Heaven’s Light is Our Guide



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Rajshahi University of Engineering & Technology, Bangladesh**

Duplicate Question Pairs Detection

Author

Md. Moshiur Rahman

Roll No. 1503113

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Supervised by

Biprodip Pal

Assistant Professor

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

ACKNOWLEDGEMENT

At first, we would like to thank the Almighty Allah for giving us the opportunity and enthusiasm along the way for the completion of our thesis work.

We would like to express our sincere appreciation, gratitude, and respect to our Biprodip Pal, Assistant Professor of Department of Computer Science and Engineering, Rajshahi University of Engineering and Technology, Rajshahi. Throughout the year he has not only given us technical guidelines, advice and necessary documents to complete the work he has also given us continuous encouragement, advice, helps and sympathetic co-operation whenever he deemed necessary. His continuous support was the most successful tool that helped us to achieve our result. Whenever we were stuck in any complex problems or situation he was there for us at any time of the day. Without his sincere care, this work not has been materialized in the final form that it is now at the present.

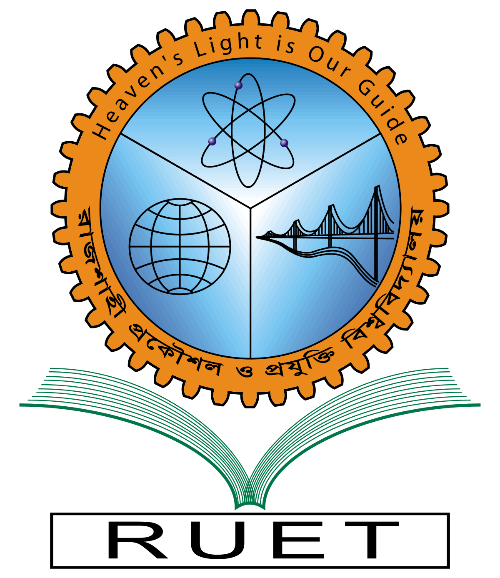
I am also grateful to all the respective teachers of Computer Science and Engineering, Rajshahi University of Engineering and Technology, Rajshahi for good & valuable suggestions and inspirations from time to time.

Finally, I convey my thanks to my parents, friends, and well-wishers for their constant inspirations and many helpful aids throughout this work.

Date: 15/12/2020 Md. Moshiur Rahman

RUET, Rajshahi

Heaven’s Light is Our Guide



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Rajshahi University of Engineering & Technology, Bangladesh

*CERTIFICATE*

*This is to certify that this thesis report entitled* ***“Duplicate Question Pairs Detection”*** *submitted by* ***Md. Moshiur Rahman, Roll: 1503113*** *in partial fulfillment of**the requirement for the award of the degree of Bachelor of Science in Computer Science & Engineering of Rajshahi University of Engineering & Technology, Bangladesh is a record of the candidate own work carried out by him under my supervision. This thesis has not been submitted for the award of any other degree*

Supervisor External Examiner

------------------------------ -----------------------------------

**Biprodip Pal**

Assistant Professor

Department of Computer Science & Engineering

Rajshahi University of Engineering

&Technology

Rajshahi-6204

**ABSTRACT**

Detection of duplicate questions from a corpus containing a pair of questions deals with identifying whether two questions in the pair convey the same meaning or not. Quora is a growing platform comprising a user generated collection of questions and answers. The questions and answers are created, edited, and organized by the users. Enormous number of users on the Quora website makes it unavoidable to have multiple questions from different users with similar intent, which raises the issue of duplicate questions. Effectively detecting duplicate questions would make it easier to find high quality answers and help save time, which in turn would result in an improved user experience for writers and readers on Quora. In this paper, Quora Question Pairs dataset is collected from Kaggle for detection of duplicate questions. To detect duplicate question pairs, we have implemented several methods and algorithms. The implemented methods are the Siamese Manhattan LSTM model with different types of embedding (Keras embedding, Google News Vector embedding, FastText Common Crawl embedding, and GloVe Common Crawl embedding), the BERT Bi-LSTM model, the Siamese BERT model, and the RoBERTa Bi-LSTM model. Our best performer model is the RoBERTa Bi-LSTM model. We achieved up to 91.20% accuracy to detect duplicate question pairs.

CONTENTS

Page no

|  |  |  |
| --- | --- | --- |
| ACKNOWLEDGEMENT | | i |
| CERTIFICATE | | ii |
| ABSTRACT | | iii |
| **CHAPTER 1** | |  |
| **Introduction** | | 1-5 |
| 1.1 | Introduction | 2 |
| 1.2 | Motivation | 2 |
| 1.3 | Problem Statements | 2 |
| 1.4 | Research Objectives | 4 |
| 1.5 | Research Contribution | 4 |
| 1.6 | Organization of Thesis | 4 |
| 1.7 | Conclusion | 5 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 2** | |  |
| **Background Study and Literature Review** | | 25-34 |
| 3.1 | Introduction | 26 |
| 3.2 | Deoxyribonucleic Acid (DNA) | 26 |
| 3.3 | Protein | 27 |
| 3.4 | Genes | 28 |
| 3.5 | Microarray Technology | 29 |
| 3.6 | MicroRNA Technology | 31 |
| 3.7 | Literature Review | 32 |
| 3.8 | Conclusion | 34 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 3** | |  |
| **Siamese Network and BERT** | | 25-34 |
| 3.1 | Introduction | 26 |
| 3.2 | Deoxyribonucleic Acid (DNA) | 26 |
| 3.3 | Protein | 27 |
| 3.4 | Genes | 28 |
| 3.5 | Microarray Technology | 29 |
| 3.6 | MicroRNA Technology | 31 |
| 3.7 | Literature Review | 32 |
| 3.8 | Conclusion | 34 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 4** | |  |
| **Methodology** | | 25-34 |
| 3.1 | Introduction | 26 |
| 3.2 | Deoxyribonucleic Acid (DNA) | 26 |
| 3.3 | Protein | 27 |
| 3.4 | Genes | 28 |
| 3.5 | Microarray Technology | 29 |
| 3.6 | MicroRNA Technology | 31 |
| 3.7 | Literature Review | 32 |
| 3.8 | Conclusion | 34 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 5** | |  |
| **Experiment Results** | | 25-34 |
| 3.1 | Introduction | 26 |
| 3.2 | Deoxyribonucleic Acid (DNA) | 26 |
| 3.3 | Protein | 27 |
| 3.4 | Genes | 28 |
| 3.5 | Microarray Technology | 29 |
| 3.6 | MicroRNA Technology | 31 |
| 3.7 | Literature Review | 32 |
| 3.8 | Conclusion | 34 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER 6** | |  |
| **Conclusion and Future Scopes** | | 25-34 |
| 3.1 | Introduction | 26 |
| 3.2 | Deoxyribonucleic Acid (DNA) | 26 |
| 3.3 | Protein | 27 |
| 3.4 | Genes | 28 |
| 3.5 | Microarray Technology | 29 |
| 3.6 | MicroRNA Technology | 31 |
| 3.7 | Literature Review | 32 |
| 3.8 | Conclusion | 34 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table Number** | **Table Title** | **Page No.** |
| 2.1 | Staging of Pancreatic Cancer. (N denotes regional lymph | 16 |
|  | nodes, M distant metastases and T primary tumor |  |
| 4.1 | mRNA Microarray Datasets Configuration at a glance | 38 |
| 4.2 | microRNA Expression Dataset Configuration at a glance | 38 |
| 6.1 | Comparison in data content between TarBase and | 57 |
|  | miRecords |  |
| 6.2 | Comparison in miRNA target prediction programs | 58 |
|  | executed among combined miRNA target resources |  |
| 7.1 | Number of identified of DEGs in mRNA datasets | 65 |
|  | applying different methods and p-adjustment |  |
| 7.2 | Number of DEMs identified for micorRNA datasets | 66 |
|  | applying different methods and p-adjustment |  |
| 7.3 | The number of Upregulated and Downregulated DEGs | 66 |
|  | for mRNA datasets |  |
| 7.4 | The identified DEMs, their p-values, adjusted p-values | 67 |
|  | and log fold-change values |  |
| 7.5 | Target genes for DEMs predicted by miRecords Tool | 68 |
| 7.6 | Comparison of log-rank P-values for different cancers for | 74 |
|  | different percentiles for ECT2 |  |
| 7.7 | Comparison of log-rank P-values for different cancers for | 75 |
|  | different percentiles for NRP2 |  |
| 7.8 | Comparison between our research with previous study | 76 |

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Figure Caption** | **Page No.** |
| 1.1 | Global Map Presenting the National Ranking of Cancer | 2 |
|  | as a Cause of Death at Ages Below 70 Years in 2015 |  |
| 1.2 | Ten Leading Cancer Types for the Estimated New | 4 |
|  | Cancer Cases and Deaths by Sex, United States, 2019 |  |
| 1.3 | Bar Chart of Region-Specific Incidence Age- | 5 |
|  | Standardized Rates by Sex for Pancreatic Cancer in |  |
|  | 2018 |  |
| 2.1 | Components of Pancreatic Cancer | 12 |
| 2.2 | Pancreatic cancer incidence and mortality in men and | 19 |
|  | women, by regions |  |
| 2.3 | Pancreatic cancer incidence in men and women | 20 |
| 2.4 | Pancreatic cancer mortality in men and women | 22 |
| 3.1 | Summary of DNA Microarrays | 29 |
| 6.1 | Example of OncoLnc search outcomes | 60 |
| 7.1 | Workflow diagram of our research | 64 |
| 7.2 | Illustration of identifying common DEGs using the | 68 |
|  | Venn-Diagram |  |
| 7.3 | Display of the common genes among DEGs and target | 70 |
|  | genes of DEMs by Venn-Diagram |  |
| 7.4 | Presentation of ECT2 & NRP2 for 10-90 percentile | 71 |
| 7.5 | Presentation of ECT2 & NRP2 for 20-80 percentile | 72 |
| 7.6 | Presentation of ECT2 & NRP2 for 40-60 percentile | 73 |

**CHAPTER 1**

**Introduction**

*Introduction*

*Motivation*

*Problem Statements*

*Research Objectives*

*Research Contribution*

*Organization of Thesis*

*Conclusion*

* 1. Introduction

Recent years have seen the rise of community question answering forums, which allow users to ask questions and to get answers in a collaborative fashion. One issue with such forums is that duplicate questions easily become ubiquitous as users often ask the same question, possibly in a slightly different formulation, making it difficult to find the best (or one correct) answer [1]. Many forums allow users to signal such duplicates, but this can only be done after the duplicate question has already been posted and has possibly received some answers, which complicates merging the question threads. Discovering possible duplicates at the time of posting is much more valuable from the perspective of both (i) the forum, as it could prevent a duplicate from being posted, and (ii) the users, as they could get an answer immediately [2].

Quora is a social media website where questions are posted by users and answered by experts who provide quality insights. Other users can cooperate by editing questions and suggesting more accurate answers to the submitted questions. According to statistics provided by the Director of Product Management at Quora on 17 September 2018, Quora receives 300 million unique visitors every month, which raises the problem of different users asking similar questions with same intent but in different words [3].

One of the most crucial tasks which can be achieved by effective natural language processing is the task of evaluating sentence similarity. Determining sentence similarity between a pair of sentences finds many applications such as automated short-answer grading and machine translation [4] to name a few.

In this report, we have presented different approaches which can be used to determine the similarity between question pairs obtained from Quora. We define a question pair to be duplicate if they express the same intent.

* 1. Motivation

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term [5]. If a duplicate question is spotted by an algorithm, the user can be directed to it and reach the answer faster. Then knowledge sharing will be more efficient and effective in many ways. If there are fewer duplicate questions in a forum then that forum will be more popular and effective. Our main motivation is that save people's important time and find the best answer in minimum time. We want to assure you that the same answer doesn't written in different locations.

* 1. Problem Statements

In this problem, we have to detect duplicate question pair among two questions. Here we have used a large dataset called "Quora Question Pairs" [5]. It's a Kaggle dataset. For each question pair, we have two different questions. We have to identify these two questions have similar intent or not. For example, questions like ‘How can I be a good photographer?’’ and ‘‘What should I do to be a great photographer?’’ are identical because both have the same meaning and should be answered only once. Some questions, like ‘‘How old are you?’’ and ‘‘What is your age?’’ do not have the same wording. However, the context remains the same. Therefore, such questions are also considered duplicates. It can be an overhead to have different pages for such questions.

