Duplicated Questions Classification Using Machine Learning

Nalinrat Laekawan
Paichana Kularb
Phuriphan Julsirikul

Bachelor of Engineering Program in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2017
KMITL-2018-IC-B-003-009

Final Report - academic year 2017

Bachelor of Engineer in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang

Title - Duplicated Questions Classification Using Machine Learning

Authors:

57090010 Nalinrat Laekawan
57090015 Paichana Kularb
57090022 Phuriphan Julsirikul

.....
Asst.Prof.Dr. Chaiwat Nuthong
Advisor
Date: 15/06/2018

Abstract

Natural Language Processing(NLP) is a fairly new topic and is yet to mature. There are many approaches to solving NLP problems, and different approaches suits to different problems. One of the approaches that is becoming more popular is to use machine learning to solve NLP problems. It is very popular due to the ability to dealt with a large amount of data.

Currently, there are many questions posted in public forums on a daily basis. The process to identify questions with the same intent is done by human which is a slow process. Some questions might not be answered and are overlooked due to the large amount of questions.

Neural network algorithms which is a subtype of machine learning is used in this project due to its ability to handle large amount of data. The experiments is conducted using Quora's labeled dataset to train and evaluate the model. In this project, the effectiveness of Siamese Architecture and LSTM model in classifying duplicated questions will be evaluated. The results have shown that Siamese Architecture reduces overfitting and using Bi-Directional with Siamese Architecture result in a highest prediction power.

Acknowledgement

This project could not be completed if we do not great advices from our advisor, Asst.Prof.Dr. Chaiwat Nuthong. We also would like to thank Dr. Isara Anatavrasilp and my colleagues for giving suggestions in presentation and useful discussions. Moreover, we are thankful to our lecturers at International College for providing support and knowledge. Additionally, we would like to give special thanks to our families for their support, motivations, and encouragement. However, for those who are not mentioned above still play an important supporting for us. The project cannot be completed and get these results without their cooperations.

Contents

Chapter 1

Introduction

1.1 Background

There are thousands of questions posted on public forums that have not been answered. However, the answers to these many questions already exist. By identifying questions with the same intent, the forum group's existed one corresponding answer can be given to satisfy such questions. In public forums such as Quora, the questions may describe the same problem which refers to them as duplicated questions. Figure 1.1 shows an example of the duplicated questions which represent the same problem about waking up early in the morning. Although users can select the tag categories of the problems, the unidentified duplicated question still exist. This can cause the difficulty in maintenance these forum. Furthermore, efforts are also regarded to answer such redundant questions.

The current method of defining the duplicated question is manually performed by humans. Since there are a lot of questions posted on the public forums, some questions might not be taken care of. The process also shows slow and not automatic response. To help solve these kinds of problems, there are two approaches that will be proposed in this work, i.e. Machine Learning and Natural Language Processing (NLP).

1.2 Problem Descriptions

Numerous amount of duplicate questions on public forums are posted on a daily basis. On the contrary, use the existing system to identify and merge the same questions is inefficient. The current process of, public forum, e.g. Quora requires reports from the users. This report will then be reviewed by

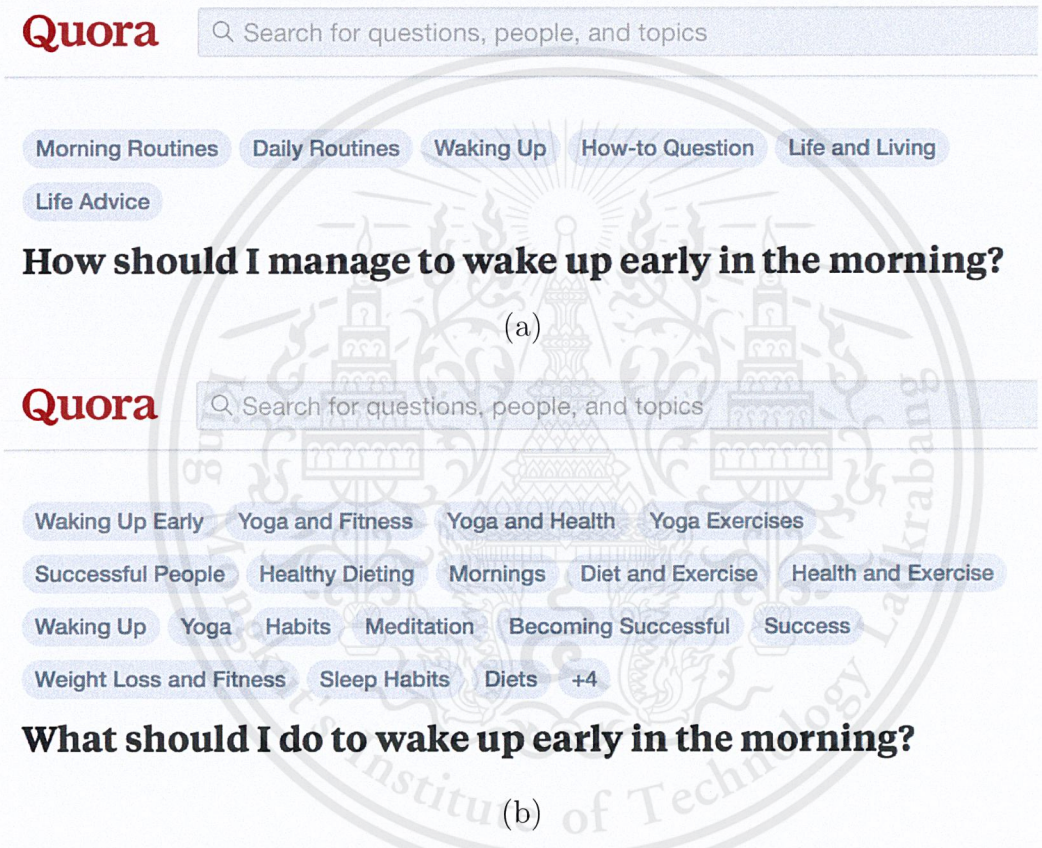

Figure 1.1: Two duplicated questions from Quora.

Quora in order to determine whether to merge the questions or not. This process is done manually which is obviously not efficient.

In order to deal with such problem, Natural Language Processing (NLP) is the field of processing human natural language and is the one the authors need for the problem. Despite the rising trend, NLP is still growing and is not mature enough to be used reliably. More research is still needed for the future improvement of NLP. This thesis focuses on applying and comparing performances between different neural network architecture such as LSTM and Feed Forward on classifying duplicate question provided by Quora. The research will mainly focus on classifying general question.

1.3 Objectives

In order to indicate the success of the project, several goals have been set. The following list shows all the goals which needed to be satisfied.

- Apply machine learning algorithms to classify duplicate questions of Quora's dataset.
- Compare performances of each type of applied Neural Network algorithm.
- Develop a working software to allow users to input the question pairs to check whether the model thinks the questions are duplicated.

1.4 Contributions

The expected contribution of this study is to improve the ability to classify whether two questions have the same intent. In particular, this study applies various types of Neural Network to Quora's dataset and compares the performance of each type of Neural Network algorithm. The problem is also similar to paraphrase detection and could be adapted accordingly.

1.5 Scope of Work

This research focuses on applying machine learning algorithm to natural language processing in order to classify whether pairs of questions have the same purpose. The result of this research would be measured in term of

performance measurement and computational time. The train and test process will be done on the Quora's labeled dataset provided by Quora. More specifically, the entire dataset is in English.(https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs)

Chapter 2

Literature Review

There are various kind of studies and researches that are related to this project. This mainly involves Machine Learning and Natural Language Processing.

2.1 Natural Language Processing

Natural Language Processing (NLP) is one of the fields of computer science specializing in the interaction between computer and human through natural language. Example tasks of NLP include but not limited to sentiment analysis, speech recognition, and machine translation (e.g. Google Translate)[1]. Human language is often ambiguous with the linguistic structure that can depend on many other variables including slang, regional dialects, and social context[2]. There are five common steps in NLP, lexical analysis, syntactic analysis, semantic analysis, disclosure integration, and pragmatic analysis, but only three steps will be discussed in detail in the following subsection[3].

2.1.1 Lexical Analysis

Lexical analysis is a process of converting a sequence of characters into words, phrases, or meaningful elements called tokens. Unmeaningful tokens can be thrown away (e.g. stopwords) to reduce noises in the data. This process is for the language that is non-segmenting. There are many methods in doing lexical analysis such as longest matching, n-gram matching, and part-of-speech matching. The ambiguous message will be avoided by tokenizing the text[4]. This method is also claimed to be used in sentiment analysis to have a better result[5].

2.1.2 Syntactic Analysis

The purpose of syntactic analysis is to understand the sentence by analyzing the words and arranging of words in the sentence. The words would be analyzed based on the grammatical rule. In this step, it shows the relationship of how the words relate to each other. However, some sentence might be rejected because of grammatical violation. Part of speech tagging (POS) is used in this step to assign each word into English's syntax. Based on the past study of A. Voutilainen.[6], the author claims that using POS with some approaches such as taggers based on n-grams automatically derived from the tagged text corpus, and taggers based on Hidden Markov Models, tend to increase the accuracy in syntactic analysis. Another study proposed by H. Schmid [7], shows a couple of extensions to a Markov Model tagger called TreeTagger. The accuracy is improved when the model is trained using a small corpus, and also the extension itself helps reduce the error rate.

2.1.3 Semantic Analysis

This step is a process of extracting relevant and useful information from large bodies of unstructured data or in other words. To find a meaning in the text data, there is a study of using semantic analysis. R. Schank's research[8] classified objects according to their semantic categories which were used as a guideline in the system to create a clear description of the language input.

2.2 Text Normalization

Text normalization is done to create a uniform representation of words with the same meaning.

