most challenging aspect for a researcher is to translate the text into machine understandable language. Normally this is achieved by numeric representation of text called as embeddings. In many researchers designing these embeddings on their own based on the understanding they have about the problem at large. However, it is not always possible to have such resources at disposal to encode quality representation of the text. To summarize is the idea of generic model to that is capable to encode these medical text into vector space which can be then used perform text classification that will achieve same or better quality of performance compared to embeddings generated from dataset?

5.3 Experimentation and Results

The evaluation metric used were same as the original research for ease of comparison. Previous research on the dataset which generated embeddings from the dataset using word2vec and CNN as classifier achieved F1-score of 0.878. This was set as the benchmark score to beat. The entire research is focussed on evaluating pre-trained embeddings generated from word2vec and BERT. While embeddings were generated using variants of BERT, the classifiers were divided broadly into two categories. First was the internal classifiers which are coupled directly with a language model like BERT, while the other was a separate custom CNN architecture that will consume BERT embeddings dynamically and train on the classifier in batch fashion. While the BERT-CNN classifier did come close to benchmark, scoring 0.87 in F1-score. An all

BERT embedding and classifier outperformed benchmark score by 3% scoring 0.91 in F1-score and accuracy. The performance achieved became more significant coupled by the fact that the score was achieved in one-tenth of the dataset used in original research.

As part of the secondary objective some other important aspects of text embedding and classification were investigated. A comparative test between word and sentence embedding was carried out by measuring their F1-score and accuracy. It was found that sentence embeddings are superior to word embedding by 1-2%. Pre-trained models was used for generating embeddings and they are trained on generic English, whereas the dataset contains domain specific terms which are unseen to the model during their pre-training. To overcome this problem BioBERT was used which is a BERT model fine-tuned on bio-medical corpus to see if any improvement was found. Only to find later that no significant improvement was achieved using BioBERT.

While the dataset comprised of two components the drug in consideration and the patient's health condition, they were tested in isolation to understand their importance and it was found that major contribution to the classification was made by patient's health condition. The drug in consideration didn't impact the final output significantly underlining the fact that there is a certain degree of uniformity when it comes to patient's health requirements to participate in clinical trials.

5.4 Contributions and Impact

The research has been able to show that leveraging pre-trained models for NLP tasks can not only match the benchmark performance but improve the accuracy. This method replaces the strenuous task of designing embeddings every time there are modification to the dataset. Once the research is finished these customized embeddings can't be reused because of its coupling which is very specific to the dataset.

The biggest contribution of this research is the framework which has the compatibility to use transformers with custom neural network. This gives independence to the researcher to focus on one task at a time. No matter the embedding technique used if the shapes of the input are maintained correctly, the

classifier can be designed in any fashion. Influence of each parameter can be observed by fine-tuning one parameter at a time. The framework provides support to optimization technique like mixed precision which will adjust the size of the vectors to be used while generating embeddings to maintain a balance between quality of embedding and performance. The AMP (Automatic Mixed Precision) technique will decide on the fly how to switch between 32 bit or 16 bit floating point to maintain quality and improve training performance. Dynamic padding is another technique to reduce the padding of zeros in a sequence. According to previous researches it increases speed by 3x in GPU and by 60% in TPU.

In future if one desires to modify any aspect of the embedding generation or classifier can easily to do so because of the accessibility to fine-tune or change components in the framework.

5.5 Future Work & Recommendations

The immediate next steps in future work is to test new dataset from bio-medical domain on the framework. This will help test the robustness of the model. Once the confidence on the embedding generation is established, the framework should be extended to other NLP tasks such as NER etc. There are a variety of options available among pre-trained embeddings models and it would be interesting to compare performance from other pre-trained language models like ELMO (Embeddings from Language Models), USE (Universal Sentence Encoder) etc. The flexibility of the model is to evaluate any new pre-trained embedding model a researcher just has to supply the name of the new model. The framework will take care of the entire process, right from downloading the model to generate embeddings and everything will be in accordance with the parameters, optimization techniques, analysis metrics the user has opted.

In this research only CNN classifier is tested but other Recurrent Neural Network techniques like LSTM or RNN are known to be popular techniques in NLP classification tasks which can be evaluated. The combination of CNN-LSTM has worked well in text classification tasks where long range dependencies and local

variations are important and can be an interesting alternative to a standard CNN classifier used in this research.