More formally, the followings are our problem statements

* Identify which questions asked on Quora are duplicates of questions that have already been asked.
* This could be useful to instantly provide answers to questions that have already been answered.
* We are tasked with predicting whether a pair of questions are duplicates or not.
  1. Research Objectives

The main objective of this research is to identify the duplicate question pairs. Identifying whether two sentences in the pair convey the same meaning or not.

More formally, the followings are our objectives

* Identify duplicate question pairs with higher accuracy
* Make knowledge sharing more efficient and effective
* Save the important time of readers and writers
  1. Research Contribution

The main contribution of the research is to identify duplicate question pairs with higher accuracy. Previous studies revealed that duplicate questions could be identified by SVM, SVM with n-grams, random forest, a continuous bag of words, LSTM, LSTM with attention, Manhattan LSTM, Siamese LSTM, Decision Tree, and XGBoost, etc. The main problem with these approaches is accuracy is not satisfied. We have implemented different new approaches like BERT Bi-LSTM, Siamese BERT, RoBERTa Bi-LSTM, RoBERTa for sequence classification, etc.

* 1. Organization of Thesis

**Chapter 2 - Background Study and Literature Review**

This chapter discusses the necessary terms required to understand for our research and describes the previous researches from diverse perspectives.

**Chapter 3 – Siamese Network, LSTM, BERT, and RoBERTa**

This chapter discusses the Siamese network, LSTM, BERT, and RoBERTa architecture.

**Chapter 4 – Methodology**

This chapter discusses the dataset, dataset analysis and pre-processing, activation functions, loss functions, learning rate, optimization algorithm,

overfitting, and our proposed methodologies to detect duplicate question pairs.

**Chapter 5 – Results and Performer Analysis**

This chapter discusses our experiment results and our findings, also compare our results to the previous work on this topic.

**Chapter 6 – Conclusion and Future Scopes**

This chapter concludes with the findings of our research mentioning the aspects and future scopes of this study.

* 1. Conclusion

In this chapter, we have attempted to present the importance of the duplicate question pair detection study. We have also introduced the significance of our research and presented why we have selected this NLP task as our research domain. Furthermore, we have discussed the research objectives, contribution and thesis organization to conclude the chapter.

**CHAPTER 2**

**Background Study and Literature Review**

*Introduction*

*Quora Question Pairs*

*Duplicate Question Pair*

*Necessity of Duplicate Question Detection*

*Literature Review*

*Conclusion*

**2.1 Introduction**

This chapter presents the necessary terms and topics for this research. Moreover, a literature review has been introduced with popular and recent researches and their findings. Finally, the domain of our research is discussed.

**2.2 Quora Question Pairs**

Quora is a place to gain and share knowledge — about anything. It’s a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world [5]. Quora is a growing platform comprising a user generated collection of questions and answers. The questions and answers are created, edited, and organized by the users. Enormous number of users on the Quora website makes it unavoidable to have multiple questions from different users with similar intent, which raises the issue of duplicate questions [6]. Quora is a question answering website where users ask questions and other users respond. The best answers are up-voted and these answers are a valuable learning resource for many topics.

Quora Question Pairs is an active Kaggle Competition, which challenges participants to tackle the natural language processing (NLP) problem of identifying duplicate questions [5]. The issue of duplicate questions stems from the enormous number of visitors on the Quora website (a platform for asking questions and connecting with people that contribute answers), making it hard to avoid having similar worded questions from different users. Effectively detecting duplicate questions not only saves time for seekers to find the best answer to their questions, but also reduces the effort of writers in terms of answering multiple versions of the same question [5].

Quora question pair is a Kaggle dataset. The dataset contains 404290 question pairs. Each line contains IDs for each question in the pair, the full text for each question, and a binary value that indicates whether the line truly contains a duplicate pair.

**2.3 Duplicate Question Pair**

The duplicate question means whether two questions in the pair convey the same meaning or not. For example, questions like ‘‘How can I be a good photographer?’’ and ‘‘What should I do to be a great photographer?’’ are identical because both have the same meaning and should be answered only once. Some questions, like ‘‘How old are you?’’ and ‘‘What is your age?’’ do not have same wording. However, the context remains the same. Therefore, such questions are also considered duplicate [6]. For instance, the question pair consisting of sentences “What is the solution for this question?” and “What is the solution to this question?” is labelled as non-duplicated, despite the fact that the two questions only differ in an insignificant stop word [7].

**2.3 Necessity of Duplicate Question Detection**

As the user base of these applications grow, the magnitude of questions and answers in their archives also grows to the possibility that a question asked by a user has already been asked before and answered. In such a scenario, it would be ideal for the application to suggest previously asked questions similar to the one which the user wishes to ask in order to improve user experience and enable efficient knowledge-sharing. This would save waiting time for the user and reduce frustration which can occur if the user sees different versions of the same question on his feed thus preventing display of other, more relevant content. Additionally, ensuring that duplicate or similar questions are not asked repeatedly reduces the burden on the storage and processing infrastructure of the application.

Identifying the duplicate questions at Quora and merging them makes knowledge sharing more efficient and effective in many ways. This way, the seekers can get answers to all the questions on a single thread and writers do not need to write the same answer on different locations for the same question. They can get larger number of readers than if the readers are divided in several threads [6].

Multiple questions with similar wording can cause readers to spend more time to find the best answer, and make writers answer multiple versions of the same question [6].

Automatic duplicate question detection alleviates labor and effort for users with high reputation [8].

**2.4 Literature Review**

Detecting semantically equivalent sentences or questions has been a long-standing problem in natural language processing and understanding. As Dey et al [9] demonstrate, traditional machine learning algorithms such as Support Vector Machines (SVMs) using hand-picked and heterogeneous features such as word overlap/ similarity, negation modeling, sentence/phrase composition [10] and extensively pre-processed data perform well on the SemEval-2015 dataset. This task is similar to the paraphrase identification problem, which is a thoroughly researched Natural Language Processing (NLP) task [11]. It uses Natural Language Sentence Matching (NLSM) to decide whether a pair of sentences with same intent is written in different words or not [12]. Feature engineering has been the center of focus for most of the traditional methods developed by different practitioners. The common features used are bag of words (BOW), term frequency and inverse document frequency (TF IDF), unigrams and bigrams. Support Vector Machine (SVM), used with different feature extraction techniques such as BOW or n-gram vectors, is one of the main methods in text categorization [13]. Recently, deep learning approaches have achieved very high performance across several Natural Language Processing (NLP) tasks especially in Semantic Text Similarity [10] [14] [15]. Deep models, trained with task-specific feature engineering, provided impressive results in semantic analysis and similarity measure. The researcher showed that meaningful semantic symmetries can be captured by using pre-trained word embeddings [16]. Deep models can be combined with word embeddings and used to express the semantic meaning of text chunks with satisfactory accuracy.

LSTM based neural networks have shown great outcomes for tasks such as categorization of text and retrieval of information [17]. A research [18] proposed supervised and semi-supervised methods based on LSTM that used region embedding method for embedding the text regions of adjustable dimensions. Another work [19], proposed a Neural Network model and studied documents represented in form of vectors in an integrated manner. First, the model used CNN or LSTM to study the vector form of the sentences. Then, the context of sentences and their relations, of a given document, was determined in the distributed vector representation with recurrent neural network (RNN). A novel approach known as the C-LSTM network was used for representation of sentences and classification of text. This architecture combined the capabilities of CNN and LSTM Neural Networks. It used CNN to extract high-level features which were then fed to LSTM [20]. Another research [21], proposed a Tree based LSTM model and used it to predict the similarity between two sentences. Skip-thought based approach was proposed which used skip-gram approach of word2vec from the word to sentence level [22]. First, the sentences were passed through RNN layer to get skip-through vector. Then, it attempted to reconstruct the previous and next sentences.

In spite of aforementioned works, Siamese architecture is one of the most frequently used learning frameworks to project question and answer pairs into a joint space [23]. In another study [10], Siamese LSTM made use of pre-trained word embedding vectors for converting the sentences. For final result, Manhattan distance was calculated to measure the closeness of the pair of sentences. CNNs have achieved great results in classification [24] and in other Natural Language Processing (NLP) tasks [25]. Another research [26] applied Siamese CNN model that used several convolution and pooling processes to produce sentence embeddings. However, using pre-trained word embeddings that are not related to the dataset limits the results of above-mentioned models. There are only few researches done on Quora dataset [27]. CNN based model used with GloVe embedding, which consists of 100 dimensions Wikipedia vectors, attained 80.4% accuracy [12]. Another work [28], applied the Siamese GRU using a bi-layer similarity network and achieved 85.0% accuracy. Support vector classifier model trained using the pre-computed features ranging from longest common substring and sub sequences to word similarity based on lexical and semantic resources also attained 85% accuracy. In [29], a bilateral multi-perspective matching (BiMPM) model was applied using the ‘‘matching-aggregation’’ framework and 88.17% accuracy was achieved.

In the paper named Duplicate Questions Pair Detection Using Siamese MaLSTM [6] applied the Siamese MaLSTM model with the blending of three pre-trained word embeddings Google News Vector, FastText Common Crawl, and FastText Common Crawl subword embedding and achieved 91.14% accuracy.

In the paper named "Natural language understanding with the quora question pairs dataset" [8] applied SVM, Random Forest, Decision Tree, and Continuous Bag of Words and achieved 83.4% accuracy from their best performer CBOW model.

**2.5 Conclusion**

In this chapter, we have reviewed Quora Question Pairs, Duplicate Question pair, the necessity of duplicate question detection. Literature review segment presented diverse studies and up-to-date researches for understanding the problem domain and deciding the domain of interest for our research.

**CHAPTER 3**

**Siamese Network, LSTM, BERT and RoBERTa**

*Introduction*

*Siamese Network*

*Why Siamese Network*

*LSTM*

*BERT*

*BERT Pre-training*

*What Makes BERT Different*

*RoBERTa*

*Conclusion*

**3.1 Introduction**

This chapter presents the Siamese network architecture, why the Siamese network is efficient to find similarity between two sentences. Further, we will discuss the LSTM, the BERT, the RoBERTa, BERT pre-training process, and why BERT is better, etc.

**3.2 Siamese Network**

A Siamese neural network (sometimes called a twin neural network) is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors [30].

The Siamese Network architecture is as follows:

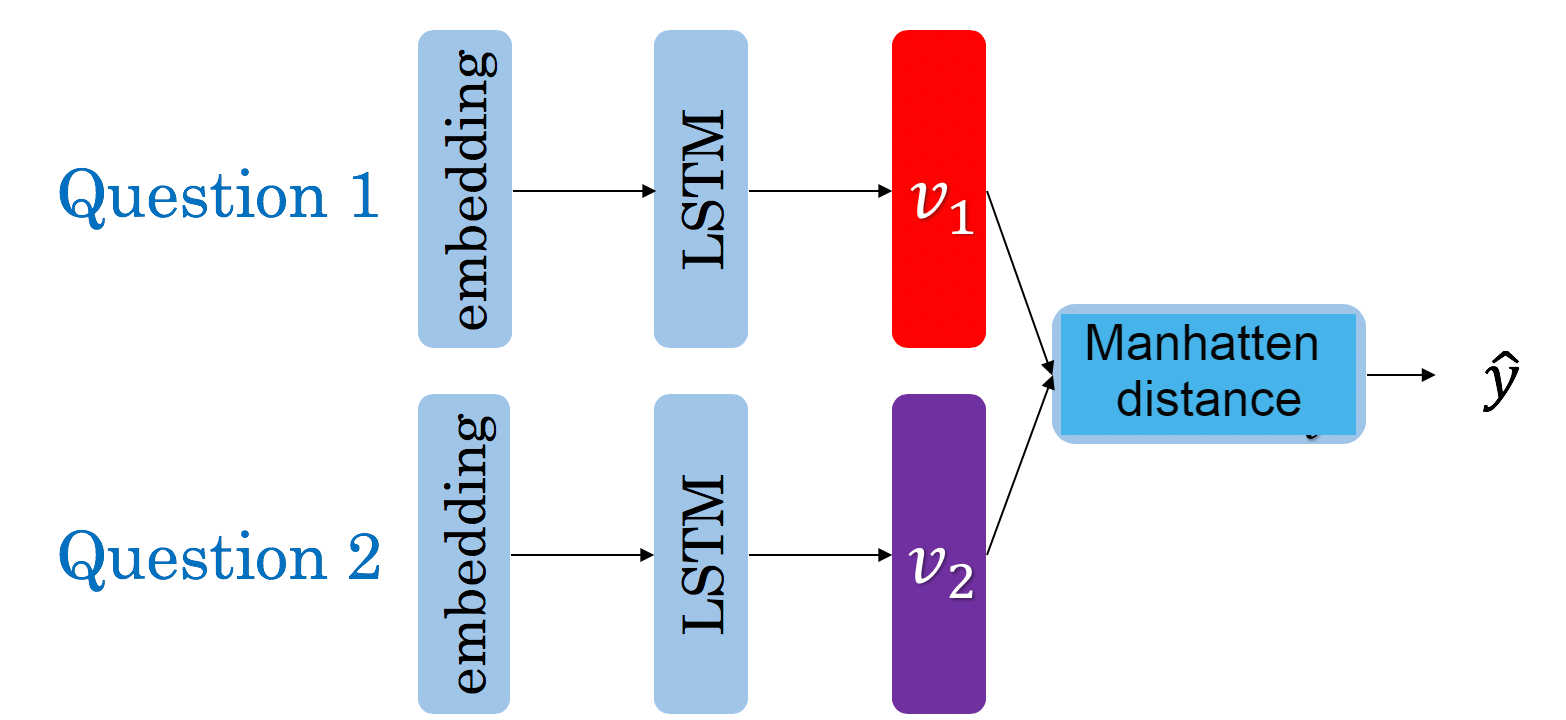


Figure 3.1: The Siamese MaLSTM architecture

Siamese Neural Networks (SNNs) are a type of neural networks that contains multiple instances of the same model and share same architecture and weights. This architecture shows its strength when it has to learn with limited data and we don’t have a complete dataset, like in Zero / One shot learning tasks [31]. Traditionally, a neural network learns to predict over multiple classes. This poses a problem when we need to add / remove new classes to the data. In this case, we have to update the neural network and retrain it on the whole dataset. Also, deep neural networks need a large volume of data to train on. SNNs on the other hand learn a similarity function. Thus, we can train it to see if two images are the same (which we will do here). This enables to classify new classes of data without training the network again [31].