2.2.1 Text Normalization for Paraphrase Detection

The text processing is done on the Agency for Toxic Substances and Disease Registry dataset which is about describing toxic products. The text in the dataset is reduced to a uniform representation of words[9]. For example, PHYS_FORM(ammonia, gas) to indicate that ammonia is a gas. Most of the normalization is used to suit this specific domain. The normalization improves the performance of the model drastically.

2.2.2 Text Normalization in Casual English

E.Clarka and K. Arakia[10] developed a system that could correct casual english words in social medias. The incorrect english includes, abbreviations (*'lol'* to *'laugh out loud'*), misspell(*'wouls'* to *'would'*, punctuation error(*'im'* to *'i'm*), wordplay(*'Sooooo good'* to *'So good'*), censored words(*'sh1t f***'*) and emoticons(*';3'* to *'heart'*). The method is to hard coded the rules in the database. The result from the evaluation shows the average error per sentence reduced by roughly 10 percent.

2.2.3 Text Normalization in Twitter messages

The dataset used in the research is the Edinburgh Twitter Corpus which contain 97 million tweets. Orthographic Normalization and Syntactic Disambiguation are used to preprocess the data and then followed with machine translation. The normalization increased a precision measure based on count vector called BLEU score by 18% [11].

2.3 Machine Learning

Machine Learning is the process of giving the machine the ability to carry out the task without explicitly programmed [12]. It is a part of Artificial Intelligence (AI) and closely related to computational statistic and data mining. The machine learning can be achieved with the tremendous amount of methods available today, one of which is Neural Network(NN), the main focus of this research.

2.3.1 Convolution Neural Network(CNN)

W. Yin and H. Schutze[13] used Bi-CNN-MI. This a double CNN used in Siamese framework that contains two identical subnetworks. The model computes representation at 4 levels which are a word, short n-gram, long n-gram, and sentence. The network is trained using the Microsoft Research Paraphrase(MSRP) corpus and English Gigaword. The evaluation of the MSRP dataset shows a greater performance than other NN approaches such as Neural Language Model(NLM) and Recursive Auto Encoder(ARC).

2.3.2 Feedforward Neural Network

Feedforward neural network is a type of an artificial neural network where the connections between the layers do not form a cycle. The information

flows only one way. This means that after the process finishes computation in a layer, it moves to the next tier of the layer and has no ways of going back to the previous tier. It's the simplest type of artificial neural network and is the first to be devised [14].

A group of researchers from Google has implemented a simple feed forward network to perform common NLP tasks. They have tested on 4 different topics which are language identification, part-of-speech tagging, word segmentation, and pre-ordering for translation. The conclusion was that complex networks perform better than the simple feedforward. However, the simple feedforward network is more efficient in both memory and time while achieving close to state-of-the-art results[15].

2.3.3 Long-Short Term Memory(LSTM) Neural Network

LSTM Neural Network is a type of the recurrent neural networks that are designed to avoid longterm dependency problems. The network achieves this by having the ability to decide what information to forget, what information to keep and what information is going to the output[16].

Detecting question pairs using CNNs, LSTMs, and a hybrid model were compared by T. Addair[17]. The result shows that LSTM has the highest accuracy of 81.07 % when compared with other models. E. Dadashov, S. Sakshuwong, and K. Yu[18] studied two approaches based on LSTM network. Siamese with LSTM and two LSTMs networks were compared in this study. The single LSTM is used to produce the output vectors. They are then fed into the model. The first approach achieved 83.2 % accuracy where the other approach is 80.8 %. When these two techniques were ensembled, the accuracy becomes 83.8 % on the test set.

Chapter 3

Background Knowledge

In this chapter, the background concepts that are related to this study will be briefly explained.

3.1 Word2Vec

Word2vec is a predictive model for learning word embeddings from raw text. It is a shallow neural model with a single hidden layer which will give out word vectors from texts. Word vectors are a numerical representation of each word which positioned in the vector space such that words that share common contexts in the corpus are located near each other in the space.

The word vectors extracted in this way will preserve the relationship with other words through the position it is embedded in the vector space. In the vector space relationships of words can be calculated using similarity measures such as Cosine Similarity. Patterns such as "Man is to Woman as Brother is to Sister" can be learned such that the vector representation of "Brother" - "Man" + "Woman" will produce the word vector close to the representation vector of the word "Sister". Word2Vec has two model architecture, i.e. Continuous bag of words and Skip-Gram.

The vector that represent relationships between words is computed by using the weights of a single hidden layer neural network. In practical, The neurons in the hidden layer are all linear neurons. Each unique words in the corpus will act as a neuron in the input layer. So, the input layer can be represented by matrix $\boldsymbol{W}$ of size $V \times N$, encode with one hot encoding, where V is the number of words in the vocabulary that come from extracting unique words in the corpus and N is the number of dimensions choose to represent

our word in. This will also be the number of neurons in the hidden layer. The connections from hidden layer to output layer then can be described by matrix $\boldsymbol{W'}$ of size $N \times V$ with each column of $\boldsymbol{W'}$ matrix represents a word from the given vocabulary, noted that the matrix $\boldsymbol{W'}$ is not the transpose of matrix $\boldsymbol{W}$. The input vector, in this case, will be the row of matrix $\boldsymbol{W}$ and the output vector will be the column of matrix $\boldsymbol{W'}$ with each vector representing one word in the vocabulary. The weights in the matrices WO and WI can be updated using back-propagation, after numerous word pairs in the sentence has been feed into the model [19].

The goal of this model is to produce probabilities for words in the output layer. This process is done in order to reflect their next word relationship with the context word as the input with the sum of neuron outputs in the output layer to add to one. Thus, the Softmax function which is a function that scales a vector of arbitrary real values will be used as an actication function in range of 0 to 1. The output of k-th neuron will be computed as shown in Equation 3.1. Where $actv(h)$ and $actv(n)$ are the activation value of the k-th neuron and n-th output layer neuron respectively and V is the number of word in the vocabulary.

$$y_k = P(word_n|word_{context}) = \frac{e^{actv(h)}}{\sum_{n=1}^{V} e^{actv(n)}} \tag{3.1}$$

The two distinct model architectures utilize this approach in different ways with main difference lies in the selected context word which is explained in details in the following section.

3.1.1 Continuous Bag of Words (CBOW)

Continuous Bag of Words architecture is the Word2Vec model architecture that will try to predict the missing word given the context of the sentence. Words in this model is represented using the one-hot encoding method. Figure 3.1 describe simple CBOW model with four context word and a single target word. t in Figure 3.1 represent the position of the interested word. In this case, four surround words are used as the input and the center word is the target.

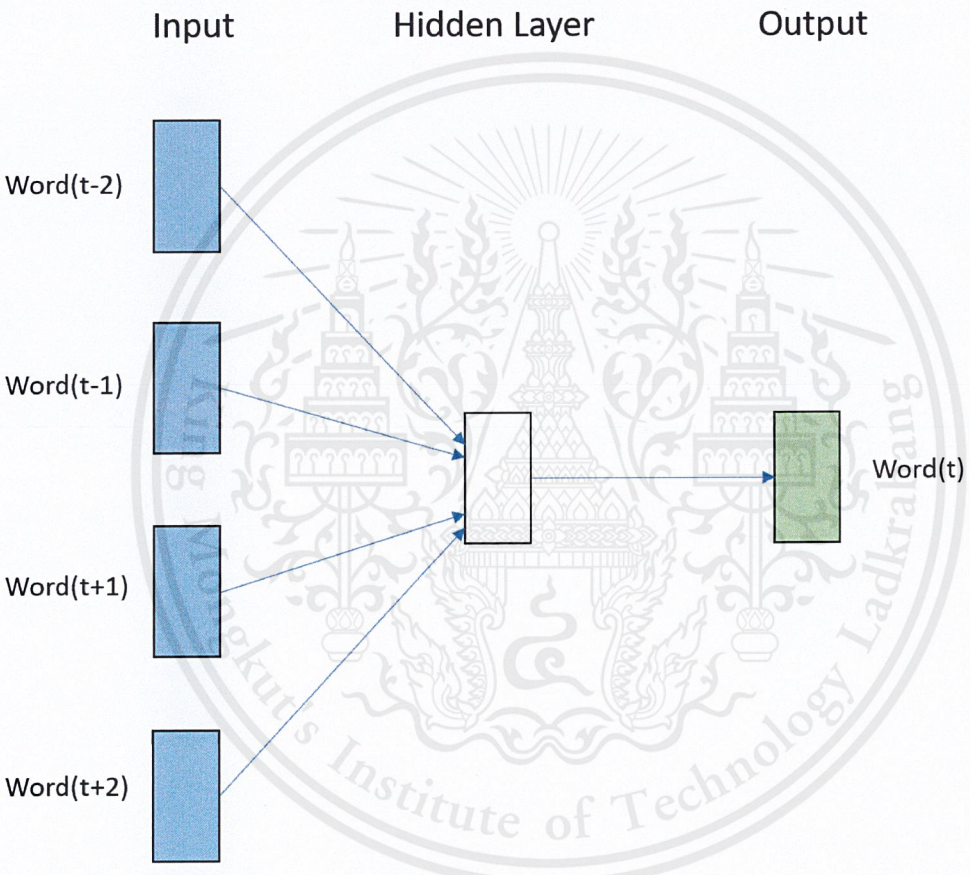

Figure 3.1: CBOW model with four context words

3.1.2 Skip-Gram

Skip-Gram model approach is the opposite to the CBOW model, with a target word is fed as an input and let the model predicts the context word of the target word[20]. The model will give out a probability of multiple words that likely to be associated with the target word. Figure 3.2 shows a Skip-gram model with four context words and a single target word. In this case the input of the model is the center word at index t and the output is the four surround context words.