Recommendations

A separate study on word entity relation where every drug's association with top 'n' common health symptoms by occurrence can be showcased. This will highlight the top n health conditions which are eligible or ineligible for a particular drug. Research on words or phrases which drive the final output can add substantial value to the model. There are some visualizing techniques like t-SNE (t-Distributed Stochastic Neighbor Embedding) that can help us project words in a reduced space and understand the noun phrases found more in ineligible or eligible criteria. It can also be used in clustering word embeddings to analyse which diseases or conditions are closely related compared to others. This will also form a part of empirical evaluation of the quality of the embeddings.

6. REFERENCES

Aronson, A. R. (2001). Effective mapping of biomedical text to the UMLS Metathesaurus: The MetaMap program. *Proceedings of the AMIA Symposium*, 17. American Medical Informatics Association.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *ArXiv Preprint ArXiv:1508.05326*.

Bassiou, N. K., & Kotropoulos, C. L. (2014). Online PLSA: Batch Updating Techniques Including Out-of-Vocabulary Words. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(11), 1953–1966.

Baayen, H. (1992). Statistical models for word frequency distributions: A linguistic evaluation. *Computers and the Humanities*, *26*(5), 347–363. https://doi.org/10.1007/BF00136980

Baillargeon, J.-T., Lamontagne, L., & Marceau, É. (2019). Weighting Words Using Bi-Normal Separation for Text Classification Tasks with Multiple Classes. *Canadian Conference on Artificial Intelligence*, 433–439. Springer.

Bustos, A., & Pertusa, A. (2018). Learning Eligibility in Cancer Clinical Trials Using Deep Neural Networks. *Applied Sciences*, *8*(7), 1206. https://doi.org/10.3390/app8071206

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., … Kurzweil, R. (2018). Universal Sentence Encoder. *ArXiv:1803.11175 [Cs]*. Retrieved from http://arxiv.org/abs/1803.11175

Che, Z., Cheng, Y., Sun, Z., & Liu, Y. (2017). Exploiting Convolutional Neural Network for Risk Prediction with Medical Feature Embedding. *ArXiv:1701.07474 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1701.07474

Dostal, M., & Ježek, K. (n.d.). *AUTOMATIC KEYPHRASE EXTRACTION BASED ON NLP AND STATISTICAL METHODS*. 2.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv:1810.04805 [Cs]*. Retrieved from http://arxiv.org/abs/1810.04805

Gonçalves, T., & Quaresma, P. (2004). The impact of nlp techniques in the multilabel text classification problem. In *Intelligent Information Processing and Web Mining* (pp. 424–428). Springer.

Heneghan, C., Goldacre, B., & Mahtani, K. R. (2017). Why clinical trial outcomes fail to translate into benefits for patients. *Trials*, *18*(1), 122.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107–116.

Johnson, R., & Zhang, T. (2015). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *ArXiv:1412.1058 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1412.1058

Johnson, R., & Zhang, T. (2016). Convolutional Neural Networks for Text Categorization: Shallow Word-level vs. Deep Character-level. *ArXiv:1609.00718 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1609.00718

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). FastText.zip: Compressing text classification models. *ArXiv:1612.03651 [Cs]*. Retrieved from http://arxiv.org/abs/1612.03651

Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. IEEE Transactions on Systems, Man, and Cybernetics, (4), 580–585.

Ling, Y., An, Y., Liu, M., Hasan, S. A., Fan, Y., & Hu, X. (2017). Integrating extra knowledge into word embedding models for biomedical NLP tasks. *2017 International Joint Conference on Neural Networks (IJCNN)*, 968–975. https://doi.org/10.1109/IJCNN.2017.7965957

Liao, M., Shi, B., Bai, X., Wang, X., & Liu, W. (2016). TextBoxes: A Fast Text Detector with a Single Deep Neural Network. *ArXiv:1611.06779 [Cs]*. Retrieved from http://arxiv.org/abs/1611.06779

Lee, K., Salant, S., Kwiatkowksi, T., Parikh, A., & Das, D. (n.d.). *LEARNING RECURRENT SPAN REPRESENTATIONS FOR EXTRACTIVE QUESTION ANSWERING*. 9.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. Twenty-Ninth AAAI Conference on Artificial Intelligence.

Le Grand, S., Götz, A. W., & Walker, R. C. (2013). SPFP: Speed without compromise—A mixed precision model for GPU accelerated molecular dynamics simulations. *Computer Physics Communications*, *184*(2), 374–380.