**3.2.1 Why Siamese Network**

Similarity has always been a key aspect in computer science and statistics. Any time two element vectors are compared, many different similarity approaches can be used, depending on the final goal of the comparison (Euclidean distance, Pearson correlation coefficient, Spearman’s rank correlation coefficient, and others). But if the comparison has to be applied to more complex data samples, with features having different dimensionality and types which might need compression before processing, these measures would be unsuitable. In these cases, a Siamese neural network may be the best choice: it consists of two identical artificial neural networks each capable of learning the hidden representation of an input vector. The two neural networks are both feedforward perceptrons, and employ error back-propagation during training; they work parallelly in tandem and compare their outputs at the end, usually through a cosine distance. The output generated by a Siamese neural network execution can be considered the semantic similarity between the projected representation of the two input vectors [30].

**3.3 LSTM Networks**

Long short-term memory (LSTM) is an artificial [recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNN) architecture used in the field of [deep learning](https://en.wikipedia.org/wiki/Deep_learning). Unlike standard [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_network), LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video) [32].

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! [33]

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer [33].

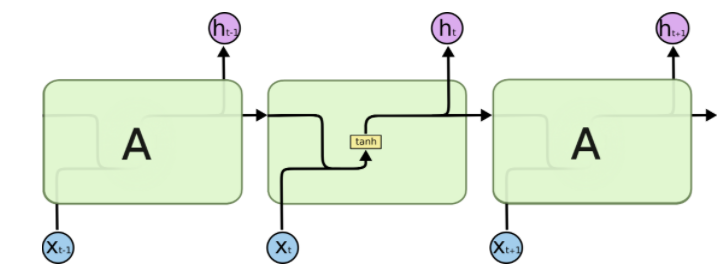


Figure 3.2: The repeating module in a standard RNN contains a single layer [33]

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

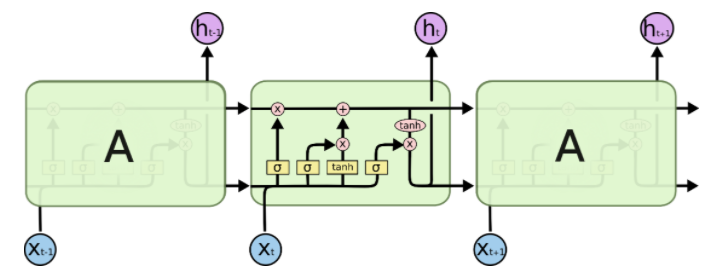


Figure 3.3: The repeating module in an LSTM contains four interacting layers [33]

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell [32].

The key to LSTMs is the cell state. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged [33].

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates [33].

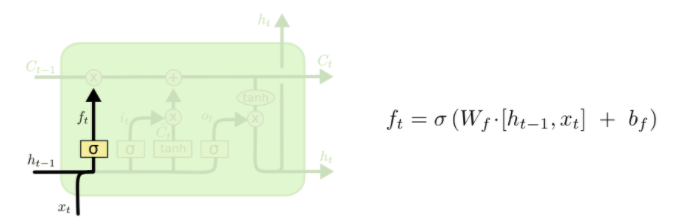
Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!” [33].

An LSTM has three of these gates, to protect and control the cell state.

**3.3.1** Step-by-Step LSTM Walk Through

The first step in our LSTM is to decide what information we’re going to throw away from the cell state. This decision is made by a sigmoid layer called the “forget gate layer.” It looks at ht−1and xt, and outputs a number between 0 and 1 for each number in the cell state Ct−1. A 1 represents “completely keep this” while a 0 represents “completely get rid of this” [33].

Figure 3.4: LSTM Forget gate [33]

The next step is to decide what new information we’re going to store in the cell state. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, t, that could be added to the state. In the next step, we’ll combine these two to create an update to the state [33].

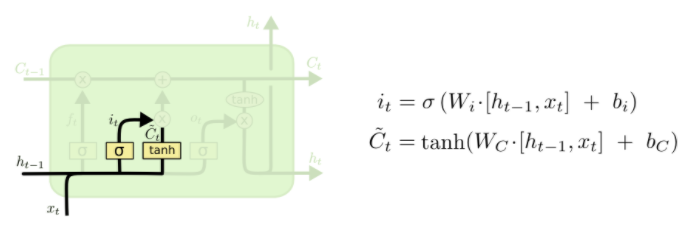


Figure 3.5: LSTM Input gate [33]

It’s now time to update the old cell state, **Ct−1**, into the new cell state **Ct**. The previous steps already decided what to do, we just need to actually do it [33].

We multiply the old state by **ft**, forgetting the things we decided to forget earlier. Then we add **it∗t** . This is the new candidate values, scaled by how much we decided to update each state value [33].

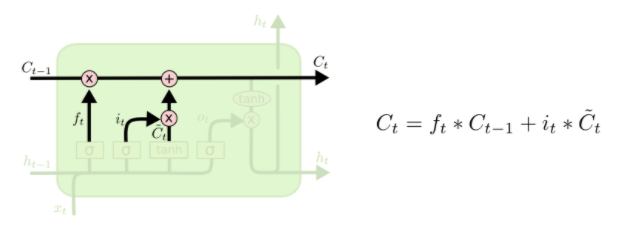


Figure 3.6: New cell state [33]

Finally, we need to decide what we’re going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we’re going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to [33].

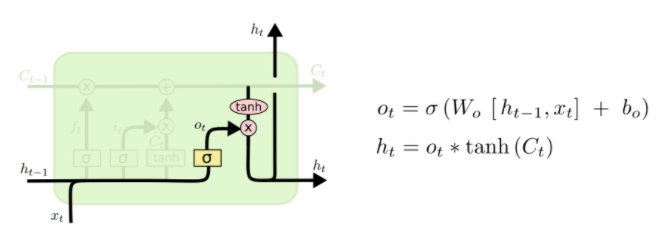


Figure 3.7: LSTM output gate [33]

**3.3.2 Necessity of LSTM**

The LSTM is introduced to overcome the problem with simple RNN. The advantage of an LSTM cell compared to a common recurrent unit is its cell memory unit. The cell vector has the ability to encapsulate the notion of forgetting part of its previously stored memory, as well as to add part of the new information.

**3.4 BERT**

Bidirectional Encoder Representations from Transformers (BERT) is a [Transformer](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model))-based [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) (NLP) pre-training developed by [Google](https://en.wikipedia.org/wiki/Google). BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google [34] [35]. As of 2019, Google has been leveraging BERT to better understand user searches [36].

Unlike recent language representation models [37], BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications [34].

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute  
improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 [34].

**3.4.1 BERT Architecture**

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT’s goal is to generate a language model, only the encoder mechanism is necessary [38].

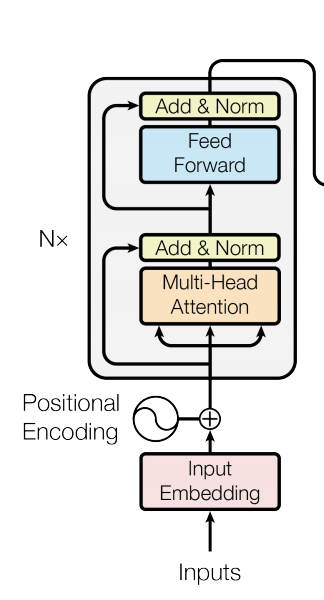


Figure 3.8: BERT Encoder [34]

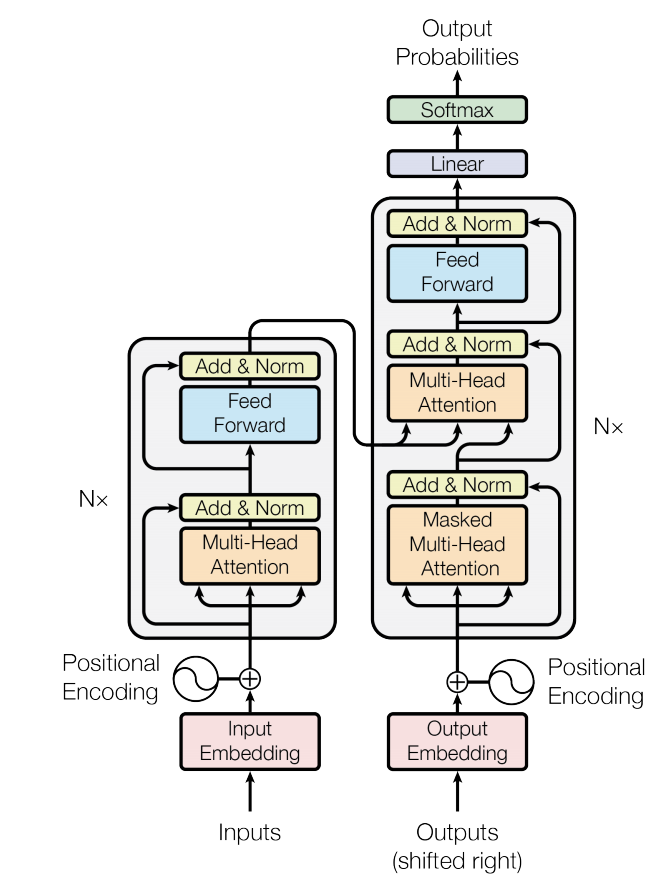


Figure 3.9: Full transformer with BERT Encoder [34]

Both encoder and decoder sections of transformer are a stack of 6 identical layers of multi-head attention and feed forward sublayers. Each sublayer takes a residual connection from its previous inputs, adds it to the sub layer output and normalises it to produce final output of the sub layer. To allow for residual addition, all sub-layers produce an output of dimension 512 [39].

The decoder has an additional sublayer which performs attention over output of encode shifted by one position. In addition to this shifting this layer is masked so that the prediction at a time step i doesn’t attend to the outputs of future time steps j, where j > I [39].