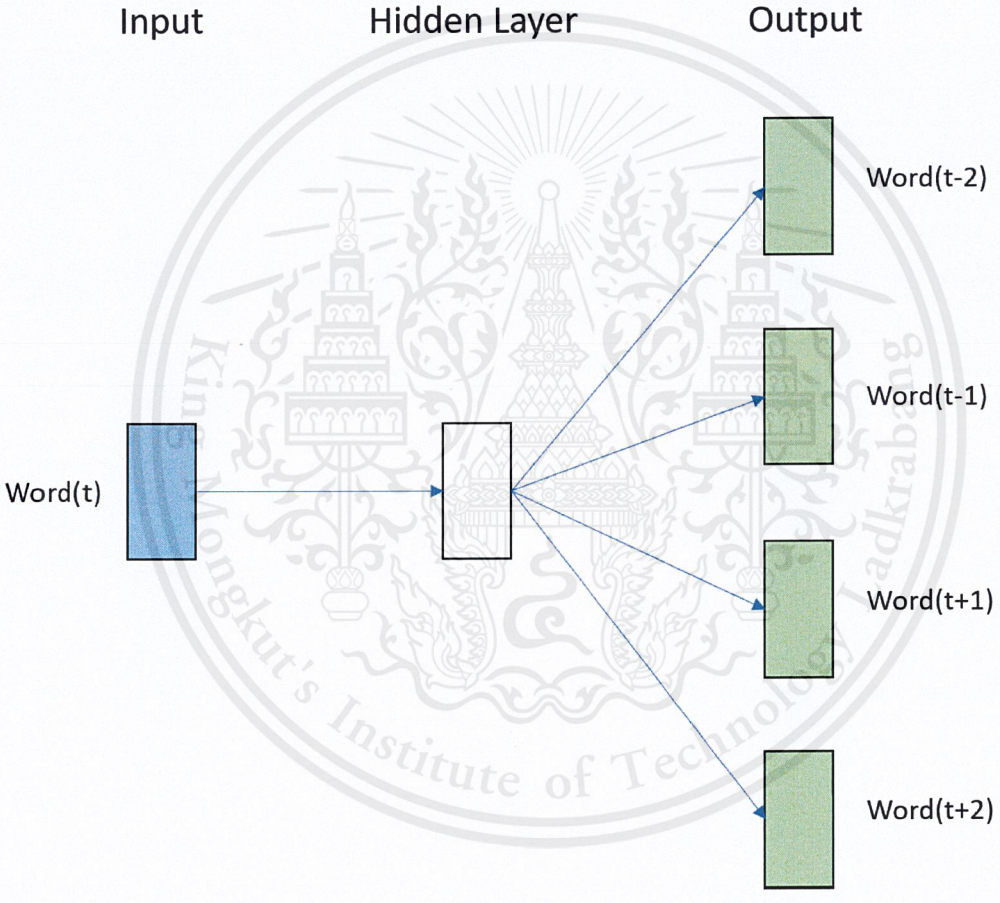

Figure 3.2: Skip-gram model with four context words

3.2 Latent Dirichlet Allocation(LDA)

Latent Dirichlet Allocation is a generative model that can randomly generate observed data. The idea of LDA is that the document is represented in a

mixture of topics where each topic has different words. It is a technique used to find a topic based on word frequency in a set of documents. To understand this algorithm, it can be divided into two probabilities[21]; Word-topic probabilities and Document-topic probabilities.

3.2.1 Word-Topic Probabilities

Word-topic probabilities are used to find the probabilities that words in a document d are assigned to a topic t. The probabilities can be found as shown in Equation3.2, where topic t is given in the document d

$$p_1 = P(topic\ t \mid document\ d) \tag{3.2}$$

3.2.2 Document-Topic Probabilities

The proportion of assignments to topic t in all documents which derived from the word w is called document-topic probabilities. The calculation is shown in Equation3.3, where the word w is given in the topic t.

$$p_2 = P(word\ w \mid topic\ t) \tag{3.3}$$

The word-topic assignment will be updated with a new topic by a probability of $P1 \times P2$, assuming all the existing word-topic assignments except the current word are correct.

3.3 Global Vectors

Global Vectors(GloVe) is an unsupervised learning algorithm that maps words into meaningful numerical vectors. The relationship between word vectors can be plot by using dimensionality reduction techniques as shown in Figure 3.3. The method claims that similarity measures such as Cosine Similarity or Euclidean Distance can be used to find semantic or linguistic similarities between the words. The vector from *women-men*, *aunt-uncle* and *queen-king* will be similar.

3.3.1 The Model

The model is trained by populating the word-word co-occurrence matrix with the frequency of word co-occurring in a context. The matrix does not include zero entries and only requires one pass through the corpus. The size of the context is a parameter where more weights are given to the closer

words. The author has proposed a new cost function as the Equation.3.4 [22].

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T \widetilde{w_j} + b_i + \widetilde{b_j} - logX_{ij})^2 \tag{3.4}$$

V = Vocabulary size

$f(X_{ij})$ = Weighting function

X_{ij} = Number of times word j occurs in context of word i in the Co-occurrence matrix

w_i = Word vector of i

$\widetilde{w}_j$ = Context word vectors of j

$\widetilde{b}_j$ = bias for $\widetilde{w}_j$

b_i = Bias for w_i

There are many weighting functions that could be used but the author of the paper has found this particular weighting function works well. It can be defined as shown in Equation 3.5

$$f(x) = \begin{cases} (x/x_{max})^{\frac{3}{4}}, & \text{if } x \leq 1 \\ 1, & \text{otherwise} \end{cases} \tag{3.5}$$

3.3.2 Model

The relation between word vectors is observed using the ratios of word-word co-occurrence probabilities as shown in Equation 3.6. The reason for using the ratio instead of the raw probability is because the property that does not distinguish the words are canceled out. Moreover, the terms that differentiate the words will get emphasized.

$$F(w_i, w_j, \widetilde{w_k}) = \frac{P_{ik}}{P_{jk}} \tag{3.6}$$

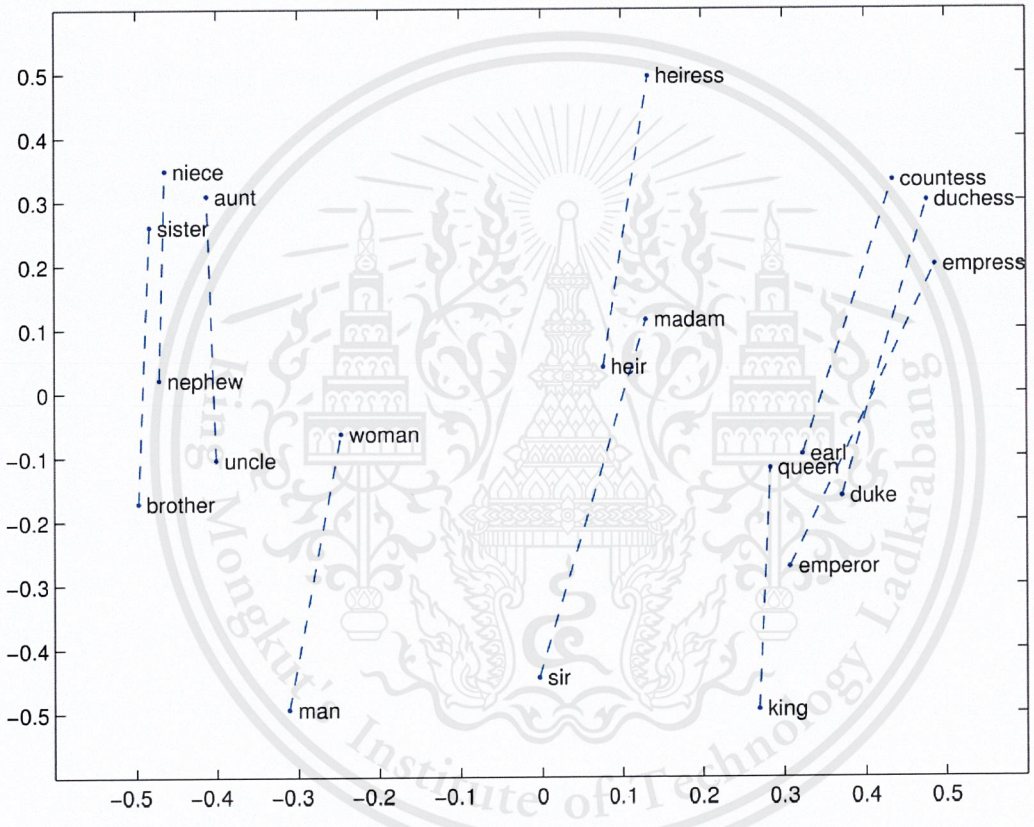

Figure 3.3: Relationships captured with GloVe

3.4 Learning Algorithm

3.4.1 Feedforward Neural Network Algorithm

A Feedforward Neural Network is the most basic type of neural network. The information only flows in one direction from the input layer towards the output layer. There is no cycle in the layers and the information does not pass through a single node twice. The network can be represented by a directed graph as shown in Figure 3.4 where nodes represent the neurons and arrows represent the links between them[23].