Milian, K., Hoekstra, R., Bucur, A., Teije, A. ten, Harmelen, F. van, & Paulissen, J. (2015). Enhancing reuse of structured eligibility criteria and supporting their relaxation. *Journal of Biomedical Informatics*, *56*, 205–219. https://doi.org/10.1016/j.jbi.2015.05.005

Milian, K., Ten Teije, A., Bucur, A., & Van Harmelen, F. (2011). Patterns of clinical trial eligibility criteria. *International Workshop on Knowledge Representation for Health Care*, 145–157. Springer.

Mikolov, T., & Zweig, G. (2012). Context dependent recurrent neural network language model. 2012 IEEE Spoken Language Technology Workshop (SLT), 234–239. IEEE.

Moradshahi, M., Palangi, H., Lam, M. S., Smolensky, P., & Gao, J. (2019). HUBERT Untangles BERT to Improve Transfer across NLP Tasks. *ArXiv:1910.12647 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1910.12647

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., … Wu, H. (2018). Mixed Precision Training. *ArXiv:1710.03740 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1710.03740

Ng, V., & Cardie, C. (2001). Improving machine learning approaches to coreference resolution. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 104. Philadelphia, Pennsylvania: Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073102

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *ArXiv:1802.05365 [Cs]*. Retrieved from http://arxiv.org/abs/1802.05365

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (n.d.). *Language Models are Unsupervised Multitask Learners*. 24.

Rogati, M., & Yang, Y. (2002). High-performing feature selection for text classification. *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, 659–661.

Ramos, J. (n.d.). *Using TF-IDF to Determine Word Relevance in Document Queries*. 4.

Soucy, P., & Mineau, G. W. (2001). A simple KNN algorithm for text categorization. *Proceedings 2001 IEEE International Conference on Data Mining*, 647–648. https://doi.org/10.1109/ICDM.2001.989592

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 5998–6008.

Wiedemann, G., Remus, S., Chawla, A., & Biemann, C. (2019). Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. *ArXiv:1909.10430 [Cs]*. Retrieved from http://arxiv.org/abs/1909.10430

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ArXiv:1804.07461 [Cs]*. Retrieved from http://arxiv.org/abs/1804.07461

Wada, S., Takeda, T., Manabe, S., Konishi, S., Kamohara, J., & Matsumura, Y. (2020.). *A pre-training technique to localize medical BERT and enhance BioBERT*. 6.

Wang, J., Wang, Z., Zhang, D., & Yan, J. (2017). Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2915–2921. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2017/406

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1–4), 43–52.

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1–4), 43–52.

Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A C-LSTM neural network for text classification. *ArXiv Preprint ArXiv:1511.08630*.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *ArXiv Preprint ArXiv:1611.06639*.

7. APPENDIX A

```
#Sentence embedding
!nohup bert-serving-start -model_dir=./uncased_L-12_H-768_A-12 -max_seq_len=150  > out.file 2>&1 &

# Word embedding
!nohup bert-serving-start -model_dir=./uncased_L-12_H-768_A-12 -max_seq_len=150 -pooling_strategy=NONE > out.file 2>&1 &
```