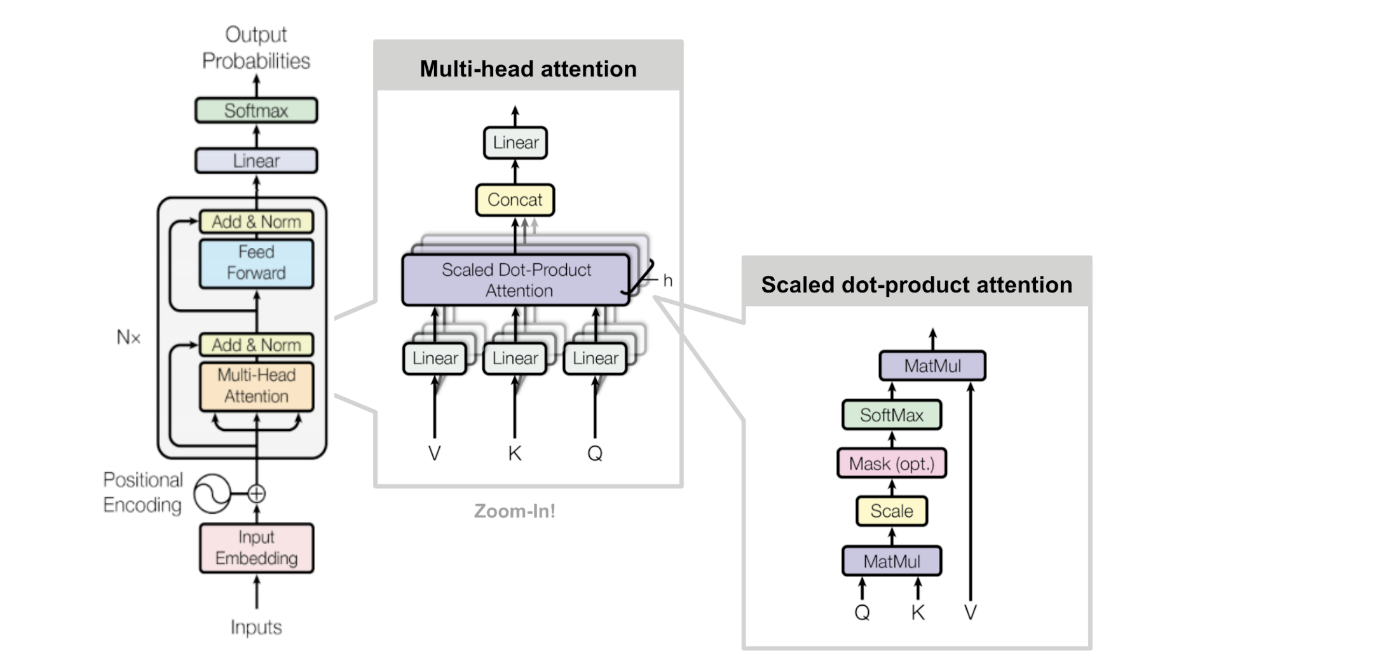
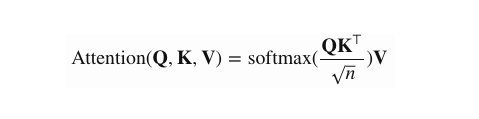
****

Figure 3.10: Multi-head attention [34]

Multi-head attention function takes input vectors query (Q) and key (K), of dimensions k, and value (V) vector of dimension v.



All vectors Q, K, V are packed together into matrices for applying computationally optimal matrix multiplication after scaling by a factor k, to avoid vanishing gradient problem [39].

BERT primarily report results on two model sizes: BERTBASE (L=12, H=768, A=12, Total Parameters=110M) and BERTLARGE (L=24, H=1024, A=16, Total Parameters=340M). They denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A [34].

**BERT input representation:** The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

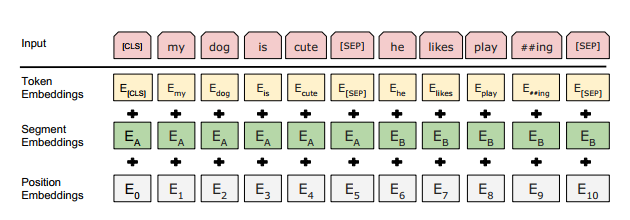


Figure 3.11: BERT input Representation [34]

**3.4.2 Pre-training BERT**

BERT was pre-trained using two unsupervised tasks, described in this section. This step  
is presented in the left part of Figure 3.5.

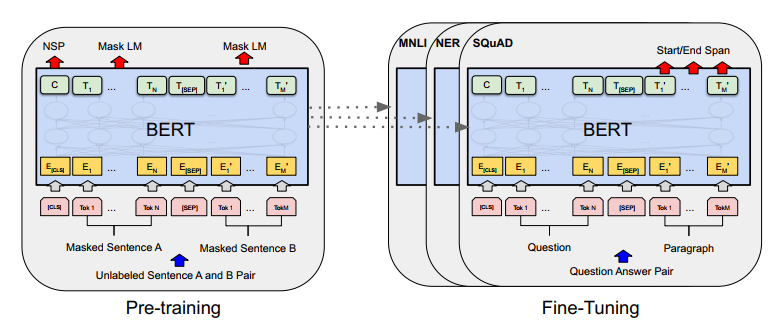


Figure 3.12: Overall pre-training and fine-tuning procedures for BERT [34]

**Task #1:** Masked LM Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-toright and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right or right-to-left, since bidirectional conditioning would allow each word to indirectly “see itself”, and the model could trivially predict the target word in a multi-layered context [34].

In order to train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a “masked LM” (MLM), although it is often referred to as a Cloze task in the literature [40].In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. In contrast to denoising auto-encoders [41], we only predict the masked words rather than reconstructing the entire input [34].

Although this allows us to obtain a bidirectional pre-trained model, a downside is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, we do not always replace “masked” words with the actual [MASK] token. The training data generator chooses 15% of the token positions at random for prediction. If the i-th token is chosen, we replace the i-th token with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time. Then, Ti will be used to predict the original token with cross entropy loss [34].

**Task #2:** Next Sentence Prediction (NSP) Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, we pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext) [34].

**3.4.3 Pre-training Data**

The pre-training procedure largely follows the existing literature on language  
model pre-training. For the pre-training corpus we use the BooksCorpus (800M words) [42] and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages and ignore lists, tables, and headers. It is critical to use a document-level corpus rather than a shuffled sentence-level corpus such as the Billion Word Benchmark [43] in order to extract long contiguous sequences [34].

**3.4.4 What Makes BERT Different?**

BERT builds upon recent work in pre-training contextual representations — including [Semi-supervised Sequence Learning](https://arxiv.org/abs/1511.01432), [Generative Pre-Training](https://blog.openai.com/language-unsupervised/), [ELMo](https://allennlp.org/elmo), and [ULMFit](http://nlp.fast.ai/classification/2018/05/15/introducting-ulmfit.html). However, unlike these previous models, BERT is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus (in this case, [Wikipedia](https://www.wikipedia.org/)) [35].

Why does this matter? Pre-trained representations can either be context-free or contextual, and contextual representations can further be unidirectional Or bidirectional. Context-free models such as [word2vec](https://en.wikipedia.org/wiki/Word2vec) or [GloVe](https://nlp.stanford.edu/projects/glove/) generate a single [word embedding](https://www.tensorflow.org/tutorials/representation/word2vec) representation for each word in the vocabulary. For example, the word “bank” would have the same context-free representation in “bank account” and “bank of the river.” Contextual models instead generate a representation of each word that is based on the other words in the sentence. For example, in the sentence “I accessed the bank account,” a unidirectional contextual model would represent “bank” based on “I accessed the” but not “account.” However, BERT represents “bank” using both its previous and next context — “I accessed the ... account” — starting from the very bottom of a deep neural network, making it deeply bidirectional [35].

A visualization of BERT’s neural network architecture compared to previous state-of-the-art contextual pre-training methods is shown below. The arrows indicate the information flow from one layer to the next. The green boxes at the top indicate the final contextualized representation of each input word: [35]

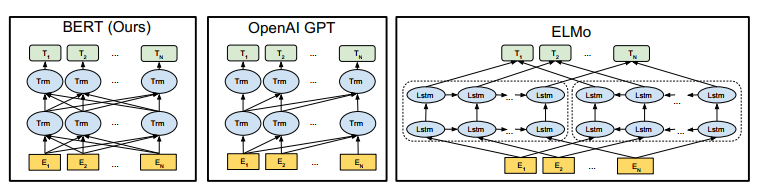


Figure 3.13: Comparison between BERT, GPT, and ELMo [34]

**The Strength of Bidirectionality**: If bidirectionality is so powerful, why hasn’t it been done before? To understand why, consider that unidirectional models are efficiently trained by predicting each word conditioned on the previous words in the sentence. However, it is not possible to train bidirectional models by simply conditioning each word on its previous and next words, since this would allow the word that’s being predicted to indirectly “see itself” in a multi-layer model [35].

**3.5 RoBERTa**

The BERT and RoBERTa have the same architecture. The only differences are the pre-training dataset, hyper parameters value, and removing next sentence prediction from pre-training task.

Authors of RoBERTa find that BERT was significantly undertrained and propose an improved recipe for training BERT models. For improving the performance of all of the post-BERT methods [44]. The modification of the BERT are

1. training the model longer, with bigger batches, over more data
2. removing the next sentence prediction objective
3. training on longer sequences
4. dynamically changing the masking pattern applied to the training data. We also collect a large new dataset (CC-NEWS) of comparable size to other privately used datasets, to better control for training set size effects.

The RoBERTa performs slightly better than the BERT because of the above-mentioned modification of the BERT.

Since we have already discussed the BERT architecture and pre-training process so we will not repeat it for RoBERTa.

**3.6 Conclusion**

In this chapter, we have reviewed the Siamese network, LSTM network, and the BERT, RoBERTa and why they are essential for this study. We briefly described the Siamese network, LSTM, and BERT with necessary figures.

**CHAPTER 4**

**Methodology**

*Introduction*

*Dataset*

*Dataset Analysis*

*Dataset Pre-processing*

*Activation Functions*

*Loss Functions*

*Learning Rate*

*Adam Optimization Algorithm*

*Overfitting*

*Evaluation Metrics*

*Manhattan Distance*

*Our Methodologies*

*Conclusion*

**4.1 Introduction**

This chapter presents the used dataset, dataset analysis and pre-processing, the study of activation functions, loss functions, learning rate, Adam optimization algorithm, overfitting, how to prevent overfitting, evaluation metrics, and our methodologies.

**4.2 Dataset**

We have used the Quora Question Pairs dataset. The dataset consists of a training set of 404351 question pairs, and is provided as part of a Kaggle competition [5].

The dataset is split in 75 and 25 ratio for training and testing respectively.

The name of dataset attributes with their description is shown in Table 4.1.

Table 4.1: Attributes description of dataset.

|  |  |
| --- | --- |
| Name of Attributes | Descriptions |
| id | unique ID of each pair |
| qid1 | ID of first question |
| qid2 | ID of second question |
| question1 | text of first question |
| question2 | text of second question |
| is duplicate | are the questions duplicates of each other (0 indicates not duplicate, 1 indicates duplicate) |

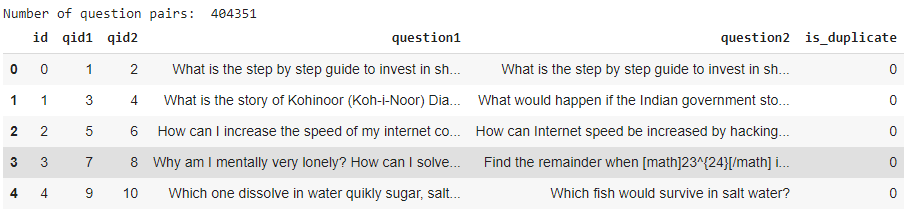


Figure 4.1: Snapshot of dataset

**4.3 Dataset Analysis**

Of the 404,351 question pairs, 255045 (63.1%) have a negative (0) label, and 149306 (36.9%) have a positive (1) label, making our dataset unbalanced.

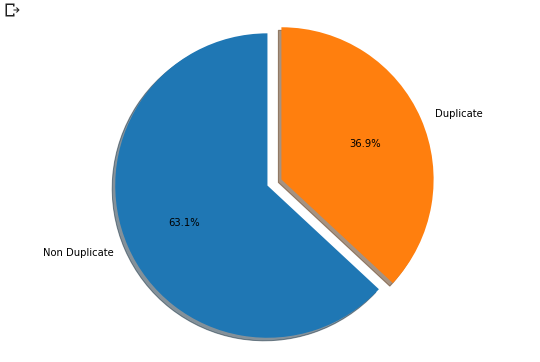


Figure 4.2: Duplicate Vs Non Duplicate pairs

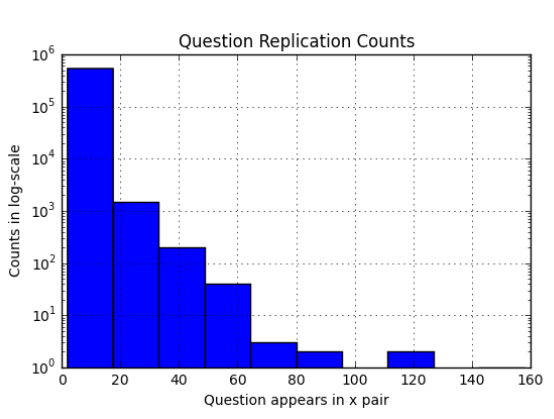
While every question pair is unique, every question within the questions pairs is not; 79:22% of questions appear more than once, with one of the  
questions appearing 158 times. Across all question pairs, there are 537,933 unique questions. Of these, 111,780 questions occur across across multiple pairs. Figure 4.3 shows the number of times a question appears against the number of questions  
for that many occurrences [8].