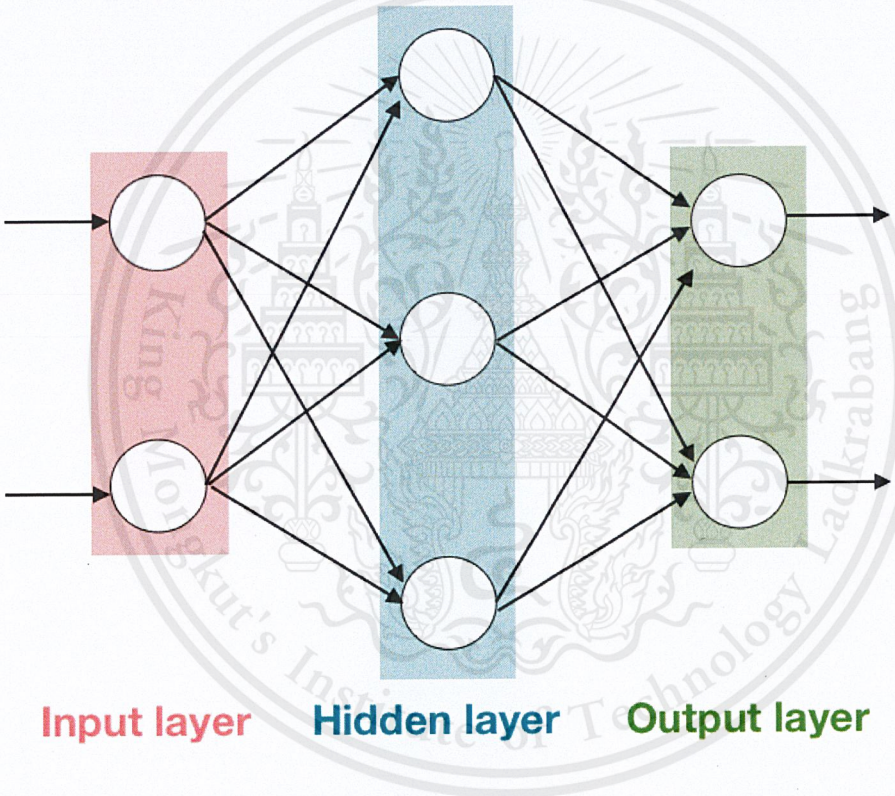

Figure 3.4: A directed graph of neural network with a single hidden layer.

Each node in a particular layer is connected to all nodes in the next layer, it is called fully connected layer. The connection between node i and node j is called weight coefficient ω_{ij} as shown in Figure3.5. The weight coefficient represents the importance of the given connection in the network.

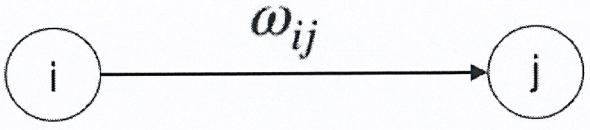

Figure 3.5: Connection between node i and j.

Feedforward network can be divided into two passes, the forward pass, and the backward pass. The forward pass is where the neural network takes inputs into calculation and traverse through the entire network resulting in a prediction. The backward pass is where the neural network updates its weights to improve its prediction. Normally the newly updated weights are calculated using gradient descent algorithm. The weight is updated from the last layer and traverse backward through the network.

The weight will be adjusted along the path in the network until the minimum cost is found during the learning phase. Modifying connection weight can be described by back-propagation network algorithm. The classification phase is used in fixing the amount of weight.

3.4.2 Long Short-Term Memory Network Algorithm

Long Short-Term Memory Network (LSTM) is an improved version of a general Recurrent Neural Network (RNN). It's designed to avoid the long-term dependency problem. Learning information over a long period in RNN can cause the information to be lost. The problem is called the vanishing gradient problem. The weights in the recurrent layer will be lower than intended and cause the network to stop learning. LSTM alleviate this problem by adding the ability to select which information to keep or discard through the recurrent step.

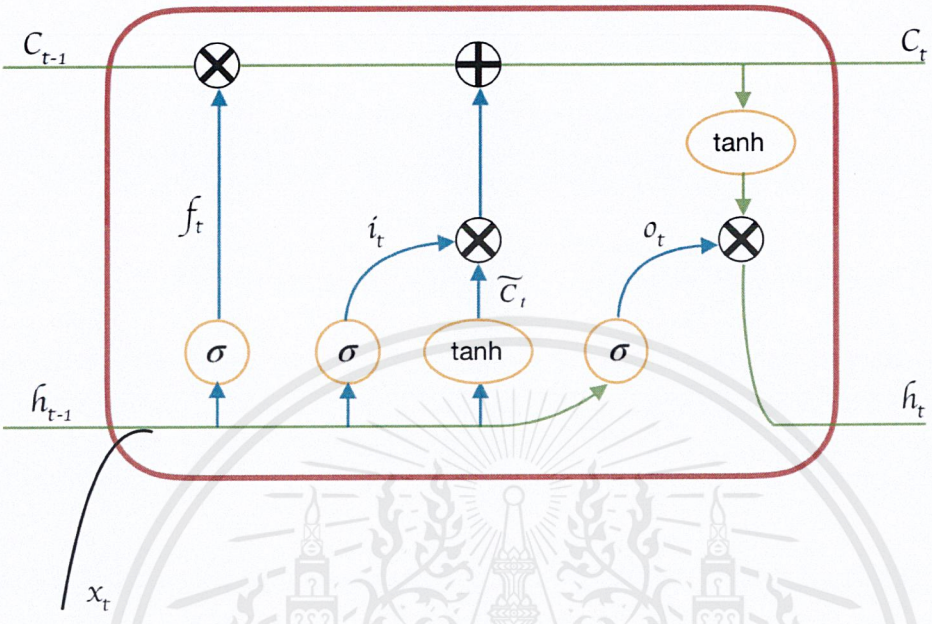

Figure 3.6: A cell of LSTM neural network

Figure 3.6 shows the architecture of a single LSTM cell. The information from the input$(C_t - 1)$ flow towards the output(C_t) of the entire information chain. The sigmoid gate is what chooses which information to let through. The sigmoid function squashes the value of $h_t - 1$ to be in the range of 0 and 1,denoted as f_t. The squashed value is then multiply with the information chain$(C_t - 1)$. By f_t being 0 means that the information is discarded while being 1 results in remembering everything.

The ability of LSTM is to remember new information. This is done using the sigmoid gate on $h_t - 1$ to choose what information to remember and result in i_t. New information is generated by passing $h_t - 1$ to the tanh function that outputs $\widetilde{C}$. By multiplying $\widetilde{C}$ and i_t results in the value that is ready to be update to the information chain. The value is then added to the information chain and result in C_t.

The output of the LSTM cell in Figure 3.6 denotes as h_t. Again the decision of what to output is generated by the sigmoid layer(o_t). The information(C_t) is pass though the tanh function and multiply with o_t resulting in the output(h_t).

LSTM have the cell state to add or remove information, located at the

topmost line in the cell according to Figure 3.6. The cell state is connected to the gates in the cell. The gates in the cell are a way to screen the information passing through the cell. The sigmoid layer outputs numbers between zero and one indicating the amount of information to be going through the gate with value one means let all information going forward and vice versa to protect and control the cell state [24].

Chapter 4

Methodology

In this chapter the methodologies that will be used to achieve the goals will be discussed. The system consists of four main phases, i.e. text preprocessing, feature extractions, training, and evaluating the models.

4.1 Text Preprocessing

This work uses the dataset provided by Quora. It is further processing into a predefined format. As a result, efficiency of the algorithm will increase because irrelevant data will be removed.

4.1.1 Tokenization

Tokenization is a process of dividing the continuous text into pieces. In the Quora's dataset, the questions are sentences and they are suitable to tokenized into words. The pieces of divided words are called tokens. The list of these tokens will become an input for the processing in the next step.

4.1.2 Lemmatization

Since the number of words can be very large, the model would need more computation. Also, the words with the same meaning might be represented with different vectors, causing inaccurate representation. Lemmatization is the method that converts words with the same meaning into a uniform representation[26]. Example of converting inflection word form into the base form is shown in Figure 4.1.

Figure 4.1: Example of lemmatization

4.1.3 Uncapitalization

In distinct computer strings are treated completely different. The word "Basketball" and "basketball" have to be replaced with a single uniform representation. In this work, it will be substituted by uncapitalizing all of the words.

4.1.4 Word Replacements

In English, there are many words that are represented differently but with identical meaning. For example, "USA" and "United States" or "USD" and "Dollar". These words are replaced with only one variation which will increase the robustness of the model.

4.1.5 Stop-words Removal

Stop-words are common words that occur frequently in a language. These words generally contain less information and give little value in differentiating the question pairs. It is removed so that the model can focus on the more important words in the question.

4.2 Feature Extractions

Feature extraction involves reducing the number of resources required to describe a large set of data. To extract meaningful data for machine learning algorithms to learn the data efficiently is also one of processes in feature extraction. In Natural Language Processing, this phase is extremely important since most of the machine learning algorithms only accept numerical data as an input. Thus, It is impractical to feed the data in text format into the model directly. Some meaningful data in the form of the numerical or numerical vector is needed to be correctly extracted. Various method

has been selected to utilize in this phase in order to retrieve the numerical representation of the data.[19][20].

4.2.1 TF-IDF

One of the most popular measurement is TF-IDF technique. It measures the importance of a word to a document in a corpus[27]. Words that frequently appear in many documents will have less important than the words that occur fewer times. TF-IDF weighs a term frequency(TF) and the inverse document frequency(IDF). Each word or term will have its own TF and IDF score. The product of the score is the TF-IDF weight of that word which is used to identify the importance of the word. The value of TF-IDF can be calculated as shown in Equation 4.1.

$$TFIDF(w, d, D) = tf(w, d) \cdot idf(w, D) \tag{4.1}$$

w = *a term that appears in a selected document*

d = *a document*

D = *a collection or a corpus*

Term Frequency(TF)

Term frequency represents the number of a word that occurs in a document. Because each document is different in length, it is possible that a word can appear more often in the document that has a longer length than the shorter length. The term frequency is normally divided by the total number of words in that document as shown in Equation 4.2.

$$TF(w, d) = \frac{w}{d} \tag{4.2}$$

w = *Number of times term w appears in a document d*

d = *Total number of terms in the document d*

Inverse Document Frequency(IDF)

Inverse document frequency is used to measure the significance of the term in a corpus. It weighs how the common a word is in all documents or a corpus. The term such as "is" and "the" may always appear in documents but has less importance. Therefore, this technique is needed to scale down the words that are used frequently while increasing the weight of the rare ones. This

term can be calculated by dividing the total number of documents by the number of documents containing the word as shown in the Equation 4.3.

$$IDF(w, D) = \log \frac{w}{D} \tag{4.3}$$

w = *Total number of documents*

D = *Number of documents with term w in it*

4.2.2 Vector Similarities

After obtaining the word embeddings a single vector representing the whole question is computed using a weighted average. The weight is the TF-IDF which defines how important the words are. After getting vectors representing each question, the cosine similarity will be used to compute the similarity between the questions.