Figure 7.1 BERT as a Service in CLI format

```
class PoolPolicy():
  def __init__(self, pooling_layer_number, policy_dict):
    self.pooling_layer_number = pooling_layer_number    # int (Eg: 11, 11 is the second last layer for a 12 layered bert_base, range of values [1, max(layer_number)]) ; Default=-1 (final hidden state)
    self.policy_dict = policy_dict                      # dict (Eg: {'base_expression': "10 + 11 + 12", 'norm_op': ('avg', [weights])}); Default = None;
                                                        # 'norm_op' can be (avg|sum); when doing sum we don't normalize;
                                                        # for weighted average supply respective weights;
                                                        # Supply equal weights as [1, 1,...] when doing simple avg or sum
    self.supported_ops = {                              # Predefined operations that base expression can support
      "+": torch.add,
      "-": torch.sub,
      "/": torch.true_divide,
      "*": torch.mul
    }
    self.policy_evaluator = 'iterative'                 # str (Eg: any one from ['iterative', 'recursive'])

  def pool(self, hidden_states_tuple):
    word_embed_matrix_tensor = None

    if not self.policy_dict is None: # if policy dict was passed we use it instead of pooling_layer_number
      # Extract and split base expression to make it parseable
      base_expression = self.policy_dict['base_expression'].strip().split()
      # Extract custom policy dictionary, contains avg
      norm_op = self.policy_dict['norm_op']

      # First evaluate base expression with respective weights
      if self.policy_evaluator == 'iterative':
        word_embed_matrix_tensor = self.iterative_eval_base_op(mat_tuple=hidden_states_tuple,
                                                               expression=base_expression,
                                                               weights=norm_op[1])
      else: # Recursive base expression evaluator
        word_embed_matrix_tensor = self.recursive_eval_base_op(mat_tuple=hidden_states_tuple,
                                                               expression=base_expression[::-1], # Recursive works inside out(R->L), but we want left to right; so we flip the list
                                                               weights=norm_op[1][::-1]) # Recursive works inside out(R->L), but we want left to right; so we flip the list
      # Normalize matrix when 'avg'(averaging) with the supplied weights
      if norm_op[0] == 'avg': # not needed while summing
        # Average the word_embed_matrix_tensor
        word_embed_matrix_tensor = word_embed_matrix_tensor / sum(norm_op[1])

    else:
      # if pooling_layer_number < 0 it means we want to select final layer states
      if self.pooling_layer_number < 0:
        self.pooling_layer_number = len(hidden_states_tuple)
      # Choose hidden state layer specified by 'pooling_layer_number'
      # Also discard state vectors for [CLS] and [SEP] tokens
      word_embed_matrix_tensor = hidden_states_tuple[self.pooling_layer_number-1][0]

    # Return pooled word_embed_matrix_tensor
    #print(f'Pooled tensor shape: {word_embed_matrix_tensor.shape}')
    return word_embed_matrix_tensor.squeeze(0)


  # Iterative base policy evaluator
  def iterative_eval_base_op(self, mat_tuple, expression, weights):
    eval_value = None
    weight_ctr = 0
    for index in range(1, len(expression)-1):
      if eval_value is None:
        eval_value = mat_tuple[int(expression[0])-1] * weights[weight_ctr]
      expression_element = expression[index]
      if not expression_element.isdigit():
        if expression_element in self.supported_ops.keys():
          weight_ctr += 1
          eval_value = self.supported_ops[expression_element](eval_value,
                                                              mat_tuple[int(expression[index+1])-1] * weights[weight_ctr])
    return eval_value

```

Figure 7.2 Customizable Pooling Policy used in Transformer

```python
class EmbeddingDataLoader(Sequence):

  def __init__(self, tokenizer, model, X, batch_size, max_length, sampler='sequential',
               y=None, num_classes=None, get_one_hot_label=None,
               model_utilizing_gpu=False, data_on_gpu=False,
               pooling_layer_number=-1, policy_dict=None, oov='avg',
               random_seed=None):

    self.tokenizer = tokenizer                          # AutoTokenizer (Eg: BertTokenizer)
    self.model = model                                  # AutoModel (Eg: BertModel)
    self.X = X                                          # nd.array
    self.batch_size = batch_size                        # int
    self.max_length = max_length                        # int
    self.sampler = sampler                              # str (Eg: any one from ['random', 'sequential'])

    self.y = y                                          # nd.array
    self.num_classes = num_classes                      # int
    self.get_one_hot_label = get_one_hot_label          # bool

    self.model_utilizing_gpu = model_utilizing_gpu      # bool
    self.data_on_gpu = data_on_gpu                      # bool

    self.random_seed = random_seed

    self.oov = oov                                      # str (Eg: any one from ['avg', 'sum', 'last'])
    self.pool_policy = PoolPolicy(pooling_layer_number, policy_dict)
    self.organize_data()

    self.ctrx = -1

    print(f'Number of batches: {self.__len__()}')

  def __len__(self):
    """
    (Eg: num_data_samples/batch_size = 230/16 = floor(14.375) = 14)
    :return: Return total number of batches in each Epoch
    """
    return int(np.floor(self.X.shape[0] / self.batch_size))

  def test_loader_get_me_next(self):
    self.ctrx += 1
    print(f'Batch No: {self.ctrx}')
    return self.__getitem__(self.ctrx)

  def __getitem__(self, index):
    """
    Generates a batch

    :param index: index of batch
    :return: (X, y) batch when training and y when testing
    """
    #print(f'Batch No: {index}')
    X_batch = self.X[index*self.batch_size : (index+1)*self.batch_size]

    # Encode and Embed the batch(each sequence in the batch)
    embedded_X_batch_array = self.batch_encode_embed(X_batch)

    # [Discard] some tensors
    X_batch = None

    if self.y is not None:
      # Also batch up Y
      y_batch = self.y[index*self.batch_size : (index+1)*self.batch_size]
      # Also generate one hot matrix for Y, if requested # num_classes required to be set
      if self.get_one_hot_label:
        y_batch = to_categorical(y=y_batch, num_classes=self.num_classes)
      else:
        y_batch = y_batch#[:, None]
```

Figure 7.3 Data Loader used for dynamic batching and training