Figure 4.3: Histogram of Question Counts [8]

The first observation is that the training data set is noisy, i.e. there are question pairs labelled as non-duplicated where the two questions share almost all the words and the exact same meaning. For instance, the question pair consisting of sentences “What is the solution for this question?” and “What is the solution to this question?” is labelled as non-duplicated, despite the fact that the two questions only differ in an insignificant stop word [7].

**4.3.1 Dataset Pre-processing**

At first, we analyze the sentence length of the questions. Plot distribution of sentences as sentence length is shown in figure 4.4.

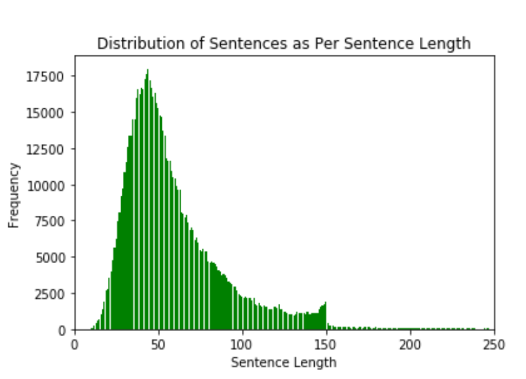


Figure 4.4: Sentence length distribution

Several NLP techniques are used such as conversion to lower letters of text,  
stop words removal, stemming, and tokenization, etc.

Then we analyze how many words have each question. After analyzing we set maximum question length 64 for the Siamese LSTM model input. Figure 4.5 shows the plot of words counts.

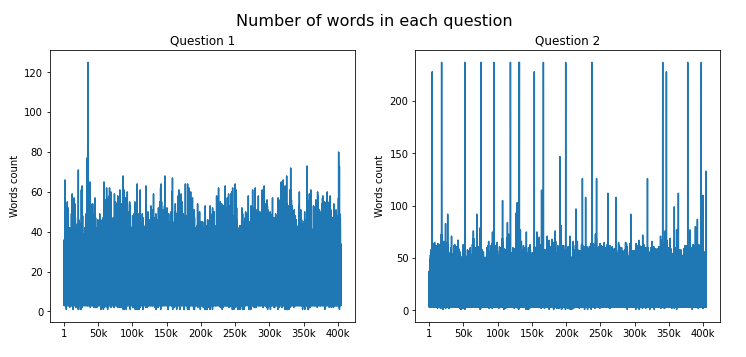


Figure 4.5: Words Counts in each question

Then we analyze how many tokens have each question after tokenizing using the BERT tokenizer. After analyzing we set maximum token length 128 for the BERT model input. Figure 4.6 shows the plot of tokens counts.

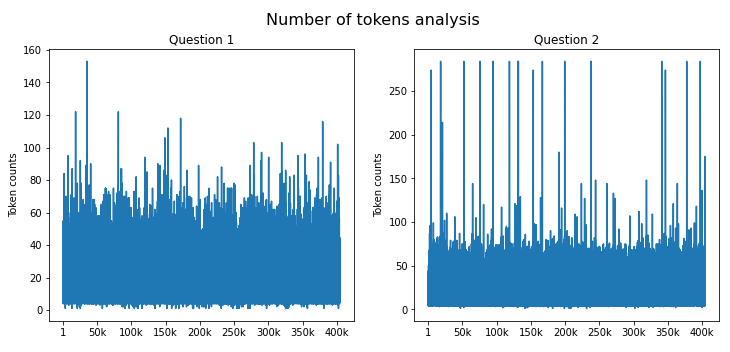


Figure 4.6: Tokens Counts in each question

**4.4 Activation Function**

The activation function is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. depending on the type of activation function. Activation function can be divided into two categories. One is the linear activation function and another is non-linear activation function. In linear activation functions, the output of the function will not be confined between any ranges. It does not help with the complexity or various parameters of usual data that is fed to the neural networks. On the other hand, non- linear activation function makes it easy for the model to generalize and adapt with a variety of data. This is the reason non-linear activation functions is the most used activation function. Some of the non-linear activation function is discussed below.

**4.4.1** Sigmoid or Logistic Activation function

The sigmoid function curve looks like an S-shape curve [45]. The Sigmoid function exists between 0 and 1. The function is differentiable so we can find the slope of the sigmoid curve at two points. The function is monotonic but its derivative is not. The logistic function can cause a neural network to get stuck at training time. The sigmoid activation function can be defined as



The sigmoid activation function has two major drawbacks.

* A very undesirable property of the sigmoid neuron is that when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. Therefore, if the local gradient is very small, it will effectively "kill” the gradient and almost no signal will flow through the neuron to its weights and recursively to its data.
* Sigmoid outputs are not zero-centered. This is undesirable since neurons in later layers of processing in a Neural Network would be receiving data that is not zero-centered.

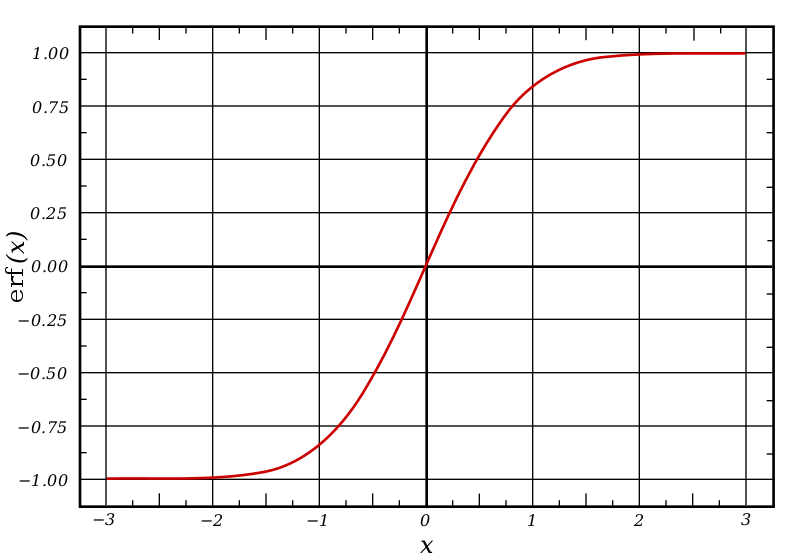


Figure 4.7: Sigmoid Function’s Curve [46]

**4.4.2** Hyperbolic Tangent or Tanh activation function

Tanh is also like a logistic sigmoid function but better. The range of the tanh function is from -1 to 1. Tanh is also sigmoid or S-shape. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. The function is differentiable. The tanh function is mainly used for classification between two class. Tanh function can be mathematically defined as



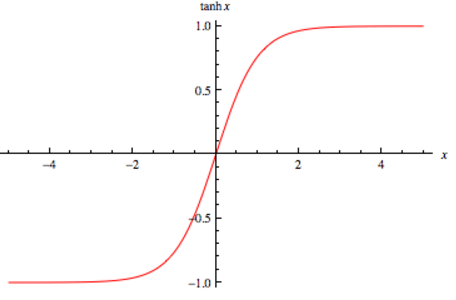


Figure 4.8: Tanh curve [47]

**4.4.3** ReLU (Rectified Linear Unit)

The ReLU is the most used activation function in the world right now. Since it is used in almost all the convolutional neural network. It is computationally efficient and converges much faster than most other activation functions [13]. The ReLU function can be mathematically defined as **f(x) = max (0,x)**.

The ReLU is half rectified, f(x) is zero when x is less than zero and f(x) is equal x when x is above or equal to zero. It ranges from zero to infinity. Some pros and cons of ReLU are described below.

* It was found to greatly accelerate the convergence of stochastic gradient descent compared to the sigmoid functions.
* Compared to sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero.
* Unfortunately, RelU units can be fragile during training and can “die".



Figure 4.9: ReLU Curve [48]

**4.4.4** Softmax Activation Function

Softmax function [36] is an interesting activation function that turns numbers aka logits into probabilities that sum to one. Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes. It's also a core element used in deep learning classification tasks.

Softmax function outputs numbers that represent probabilities, each number's value is between 0 and 1 valid value range of probabilities. The range is denoted as [0, 1]. The numbers are zero or positive. The entire output vector sums to 1. That is to say, when all probabilities are accounted for, that's 100%. The softmax activation function is a more generalized version of the sigmoid activation function used for multiclass classification problem.

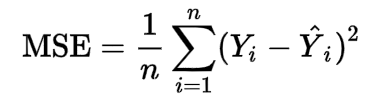
There are many other activation functions such as Leaky ReLU, sine ReLU. But the above discussed activation functions are used more often.

4.5 Loss Function

Loss function plays an important role in the artificial neural network that is used for measuring the inconsistency between the actual value and predicted value. It is a non-negative value, where the robustness of the model increases with the decrease of the value of loss function. The loss function is the hard core of empirical risk function as well as a significant component of the structural risk function. Neural networks are trained using stochastic gradient descent and require that you choose a loss function when designing and configuring your model. Different kind of loss function is discussed below.

4.5.1 Mean Squared Error

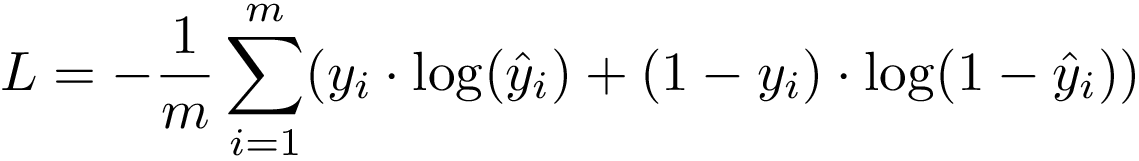
Mean Squared Error (MSE), or quadratic loss function is mainly used in linear regression as the performance measure, and the method of minimizing the error is called Ordinary Least Squares, the basic principle of OSL is that the optimized fitting line should be a line which minimizes the sum of squared distance of each of the regression line [49]. In other words, the line fitting the data must minimize the quadratic sum. The Mean Square can be mathematically defined as



Where, n is the number of training sample for each epoch and Yi is the actual label of i'th sample, and i is the predicted label of i'th sample. If we use the sigmoid activation function then MSE will suffer from slow convergence.

4.5.2 Cross-Entropy

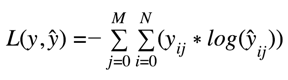
Cross-Entropy or Binary Cross Entropy is also called logarithmic loss and used for binary classification problem. Cross-entropy mainly measures the divergence between two probability distribution, if the value of cross-entropy is large, which means that the difference between two distribution is large, on the other hand, if the cross-entropy is small, which means that two distribution is similar to each other [50]. Cross-Entropy can be mathematically defined as the following equation, where the symbols bear their usual meaning.



Cross-entropy doesn't suffer from slow convergence while using sigmoid activation function Cross-entropy offers more training than Mean square error. So using Cross-entropy as loss function is more preferable than mean squared error.

4.5.3 Categorical Cross-Entropy

Categorical Cross-Entropy is just an extended version of cross entropy for multiclass classification problem which can be mathematically defined as the following equation where the symbols bear their usual meaning [50].



Choosing Loss function is an important part of the neural network construction The loss function is mainly application specific. In general, we use MSE for regression problem, binary cross-entropy for binary classification problem and categorical cross-entropy for multiclass classification problem.

4.6 Learning Rate

The rate in which the weights of the neuron or nodes are updated is known as the learning rate. The learning rate is an important factor if the learning rate is too small, it will take more time to converge on the other hand if the learning rate is too large it may overshoot the global minimum. The learning rate needs to be controlled as the effects of different learning rates can produce different results and the goal is to minimize the loss function. This is depicted in figure 5.5 as with low learning rates the improvements will be linear. With high learning rates, they will start to look more exponential. Higher learning rates will decay the loss faster, but they get stuck at worse values of loss (green line). This is because there is too much "energy" in the optimization and the parameters are bouncing around chaotically, unable to settle in a nice spot in the optimization landscape.