Cosine Similarity

The cosine similarity measures the cosine of the angle between the 2 vectors, not the magnitude. The way of finding the cosine similarity is shown in the Equation 4.4

$$sim(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sum_{i=1}^{n} A_i^2 \sum_{i=1}^{n} B_i^2} \tag{4.4}$$

A_i = *Component of vector A*

B_i = *Component of vector B*

n = *Dimension of vector*

4.2.3 String Distance

To measure the similarity between strings in the question, string metric is used. The string metric only captures the distance between words and not the semantic meaning. A few string similarity measures have been used in this project.

Levenshtein Distance

Levenshtein distance measures the similarity between the two strings by using the minimal number of operations to transform one string to another. The operations are insert, remove, and replace [38].

Hamming Distance

Hamming distance measures the similarity between the two strings by counting the number of different character between the two strings [38].

Longest Common Substring

Longest Common Substring measures the minimum number of character needed to be removed to form a substring of one another.

4.3 Neural Network Architecture

4.3.1 Siamese Neural Network Architecture

The Siamese Neural Network is used to learn the similarities between two inputs. It consists of two subnetworks which have identical weight and biases [30]. It is possible to adopt this architecture in various types of neural network. Each input in the input pair is fed to one of these sub-networks. The structure will be shown in the following section.

The output from each sub-network needs to be fed into a function that outputs how similar they are which will comes in a range of value between 0 to 1. A high value indicates high similarity. In this experiment, threshold of 0.5 is selected so the value at 0.5 or above will be classified as duplicated question pair by the system[31]. In this work, the function that used to measure the similarities is the exponent of negative sum differences between the two output vectors. The detail implementation will be mention in chapter 5.

$$s(x, y) = exp(-\sum_{i}^{n} \|x_i - y_i\|) \tag{4.5}$$

x_i = *Component of vector* x

y_i = *Component of vector* y

4.3.2 LSTM

Bi-Directional LSTM

The bi-directional method can be used in any type of recurrent. The modification allows sequence data to be fed in both ways. There will be two

output vectors from Bi-Directional LSTM. The first vector comes from the data being fed in sequentially. This is identical to a normal LSTM. The second vector is generated from the data being fed in reverse. The output vector can be combined in many ways such as multiplication, addition, concatenation or average[32]. This work uses the concatenation since it preserves most of the data but it does increase the number of parameter of the network.

By having this architecture the network will be able to construct the information without losing deeper information from vanishing gradient descent. It is also meaningful to do this since in natural language is dependency of a sentence might not always be straight forward.

Many-to-one LSTM

The sequential data is input to the model in each time-step. In this case the time-step is the sentence, inputting word by word into the model. The output vector from the LSTM layer is the last state of the calculated final time-step. The diagram of Many-to-one LSTM is the second image in the Figure 4.2.

Many-to-many LSTM

Recurrent layer contains many types of input and output combination. Without the Many-to-many configuration, the recurrent model will only make use of the output from the final time step. The output from every other time-step is then neglected. By constructing a many-to-many LSTM as shown in the last image on Figure 4.2 [33], every vector that is output from each time-step is used by applying a fully connected layer with identical weights.

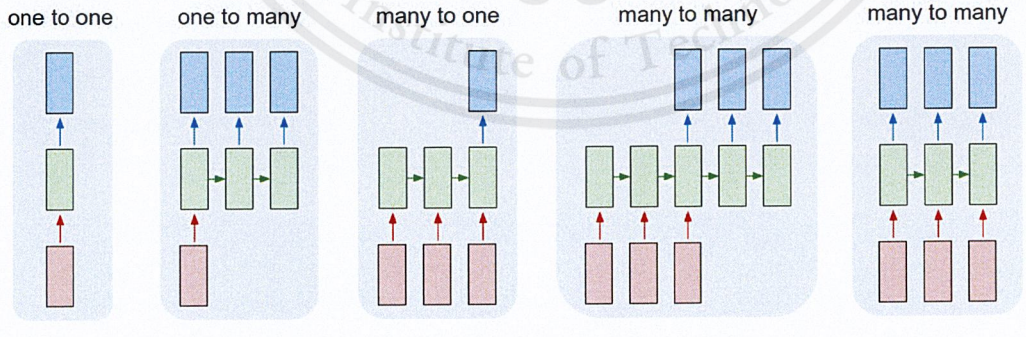

Figure 4.2: Combination of LSTM inputs and outputs

4.4 Hyper-parameters

Hyper-parameters are parameters that have to be set before hand manually, it is not learn by the model,

4.4.1 Loss Function

Loss function tells the neural network how good are the predictions. The model will try to optimize the loss function to reduce the errors.

Mean Square Error

Mean Square Error is the loss function that is used in many machine learning algorithms since it is the easiest to understand. The function measures the sum of squared differences between the label and its prediction [28].

$$MSE := \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{4.6}$$

e_t = *Error of prediction from truth value*

n = *Number of classes*

4.4.2 Regularization

Regularization helps the neural network model to generalize better and helps prevent over-fitting to the training data.

Dropout Regularization

Dropout is a neural network specific regularization technique to reduce over-fitting. By using dropout, some neurons in the neural network will be randomly ignored in the training. This forces the remaining neurons to learn something and not depends only on the dominating neurons. The weights will also be spread between the neurons causing utilization of every input features.[34]

4.4.3 Activation Function

Activation function is to squash the output from the neuron into a certain range. The function is often non-linear which increases the learning power of the network. Without the non-linear activation the model will just act like a regression model[36].

Sigmoid Function

Sigmoid Function squash the values to be between 0 and 1. The function can be used in the output of the neural network to gives out predictions of classification problems.

$$f(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Rectifier Linear Unit

Rectifier Linear Unit or ReLu is a simple activation function. The function turns all of the negatives to zero and keep the positives as is. Researches have shown that ReLu trains faster on larger network than other activation functions [37].

$$f(x) = max(0, x)$$

Chapter 5

Experimentation

5.1 Dataset

The following experiment is done on the dataset which is provided by Quora. The data is available on Quora's website.

The dataset consists of 404,290 rows and 5 columns. The sample of the dataset is shown in Figure 5.1. Columns with label 'qid1' and 'qid2' indicates the id of question 1 and question 2, 'id' is the identification number of each question pair, 'question1' and 'question2' are the unprocessed textual data of the question pairs. Lastly, 'is_duplicate' is the label that identifies whether the question pairs are duplicated.

id	qid1	qid2	question1	question2	is_duplicate
5	11	12	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1
6	13	14	Should I buy tiago?	What keeps childern active and far from phone ...	0
7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
8	17	18	When do you use シ instead of し?	When do you use "&" instead of "and"?	0
9	19	20	Motorola (company): Can I hack my Charter Moto...	How do I hack Motorola DCX3400 for free internet?	0

Figure 5.1: A Sample of the Dataset.

5.2 Development Tools

The system is implemented in python and libraries that are used in the development are shown in Table 5.1.

Table 5.1: Libraries used

Libraries	Purpose and Usage
Keras, Tensorflow	Neural Networks implementation
Pandas	Manage data
Numpy	Computation and Data Manipulation
Matplotlib	Data Visualisation
Gensim	Word Embedding
Cuda, CuDNN	GPU Computation

5.3 Results

5.3.1 Experiment Setup

Experiment setup is shown so the the results from our research can be replicated.

Resampling

The dataset that is provided by Quora is not balanced. There is less duplicated question than the one that is not duplicated. The non-duplicated rows are randomly removed until the data is balanced. After the removal, there is 298,526 records left.

The data is separated into two sets, training and testing set. The sets are created by sampling without replacement on the original data. The size of the testing set is 25% while the remaining 75% is the training set.

The training set is used to train the model, which allows the model to learn the parameters. The testing set will be used to measure the performance of the model and how it performs. The hyperparameters of the subnetwork in the Siamese model are identical to that of the network which does not use the Siamese architecture.

Optimization

AdaDelta is an adaptive learning rate method, the algorithm optimizes the neural network's weights and biases using gradient descent. The learning rate is not required to be tuned. The author also claimed that the method was robust from noisy gradients [35]. The algorithm works by using a learning

rate from the past that is in a specific window to compute the new learning rate.

The loss function that is used to train these neural network is the Mean Squared Error since it is the most intuitive and easy to understand.

Evaluation

The experiments will use accuracy, loss, precision, and recall as a performance measure metrics. Since the class in the dataset is now balanced, accuracy in this experiment can be a trustworthy overall measurement metric. While precision and recall will provide specific information on the experiment. The performance measures and confusion matrix is constructed using the test set. Loss value indicated how certain model behaves after each iteration, in this case, after each epoch.

The loss vs epoch and accuracy vs epoch of each model is plotted in respective section to show how effective each model learns. The accuracy in each epoch is obtained by using the testing set at the end of every epoch. The plot also shows whether the model over-fits or does it generalize correctly.

The loss function that is used to train these neural network is the Mean Squared Error since it is the most intuitive and easy to understand.

Neural Network Implementation

The Neural networks is implemented in python using Keras version 2.1.5 with Tensorflow as backend. The CPU is used to train the model due to the insufficient GPU's ram.Every layer is learned using 25% dropout and RELU activation except for the last layer which uses sigmoid activation and no dropouts.

The implementation of the Siamese model is shown in 5.2 and 5.3.The model consists of 4 inputs which are ,Question 1 Word embedding, Question 1 general features, Question 2 Word embedding and Question 2 general features. Each input is fed in different subnetworks of the model. All of the general features are fed in a fully-connected layer, while the section that receives word embedding changes. Each subnetwork of the Siamese model contains the same weight as shown with dotted box around each subnetwork. The final output vector is then combine using a distance measure. This distance measure will output the similarity score ranging between 0 and 1. If

the vector is similar the output from the distance measure will be close to 1, while if it is not the output close to 0.

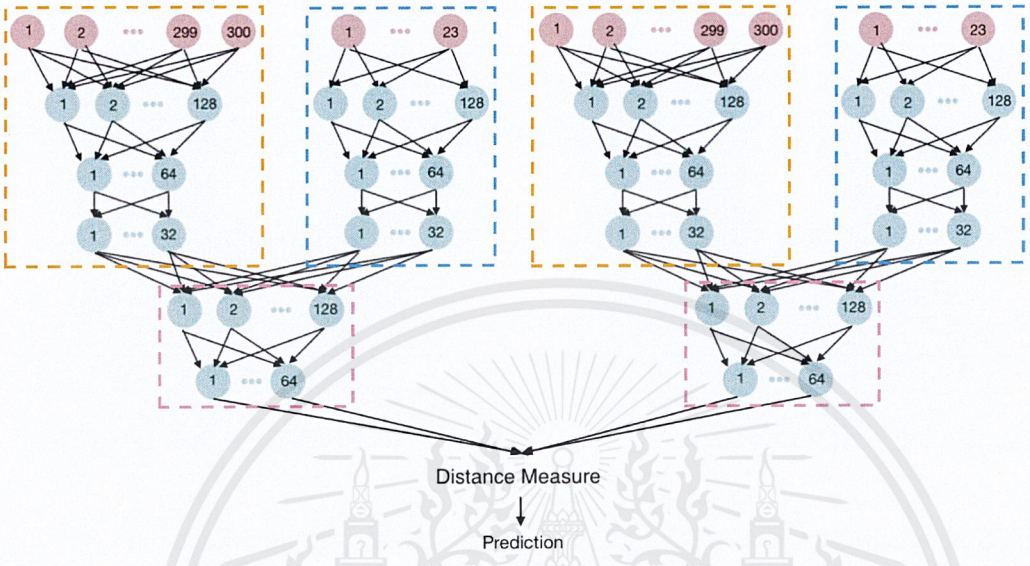

Figure 5.2: Siamese Feedforward Implementation

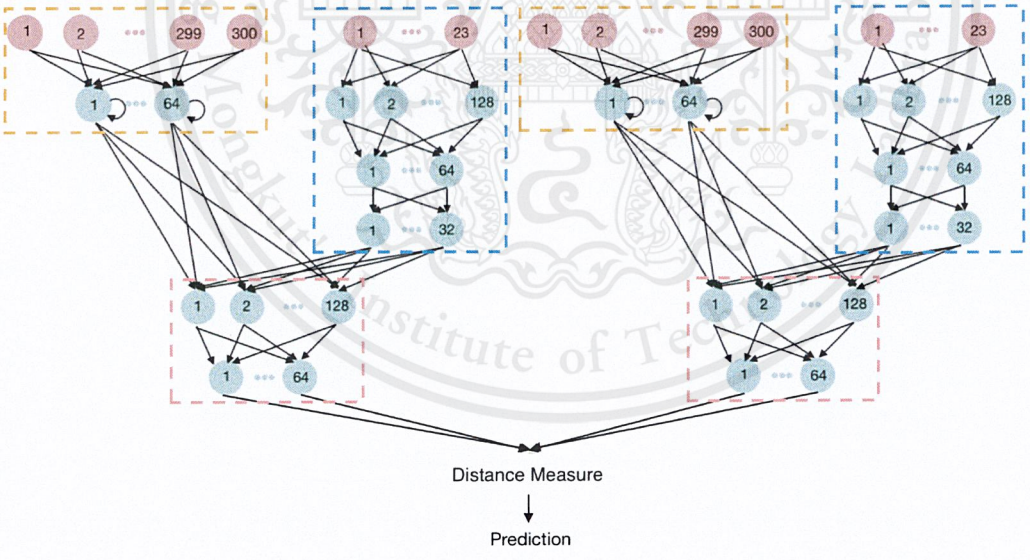

Figure 5.3: Siamese LSTM Implementation

The implementation of the Non-Siamese models are similar to that of the Siamese models. The model also contains 4 inputs identical to the Siamese

model as shown in 5.4 and 5.5. In this implementation the weights and biases are not shared and is updated independently. The final output vector is instead combined using a single neuron with the sigmoid activation instead of the distance measure. The model is trained so that the prediction is 1 if the model thinks the question is duplicated, otherwise 0.