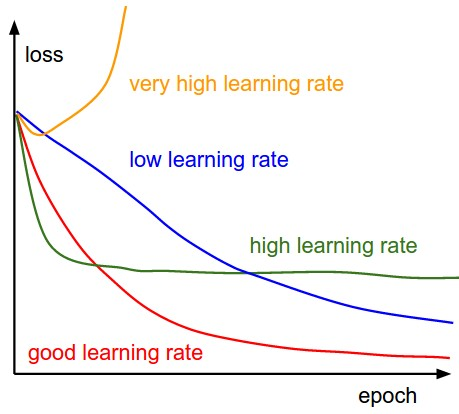


Figure 4.10: Learning rate effect [51]

4.7 Mini-Batch Gradient

Descent Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples

Xi:i+n , Yi:i+n)

It has two basic advantages one is, it reduces the variance of the parameter updates which leads to more stable convergence other is it can make use of highly optimized matrix optimizations that make computing gradient of a mini-batch very efficient. Common mini- batch sizes range between 32 and 256 which can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a neural network and the term SGD usually is employed also when mini-batches are used.

4.8 Adam Optimization Algorithm

Adaptive Moment Estimation (Adam) [52] is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface [53]. Adam optimization algorithms combine the heuristics of both Momentum and RMSProp. While momentum accelerates our search in direction of minima, RMS Prop impedes our search in the direction of oscillations. Adam optimization algorithm tries to combine both.

4.9 Overfitting

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize. Overfitting reduces the generalization power of artificial neural networks. There are various approaches in order to avoid overfitting. Dropout technique is discussed briefly in this section.

4.9.1 Dropout

The term "dropout” refers to dropping out units (both hidden and visible) in a neural network. Dropout is an extremely effective, simple and recently introduced regularization technique [54]. Dropout refers to ignoring units (i.e. neurons) during the training phase of a certain set of neurons which is chosen at random. During the training phase, for each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p, of nodes (and corresponding activations). During the testing phase, Use all activations, but reduce them by a factor p (to account for the missing activations during training).

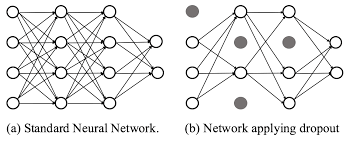


Figure 4.11: Dropout in neural network [55]

4.10 Evaluation Metrics

Evaluating our trained model use only accuracy metrics. Our main focus was to gain higher accuracy to detect duplicate question pairs.

Accuracy: Accuracy means that how many data points are predicted correctly. It is one of the simplest form of evaluation metrics. Accuracy is,

Accuracy =

4.11 Manhattan Distance

The distance between two points measured along axes at right angles. In a plane with p1 at (x1, y1) and p2 at (x2, y2), it is |x1 - x2| + |y1 - y2|.

4.12 Our Methodologies

To detect duplicate questions we have implemented several models. The approaches are explained below.

4.12.1 The Siamese MaLSTM Model

The model architecture is as follows

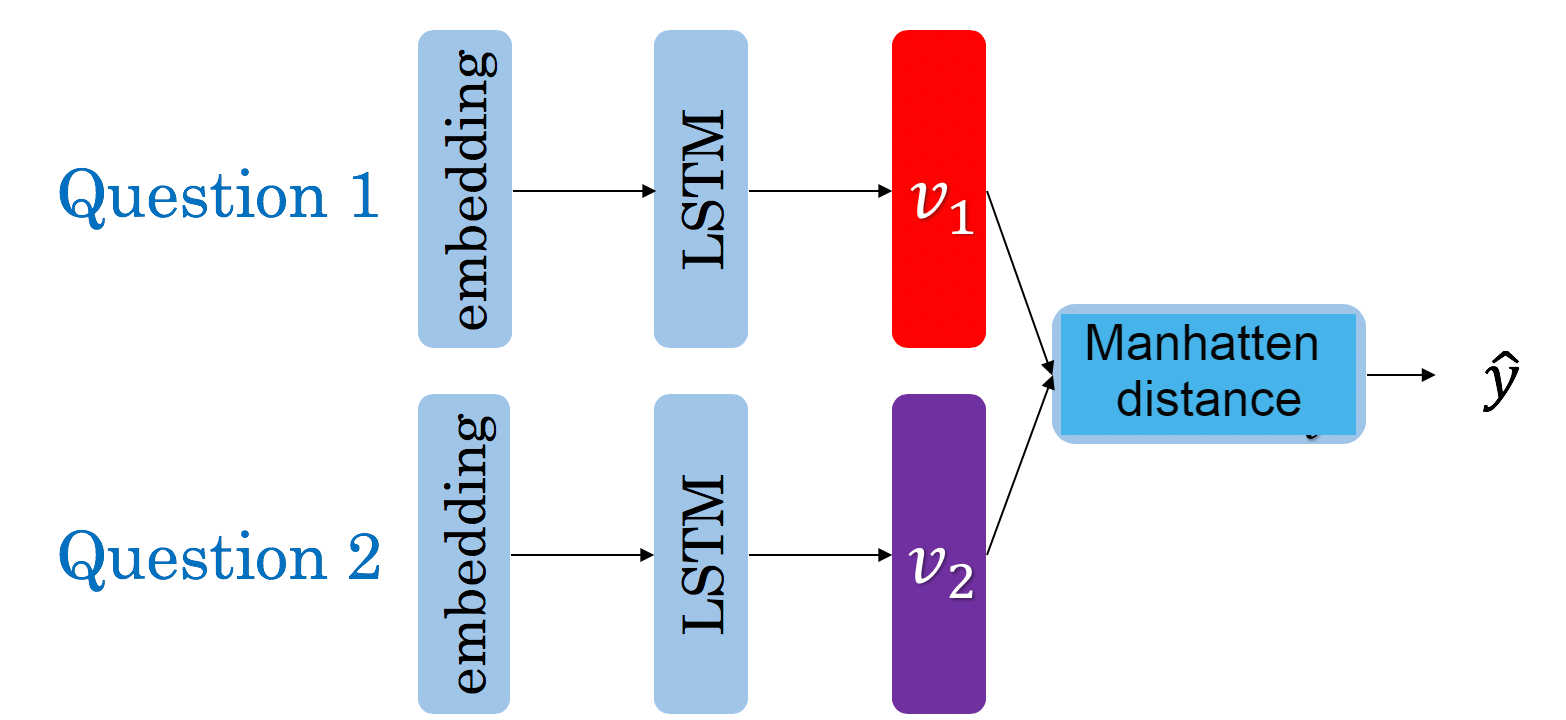


Figure 4.12: Siamese MaLSTM model architecture

Table 4.2: Siamese MaLSTM model parameters value

|  |  |
| --- | --- |
| **Siamese MaLSTM Model Parameters** | **Parameters Value** |
| **Maximum Question Length** | 64 |
| **Embedding dimension for keras embedding layer (without pre-trained embedding)** | 128 |
| **Embedding dimension for pre-trained word embeddings**  **(Google news vector, GloVe, FastText Crawl)** | 300 |
| **Number of LSTM units** | 64 |
| **Batch size** | 256 |
| **Learning rate** | 0.001 |
| **Loss function** | **Mean Squared Error** |

At first, we preprocess the dataset such as conversion to lower letters of text, stop words removal, stemming, and tokenization, etc.

Then we build the vocabulary and use different types of embeddings for feature extraction. We used 4 pre-trained embeddings. Our used embeddings are Google news vector embedding, GloVe embedding with 300 dimensions, FastText crawl embedding with 300 dimensions, and FastText crawl subwords embedding with 300 dimensions.

Flows of execution of the model as follows

* Two questions are given as the inputs (tensors of length 64) of the Siamese MaLSTM model.
* Then an embedding layer converts each word to a fixed dimension vector. The embedding layer used to overcome problems of one-hot encoding and learning aspects.
* After that, a shared weight LSTM layer generates the two different output vectors V1 and V2.
* Classification layer, the manhattan distance of the two vectors are calculated here if the manhattan distance is greater than 0.5 then the question pair is detected as a duplicate question pair otherwise detected as a non-duplicate question pair.

4.12.2 The BERT Bi-LSTM model

The model architecture is as follows

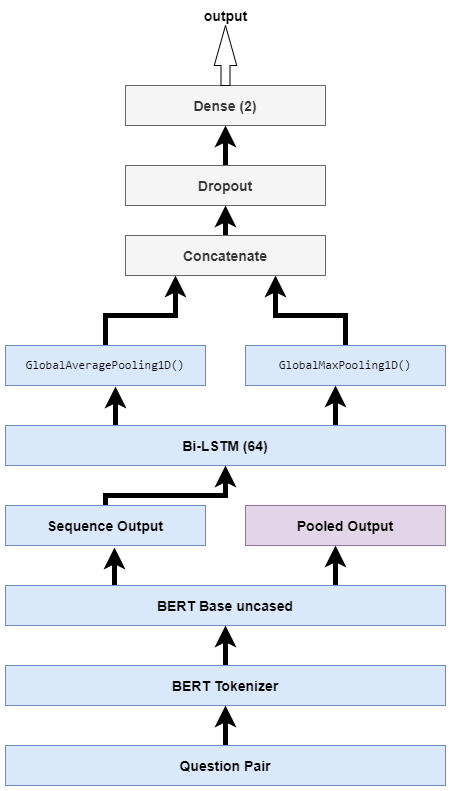
****

Figure 4.13: BERT Bi-LSTM model architecture

Table 4.3: BERT Bi-LSTM model hyper parameters value

|  |  |
| --- | --- |
| **BERT Bi-LSTM Model Parameters** | **Parameters Value** |
| **Sequence Length** | 64 |
| **Optimization Algorithm** | Adam |
| **Learning Rate** | 5e-5, 2e-5, 5e-6 |
| **Epsilon** | 1e-8 |
| **Batch size** | 32 |

4.12.3 The Siamese BERT model

The model architecture is as follows

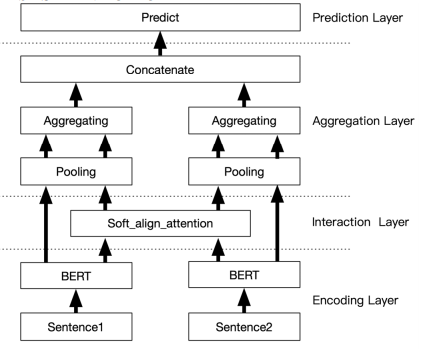


Figure 4.14: Siamese BERT model architecture [56]

Our implementation keynotes are below

* Here we used the BERT Base Cased model.
* Our pooling strategy was MEAN pooling.
* We concatenate the two output vectors u, v as follows [u, v, |u-v|].

4.12.4 The RoBERTa Bi-LSTM model

The model architecture as follows

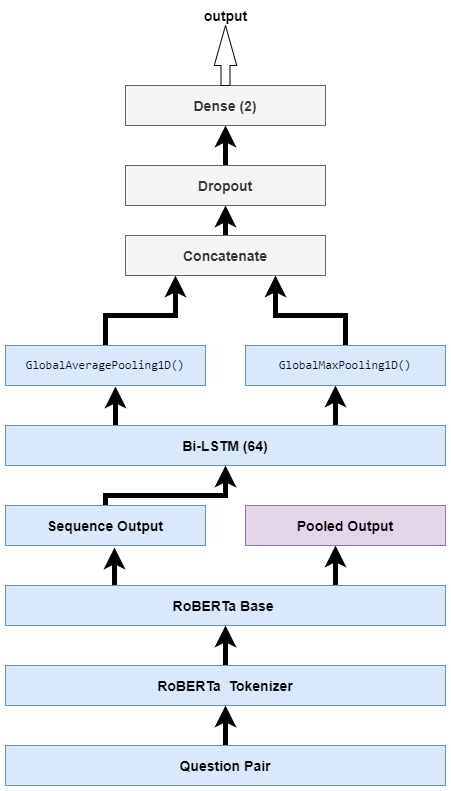


Figure 12.15: The RoBERTa Bi-LSTM model architecture

All the model hyper parameters values are the same as the BERT Bi\_LSTM model's hyper parameters.

4.13 Conclusion

In this chapter, we discussed the dataset, dataset analysis, and pre-processing. We also have discussed different activation functions and loss functions and which activation function and loss function have to choose, how the learning rate impact, how to prevent overfitting, etc. At the last of this chapter, we presented our methodologies to detect duplicate questions.

**CHAPTER 5**

**Results and Performance Analysis**

*Introduction*

*Dataset Splits*

*Experimental Environments*

*Results*

*Performance Analysis*

*Comparison with previous works*

*Conclusion*

5.1 Introduction

In the previous chapter, we have discussed our methodologies and implementation process to detect duplicate question pairs. In this chapter, we will present the results of our methodologies. Then we will describe the performance analysis of different strategies. At last, we will compare our results with previously done work on this topic.

5.2 Dataset Splits

The dataset is split into 75 percent training data and 25 percent test data. We set the random state value 0 of sci-kit learn's the model selection function train\_test\_split() to get the same training and test set all time.

Number of question pairs in the training dataset: **303261**

Number of question pairs in the testing dataset: **101087**

5.3 Experimental Environments

We have used Google Colab to train the model. Google Colab is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs [57].