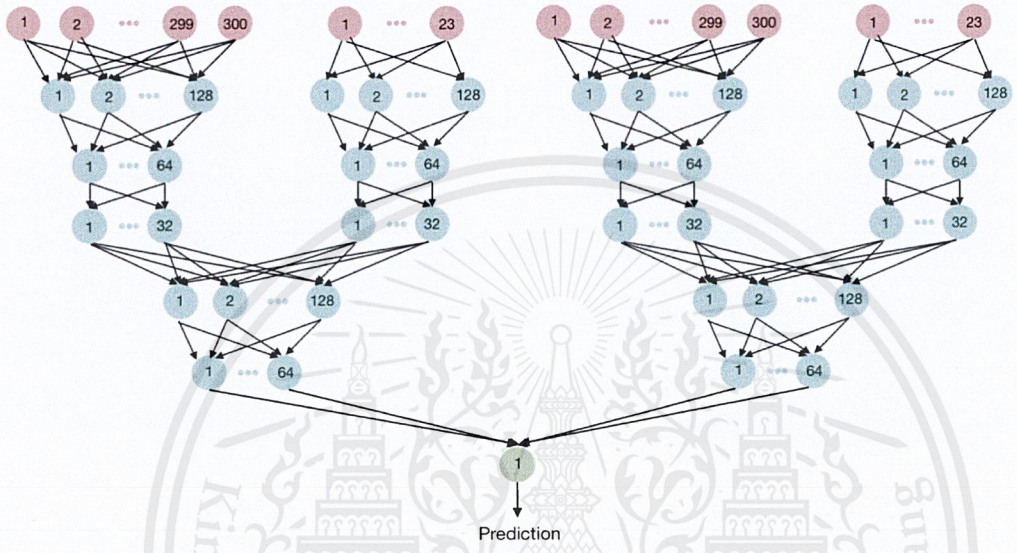

Figure 5.4: Feedforward Implementation

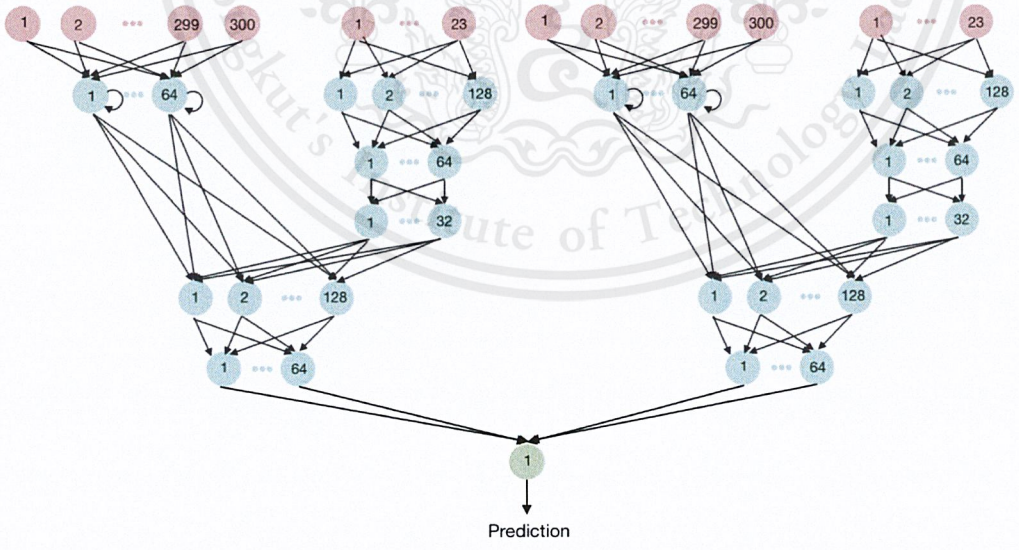

Figure 5.5: LSTM implementation

Table 5.2: Development environment

CPU	Intel i5 8400
GPU	GeForce GTX 960 2Gb
Ram	8Gb 3000Mhz DDR4

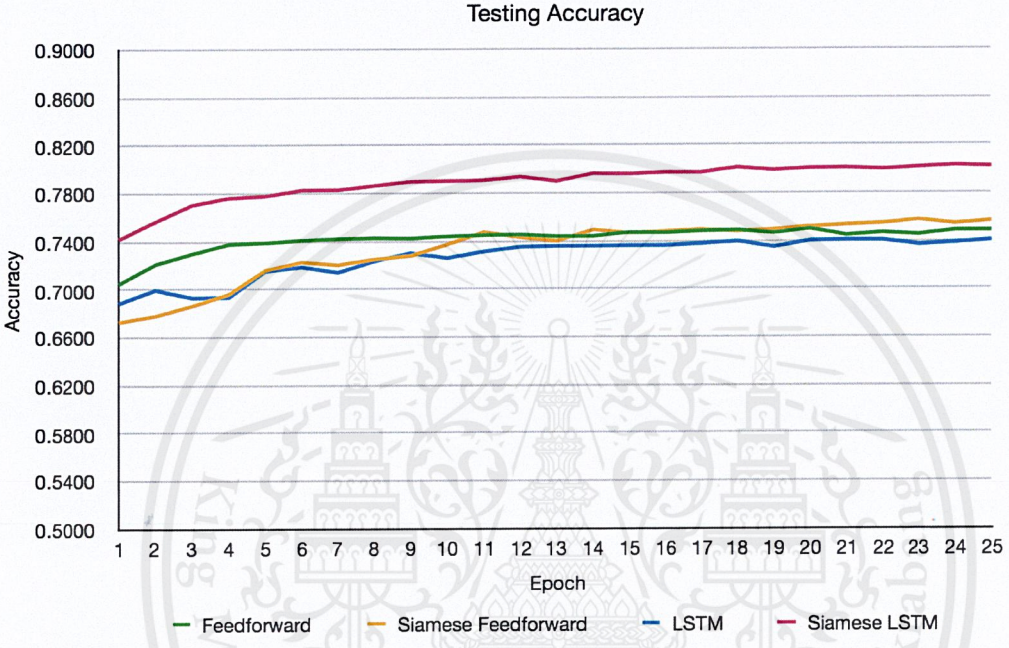

Figure 5.6: Testing Accuracy

5.3.2 Words Embedding as Features

In this section the only feature used in this model is Word2Vec embedding. Figure 5.6 shows the accuracy value obtain by testing each model against the test set. Overall, Siamese LSTM model has the best performance measured by accuracy.

Feedforward model's accuracy starts from around 70% and constantly increase to stabilized at around 74%. The Feedforward model stop learning at around the fourth epoch. LSTM model accuracy fluctuates at the first few epochs. The model then start to converge at around twelveth epoch.

For Siamese variants of the model, Feedforward Siamese model has the lowest accuracy out of all model in early training stage but end up as second most accuracy score at the end of training. Siamese LSTM has the best ac-

curacy score out of all models in experimentation at all stage of the training with final accuracy score around 80%. This indicated that Siamese architecture has the potential in classifying duplicated data when applied in neural network model when provided with enough data and time.

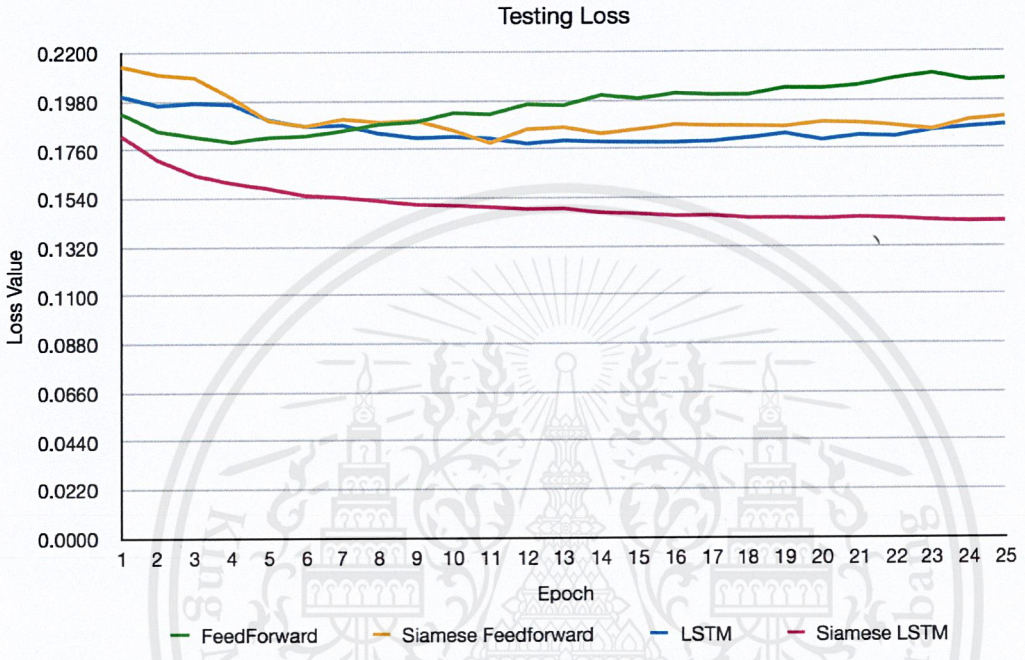

Figure 5.7: Loss Value of each applied model

Figure 5.7 shows the loss value of the test set predictions of each model. Again, Siamese LSTM has the best loss value after the training. Feedforward model's loss start to increase gradually around the fourth epoch which is a sign of overfitting. LSTM model also has the same problem although not as obvious as Feedforward model.

Looking at Siamese variants of the model, Siamese Feedforward loss value fluctuates at all stage of training and has the end result about the same as LSTM model. Siamese LSTM model has the best result at all stage of training. Siamese LSTM model results in Mean Square Loss of 0.14 after the training. Looking at the graph the loss value still has the downward trend. This indicates that Siamese LSTM have the potential to lower the loss value the model keeps training.

Table 5.3: Evaluation

	Feedforward	**Siamese Feedforword**	**LSTM**	**Siamese LSTM**
Precision	0.7309	**0.8028**	0.6914	0.7866
Recall	0.7588	0.7357	0.7678	**0.8124**
Accuracy	0.7484	0.7564	0.7403	**0.8018**
Sec/Epoch	94	**64**	480	462

Looking at the results of the implemented model in Table 5.3. The Siamese architecture reduces the training time of both the LSTM and the Feedforward model by around 20 to 30 seconds. The Siamese architecture also improves the accuracy of the two models. The Feedforward model's accuracy improves by around 0.8% and LSTM model's accuracy improves over 6%. Overall, the Siamese LSTM performs the best since after training, the accuracy and recall is the best out of the implemented models. The Precision is also the second best after Siamese Feedforward model. In our experiment we value training time less than the prediction power of the model.

5.3.3 Additional Features

In this section more features are extracted to train the model. There are three main types of features added which are counts, vector similarity and string distance. Count features include count of words, character, capital letter, numbers, average of numbers, symbol and common words. Vector similarity feature is obtain by averaging the vector of every word in the question to get a vector representation of the question. Cosine similarity is then used to find the similarity between the two sentence vectors. String distance such as Hamming, Jaccard and Levenshtein distance is used to find the literal difference between the two question's strings.

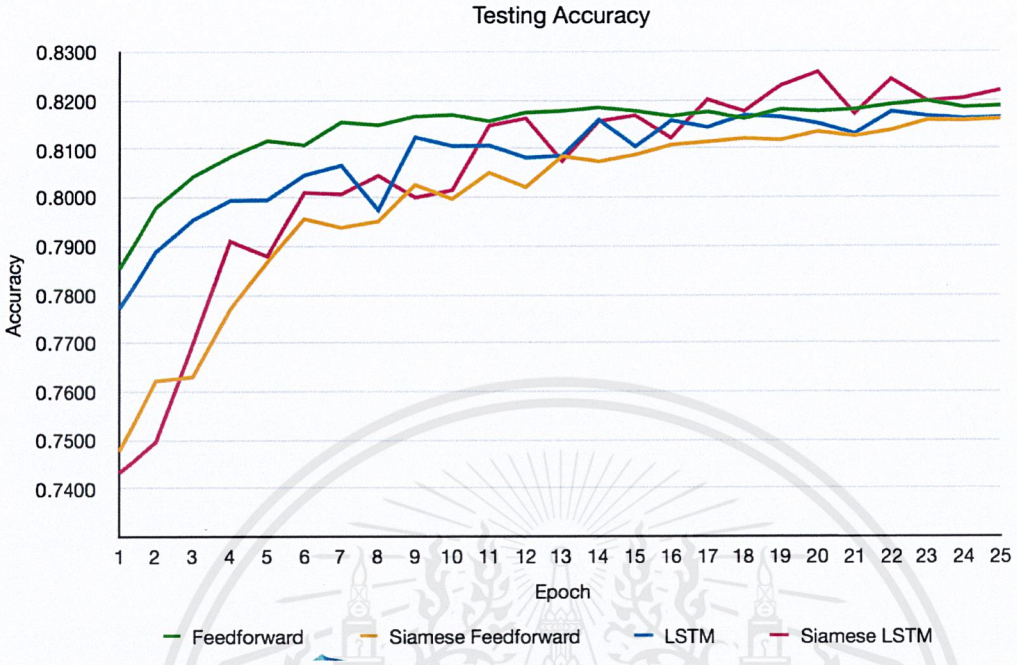

Figure 5.8: Testing Accuracy

After the features is added, every neural network's accuracy increases. 5.8 shows that the model start to converge at a very similar value at around 82%. The Siamese LSTM does not outperform every other model as much as before. The Non-Siamese model learns faster since its accuracy is higher in the first few epochs. Judging from Figure 5.8 the performance of the model except Feedforward can be improved if it is trained on more epochs. This could be because increasing number of features leads to more parameters in the neural network.