Most of the time I got Google Colab Hardware specs as follows

Table 5.1: Google Colab Hardware specs

|  |  |
| --- | --- |
| CPU | 4 Intel(R) Xeon(R) CPU @ 2.20GHz |
| Number of Core in each processor | 2 |
| GPU | Tesla P100-PCIE-16GB |
| RAM | 25.51 GB |
| Disk | 68.40 GB |

5.4 Experiment Results

At first, we will show all the different model results separately then we will find out best performer model to detect duplicate question pairs.

5.4.1 The Siamese MaLSTM Model

We have implemented 4 different versions of this model. Among these four versions, the only differentiating factor is the embedding layer of the Siamese MaLSTM model.

1. Siamese MaLSTM model with Keras Embedding

Experiment results for the Siamese MaLSTM model with Keras embedding layer with 128 embedding dimensions.

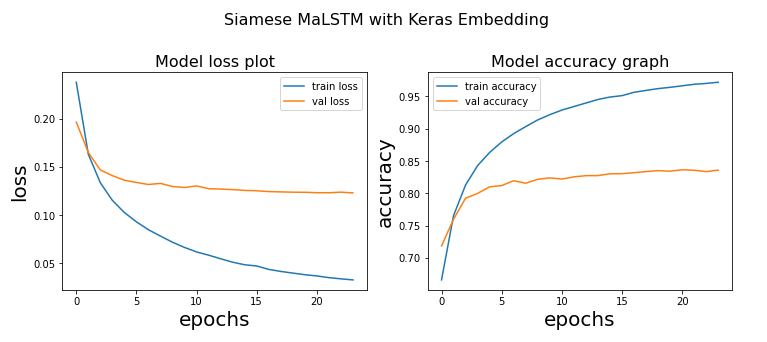


Figure 5.1: Loss and Accuracy plot of the Siamese MaLSTM model

with Keras embedding

We got 83.63 percent accuracy to detect duplicate question pairs using the Siamese MaLSTM model with Keras embedding layer.

1. Siamese MaLSTM model with Google News Vector Embedding

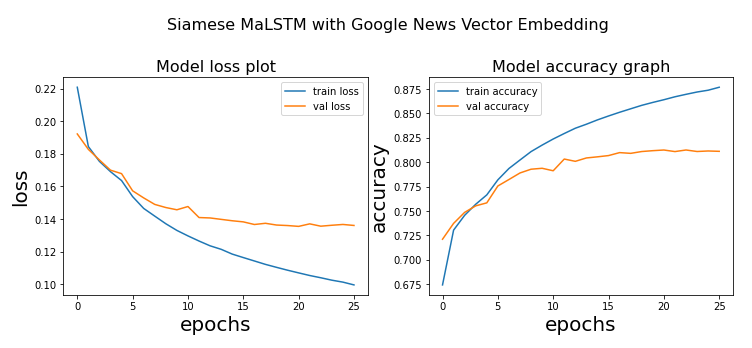


Figure 5.2: Loss and Accuracy plot of the Siamese MaLSTM model

with Google News Vector Embedding

We got 81.24 percent accuracy to detect duplicate question pairs using the Siamese MaLSTM model with Google News Vector Embedding.

1. Siamese MaLSTM model with GloVe Common Crawl Uncased Embedding

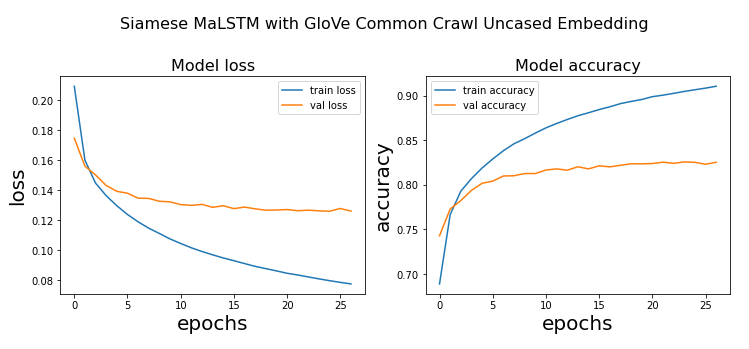


Figure 5.3: Loss and Accuracy plot of the Siamese MaLSTM model

with Glove Common Crawl uncased embedding

We got 82.54 percent accuracy to detect duplicate question pairs using the Siamese MaLSTM model with Glove Common Crawl uncased embedding.

1. Siamese MaLSTM model with FastText Common Crawl Embedding

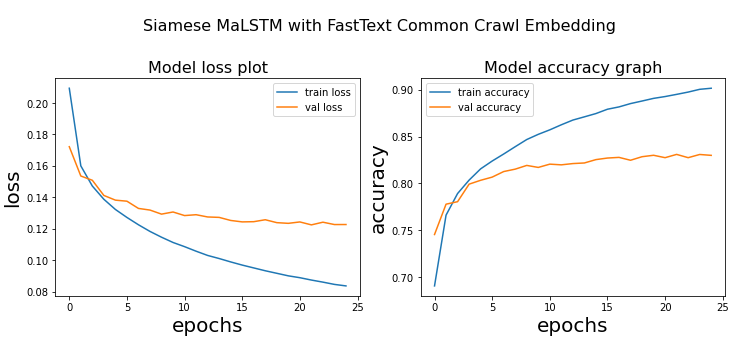


Figure 5.4: Loss and Accuracy plot of the Siamese MaLSTM model

with FastText Common Crawl Embedding

We got 83.10 percent accuracy to detect duplicate question pairs using the Siamese MaLSTM model with FastText Common Crawl Embedding.

5.4.2 The BERT Bi-LSTM Model

We got 90.40 percent accuracy to detect duplicate question pairs using the BERT Bi-LSTM model.

We also tried another BERT based model called BERT for Sequence Classification and got 90.50% accuracy to detect duplicate question pairs.

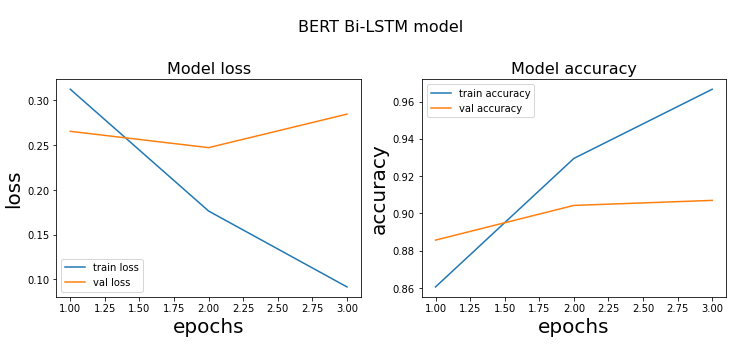


Figure 5.5: Loss and Accuracy plot of the BERT Bi-LSTM model

5.4.3 The Siamese BERT Model

We got 89.40 percent accuracy to detect duplicate question pairs using the Siamese BERT model.

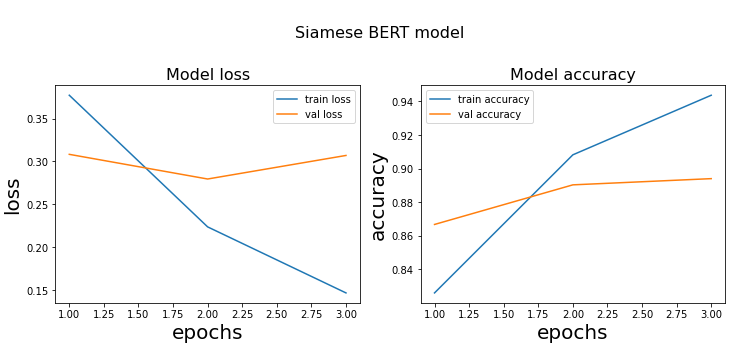
****

Figure 5.5: Loss and Accuracy plot of the Siamese BERT model

5.4.4 The RoBERTa Bi-LSTM Model

Now we will present the results of our best performer model to detect duplicate question pairs.

We got 91.20 percent accuracy to detect duplicate question pairs using the RoBERTa Bi-LSTM model.

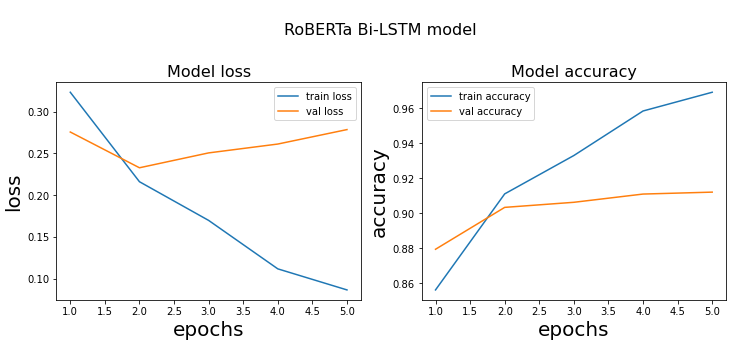


Figure 5.5: Loss and Accuracy plot of the RoBERTa Bi-LSTM model

We also tried another RoBERTa based model called RoBERTa for Sequence Classification and got 90.85% accuracy to detect duplicate question pairs.

5.5 Results Analysis

All of our implemented model results are given in Table 5.3. For the Siamese manhattan LSTM model, we get the best results using the Keras embedding layer with 128 dimensions. The reason behind of this performance that the Quora Questions Pair dataset contains too many misspelled words. Those misspelled words are not present in pre-trained word embeddings. That's why results of the pre-trained word embedding models fall behind the Keras continuously learnable embedding layer.

Table 5.2: Results of our implemented models.

|  |  |
| --- | --- |
| **Model** | **Model Accuracy** |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Siamese MaLSTM | |  | | --- | | Keras Embedding | | Google News Vector Embedding | | GloVe Common Crawl Embedding | | FastText Common Crawl Embedding | | | |  | | --- | | 83.63% | | 81.24% | | 82.54% | | 83.10% | |
| The BERT Bi-LSTM model | 90.40% |
| The BERT base cased for Sequence Classification model | 90.50% |
| The Siamese BERT model | 89.40% |
| **The RoBERTa Bi-LSTM model** | 91.20% |
| The RoBERTa base for Sequence Classification model | 90.85% |

For the transfer learning-based approach, We got the best results from the RoBERTa Bilstm model. The RoBERTa based model performs slightly better than the BERT based model with the same model architecture. The reason behind that the Robustly Optimized BERT approach is pre-trained on more data than BERT and the hyper parameters are more optimized than BERT [44].

5.5 Comparison with previous works

In the literature review section, we have mentioned all the previous works with their strategies. Now we will compare our best performer model results with the previous works on the duplicate question pairs detection techniques.

Table 5.3 shows the comparisons between our works and all the previous works on this topic. From the comparison, we can see that our RoBERTa Bi-LSTM model performs better than all other previous works on the duplicate question pair detection on the Quora Questions Pair dataset.

Table 5.3: Comparison with previous works

|  |  |
| --- | --- |
| Paper | Accuracy |
| Detecting duplicate questions with deep learning [28] | 85.0% |
| An enhanced deep learning model for duplicate question pairs recognition [29] | 88.17% |
| Bilateral multi-perspective matching for natural language sentences [12] | 80.40% |
| Duplicate Questions Pair Detection Using Siamese MaLSTM [6] | 91.14% |
| Natural language understanding with the quora question pairs dataset [8] | 83.4% |
| Our RoBERTa Bi-LSTM model | 91.20% |

5.6 Conclusion

It can be concluded that the Siamese MaLSTM model performance is lower than the state of the art models like BERT and RoBERTa to detect duplicate question pairs on the Quora Questions Pair dataset. The Siamese BERT model performs almost similar to the BERT model. The RoBERTa Bi-LSTM performs better than all of our strategies.

**CHAPTER 6**

**Conclusion and Future Scopes**

6.1 Introduction

This chapter summarizes the whole research in several words describing the problem domain, previous researches, our contribution, experimental analysis and the outcomes of this research. Moreover, we have provided a brief study about the future scopes of this research.

6.2 Conclusion

Detection of duplicate questions from a corpus containing a pair of questions deals with identifying whether two questions in the pair convey the same meaning or not. Quora is a growing platform comprising a user generated collection of questions and answers. The questions and answers are created, edited, and organized by the users. Enormous number of users on the Quora website makes it unavoidable to have multiple questions from different users with similar intent, which raises the issue of duplicate questions. Effectively detecting duplicate questions would make it easier to find high quality answers and help save time, which in turn would result in an improved user experience for writers and readers on Quora.

Previous studies revealed that duplicate questions could be identified by SVM, SVM with n-grams, random forest, a continuous bag of words, LSTM, LSTM with attention, Manhattan LSTM, Siamese MaLSTM, Decision Tree, and XGBoost, etc. The previous highest accuracy was 91.14%. We achieved higher accuracy than previous works.

Our experiments concluded that the Siamese MaLSTM model with Keras embedding works better than other pre-trained word embeddings(Google News Vector, GloVe Common Crawl, FastText Common Crawl) because the Quora Questions Pair dataset contains too many misspelled words. Those misspelled words are not present in pre-trained word embeddings. That's why results of the pre-trained word embedding models fall behind the Keras continuously learnable embedding layer.

The Siamese MaLSTM model performance is lower than the state of the art models like BERT and RoBERTa to detect duplicate question pairs on the Quora Questions Pair dataset. The Siamese BERT model performs almost similar to the BERT model. The RoBERTa Bi-LSTM performs better than all of our strategies.

From our studies, it can be concluded that the RoBERTa Bi-LSTM model is the best to detect duplicate question pairs for the Quora Questions Pair dataset.

6.3 Limitations and Future Scopes

The Quora Questions Pair dataset is an imbalanced dataset. The number of duplicate question pairs is very less compared to non-duplicate question pairs. So to get better performance, we should solve the dataset imbalance problem but we didn’t solve this problem. This is a limitation.

Another limitation is the misspelled word processing. How can we get rid of the misspelled words problem? We didn't find any ideas to solve this problem.

In the future, we will try to fix the above two limitations imbalanced dataset and misspelled word correction. We also implement the Siamese RoBERTa approach to detect duplicate question pairs.

# **References**

|  |  |
| --- | --- |
| [1] | Hoogeveen, Doris and Wang, Li and Baldwin, Timothy and Verspoor, Karin M, "Web forum retrieval and text analytics: A survey," *Foundations and Trends in Information Retrieval,* vol. 12, pp. 1-163, 2018. |
| [2] | Shah, Darsh J and Lei, Tao and Moschitti, Alessandro and Romeo, Salvatore and Nakov, Preslav, "Adversarial domain adaptation for duplicate question detection," *arXiv preprint arXiv:1809.02255,* 2018. |
| [3] | Bromley, Jane and Bentz, James W and Bottou, L{\'e}on and Guyon, Isabelle and LeCun, Yann and Moore, Cliff and S{\"a}ckinger, Eduard and Shah, Roopak, "Signature verification using a “siamese” time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 7, pp. 669-688, 1993. |
| [4] | Sanborn, Adrian and Skryzalin, Jacek, "Deep learning for semantic similarity," *CS224d: Deep Learning for Natural Language Processing Stanford, CA, USA: Stanford University,* 2015. |
| [5] | Quora, "Quora Question Pairs," *Kaggle,* pp. Accessed: 01-12-2020. [Online]. Available: https://www.kaggle.com/c/quora-question-pairs, 2017. |
| [6] | Imtiaz, Zainab and Umer, Muhammad and Ahmad, Muhammad and Ullah, Saleem and Choi, Gyu Sang and Mehmood, Arif, "Duplicate Questions Pair Detection Using Siamese MaLSTM," *IEEE Access,* vol. 8, pp. 21932-21942, 2020. |
| [7] | Chen, Zihan and Zhang, Hongbo and Zhang, Xiaoji and Zhao, Leqi, "Quora question pairs," *Quora,* 2018. |
| [8] | Sharma, Lakshay and Graesser, Laura and Nangia, Nikita and Evci, Utku, "Natural language understanding with the quora question pairs dataset," *arXiv preprint arXiv:1907.01041,* 2019. |
| [9] | Dey, Kuntal and Shrivastava, Ritvik and Kaushik, Saroj, "A paraphrase and semantic similarity detection system for user generated short-text content on microblogs," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2880-2890. |
| [10] | Mueller, Jonas and Thyagarajan, Aditya, "Siamese recurrent architectures for learning sentence similarity," in *thirtieth AAAI conference on artificial intelligence*, 2016. |
| [11] | Zhu, Wenhao and Yao, Tengjun and Ni, Jianyue and Wei, Baogang and Lu, Zhiguo, "Dependency-based Siamese long short-term memory network for learning sentence representations," *PloS one,* vol. 13, p. e0193919, 2018. |
| [12] | Wang, Zhiguo and Hamza, Wael and Florian, Radu, "Bilateral multi-perspective matching for natural language sentences," *arXiv preprint arXiv:1702.03814,* 2017. |
| [13] | Patro, Badri N and Kurmi, Vinod K and Kumar, Sandeep and Namboodiri, Vinay P, "Learning semantic sentence embeddings using sequential pair-wise discriminator," *arXiv preprint arXiv:1806.00807,* 2018. |
| [14] | Tsubaki, Masashi and Duh, Kevin and Shimbo, Masashi and Matsumoto, Yuji, "Non-Linear Similarity Learning for Compositionality," *AAAI,* pp. 2828-2834, 2016. |
| [15] | B. a. P. K. a. C. K. a. W. W. a. A. P. Rychalska, "Samsung Poland NLP Team at SemEval-2016 Task 1: Necessity for diversity; combining recursive autoencoders, WordNet and ensemble methods to measure semantic similarity," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 602--608. |
| [16] | Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems,* vol. 26, pp. 3111--3119, 2013. |
| [17] | Sravanthi, Pantulkar and Srinivasu, B, "Semantic similarity between sentences," *International Research Journal of Engineering and Technology (IRJET),* vol. 4, pp. 156--161, 2017. |
| [18] | Johnson, Rie and Zhang, Tong, "Supervised and semi-supervised text categorization using LSTM for region embeddings," *arXiv preprint arXiv:1602.02373,* 2016. |
| [19] | Tang, Duyu and Qin, Bing and Liu, Ting, "Document modeling with gated recurrent neural network for sentiment classification}," in *roceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422--1432. |
| [20] | Zhou, Chunting and Sun, Chonglin and Liu, Zhiyuan and Lau, Francis, "A C-LSTM neural network for text classification," *arXiv preprint arXiv:1511.08630,* 2015. |
| [21] | Tai, Kai Sheng and Socher, Richard and Manning, Christopher D, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075,* 2015. |
| [22] | Kiros, Ryan and Zhu, Yukun and Salakhutdinov, Russ R and Zemel, Richard and Urtasun, Raquel and Torralba, Antonio and Fidler, Sanja, "Skip-thought vectors," *Advances in neural information processing systems,* vol. 28, pp. 3294--3302, 2015. |
| [23] | Yu, Lei and Hermann, Karl Moritz and Blunsom, Phil and Pulman, Stephen, "Deep learning for answer sentence selection," *arXiv preprint arXiv:1412.1632,* 2014. |
| [24] | Kim, Yoon, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882,* 2014. |
| [25] | Collobert, Ronan and Weston, Jason and Bottou, L{\'e}on and Karlen, Michael and Kavukcuoglu, Koray and Kuksa, Pavel, "Natural language processing (almost) from scratch," *Journal of machine learning research,* vol. 12, pp. 2493--2537, 2011. |
| [26] | He, Hua and Gimpel, Kevin and Lin, Jimmy, "Multi-perspective sentence similarity modeling with convolutional neural networks," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1576--1586. |
| [27] | Shih, Chin-Hong and Yan, Bi-Cheng and Liu, Shih-Hung and Chen, Berlin, "Investigating Siamese LSTM networks for text categorization," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017, pp. 641--646. |
| [28] | Homma, Yushi and Sy, Stuart and Yeh, Christopher, "Detecting duplicate questions with deep learning," *Proceedings of the International Conference on Neural Information Processing Systems (NIPS),* 2016. |
| [29] | Abishek, K and Hariharan, Basuthkar Rajaram and Valliyammai, C, "An enhanced deep learning model for duplicate question pairs recognition," in *Soft Computing in Data Analytics*, Springer, 2019, pp. 769--777. |
| [30] | Chicco, Davide, "Siamese neural networks: An overview," *Artificial Neural Networks,* no. Springer, pp. 73--94, 2020. |
| [31] | E. Shrestha, "Learning Similarity with Siamese Neural Networks," *Medium,* pp. Available: https://medium.com/@enoshshr/learning-similarity-with-siamese-neural-networks-51c9ef534ae4, 2020. |
| [32] | Hochreiter, Sepp and Schmidhuber, J{\"u}rgen, "Long short-term memory," *Neural computation,* vol. 9, pp. 1735--1780, 1997. |
| [33] | colah, "Understanding LSTM Networks," *colah's blog,* pp. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1, 2015. |
| [34] | Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805,* 2018. |
| [35] | Jacob, Devlin, Ming-Wei, Chang, "Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing," *Google AI Blog,* 2018. |
| [36] | P. Nayak, "Understanding searches better than ever before," *Google,* Retrieved 8-12-2020. |
| [37] | Phang, Jason and F{\'e}vry, Thibault and Bowman, Samuel R, "Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks," *arXiv preprint arXiv:1811.01088,* 2018. |
| [38] | R. Horev, "BERT Explained: State of the art language model for NLP," *Towards Data Science,* pp. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270, 2018. |
| [39] | S. Karla, "BERT Language Model," *Medium,* pp. Available: https://medium.com/@shreyasikalra25/predict-movie-reviews-with-bert-88d8b79f5718, 2019. |
| [40] | Taylor, Wilson L, "“Cloze procedure”: A new tool for measuring readability," *Journalism quarterly,* vol. 4, pp. 415--433, 1953. |
| [41] | Vincent, Pascal and Larochelle, Hugo and Bengio, Yoshua and Manzagol, Pierre-Antoine, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096--1103. |
| [42] | Zhu, Yukun and Kiros, Ryan and Zemel, Rich and Salakhutdinov, Ruslan and Urtasun, Raquel and Torralba, Antonio and Fidler, Sanja, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," *Proceedings of the IEEE international conference on computer vision,* pp. 19--27, 2015. |
| [43] | Chelba, Ciprian and Mikolov, Tomas and Schuster, Mike and Ge, Qi and Brants, Thorsten and Koehn, Phillipp and Robinson, Tony, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005,* 2013. |
| [44] | Liu, Yinhan and Ott, Myle and Goyal, Naman and Du, Jingfei and Joshi, Mandar and Chen, Danqi and Levy, Omer and Lewis, Mike and Zettlemoyer, Luke and Stoyanov, Veselin, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692,* 2019. |
| [45] | Han, Jun and Moraga, Claudio, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Workshop on Artificial Neural Networks*, Springer, 1195, pp. 195--201. |
| [46] | Wikipedia, "Sigmoid function," *Wikipedia,* pp. Accessed: 7-10-12. [Online]. Available: https://en.wikipedia.org/wiki/Sigmoid\_function. |
| [47] | S. I. Serengil, "Hyperbolic Tangent as Neural Network Activation Function," pp. Accessed: 8-12-2020. [Online]. Available: https://sefiks.com/2017/01/29/hyperbolic-tangent-as-neural-network-activation-function/, 2017. |
| [48] | K. Sarkar, "ReLU: Not a Differentiable Function," *Medium,* pp. Accessed: 8-10-2020. [Online]. Available: https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec. |
| [49] | Koksoy, Onur, "Multiresponse robust design: Mean square error (MSE) criterion," *Applied Mathematics and Computation,* vol. 175, pp. 1716--1729, 2006. |
| [50] | Zhang, Zhilu and Sabuncu, Mert, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems,* vol. 31, pp. 8778--8788, 2018. |
| [51] | H. Zulkifli, "Understanding Learning Rates and How It Improves Performance in Deep Learning," *Towards Data Science,* pp. Accessed: 8-12-2020. [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10, 2018. |
| [52] | Kingma, Diederik P and Ba, Jimmy, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014. |
| [53] | Heusel, Martin and Ramsauer, Hubert and Unterthiner, Thomas and Nessler, Bernhard and Hochreiter, Sepp, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," *Advances in Neural Information Processing Systems,* vol. 30, pp. 6626--6637, 2017. |
| [54] | Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, pp. 1929--1958, 2014. |
| [55] | Wang, Siyue and Wang, Xiao and Zhao, Pu and Wen, Wujie and Kaeli, David and Chin, Peter and Lin, Xue, "Defensive dropout for hardening deep neural networks under adversarial attacks," *Proceedings of the International Conference on Computer-Aided Design,* pp. 1--8, 2018. |
| [56] | Reimers, Nils and Gurevych, Iryna, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084,* 2019. |
| [57] | Bisong, Ekaba, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, 2019, pp. 59--64. |