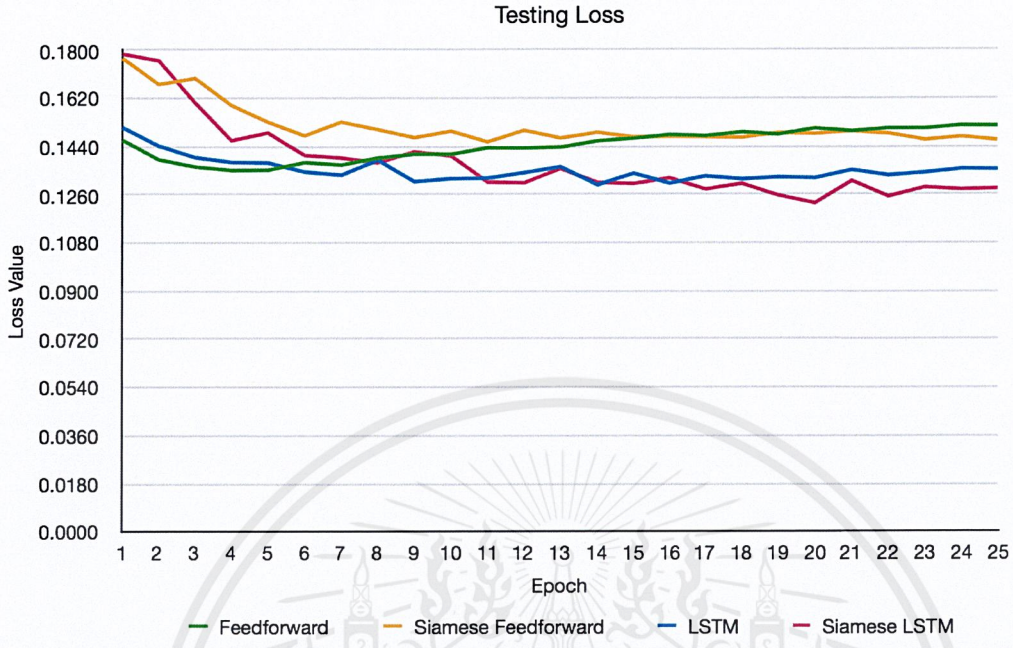

Figure 5.9: Testing Loss

Table 5.4: Evaluation

	Feedforward	**Siamese Feedforward**	**LSTM**	**Siamese LSTM**
Precision	0.8446	0.8677	0.8754	**0.9129**
Recall	**0.8041**	0.7874	0.7838	0.7733
Accuracy	0.8188	0.8161	0.8164	**0.8220**
Sec/Epoch	100	**69**	472	485

The mean squared error of the Siamese variants start of higher than the Non-Siamese variants. However, as the model trains the Siamese variants outperform the Non-Siamese variants. The loss of the Feedforward model also increases which indicates overfitting, while the Siamese Feedforward loss gradually decreases which shows that the Siamese architecture is more robust to overfitting.

5.3.4 Model Improvements

From the previous experiment the Siamese LSTM has found to be the best out of the original four models. The variations of Siamese LSTM have been chosen with an expectation to improve the model's performance. All of

the variations are implemented using the Siamese LSTM architecture. The LSTM used in the previous experiments is Many-to-one LSTM and for the improvements Many-to-many and Bi-directional Siamese LSTM will be used.

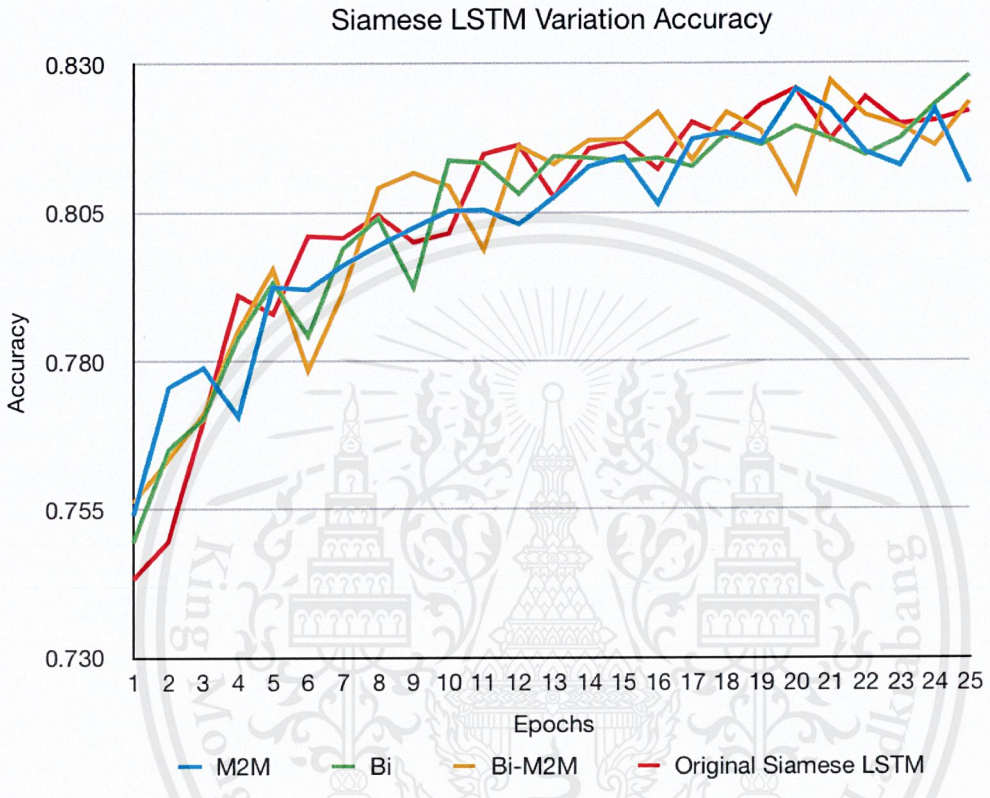

Figure 5.10: Accuracy of Siamese LSTM Variations

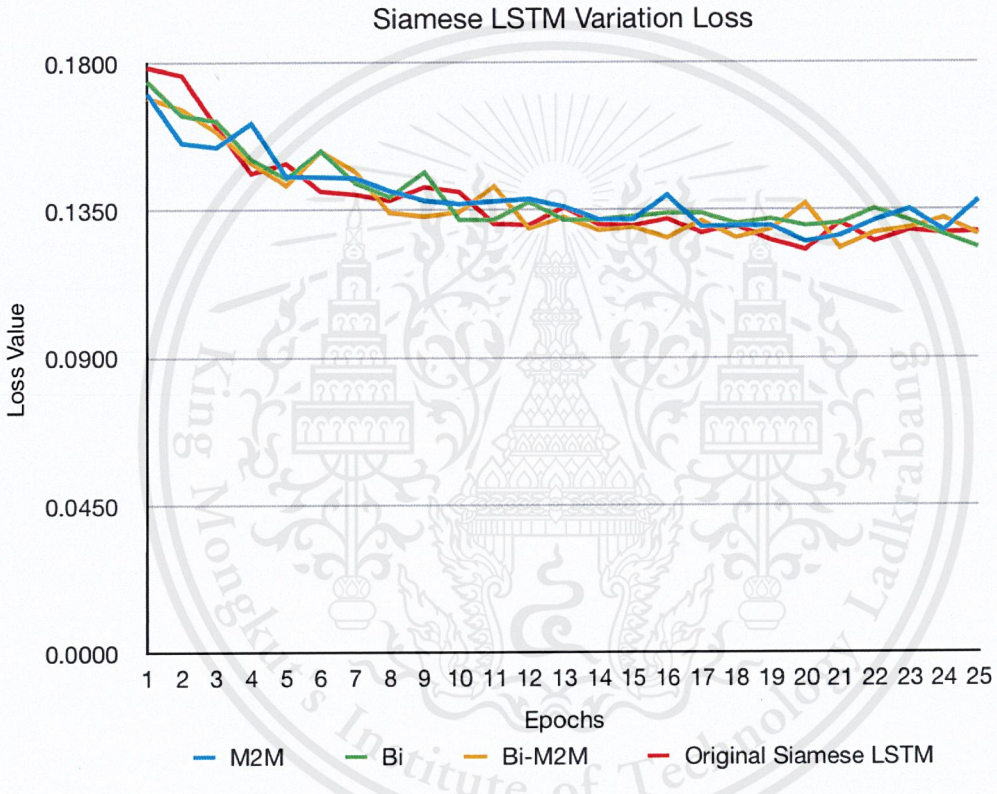

Figure 5.11: Loss Value of Siamese LSTM Variation

Figure 5.10 is the results of each variation of the Siamese LSTM which is original Siamese LSTM, Bi-directional LSTM, Many-to-Many LSTM and Bi-directional Many-to-many LSTM. From graph, it can be seen that the rate of how fast the model learn is very similar. All of the model could be improved by increasing the epochs in the training phrase since all of the model does not fully converge yet. After the four models start converging the accuracy is very similar at around 82%. At 24th epoch there is a slight jump in Bi-directional LSTM's accuracy which is the highest amongst the three. The loss also conveys identical trend to the accuracy plot which is shown in Figure 5.11

Table 5.5: Siamese LSTM Variation Evaluation

	M2M	**Bi**	**Bi-M2M**	**Original Siamese LSTM**
Precision	0.7517	**0.7975**	0.7776	0.7733
Recall	**0.9271**	0.8810	0.9078	0.9129
Accuracy	0.8098	**0.8280**	0.8235	0.8220
Sec/Epoch	535	864	1011	**85**

Table 5.5 shows that every variations of the Siamese LSTM model increases the training time from the original model. The Siamese Bi-directional LSTM doubles the training of the Siamese LSTM, which is reasonable because the model uses both the forward sequence and the backward sequence as an input. The training time of Many-to-many LSTM also increases because calculations need to be performed on every timestep. The accuracy of Many-to-many LSTM is lower that original LSTM. This shows that using every sequence from each timestep is not beneficial. The Bi-directional Siamese LSTM resulted in the highest accuracy at 82.8% on the testing data.

5.4 Duplicate Question Classifier Application

An application is built to test the proposed system on real-world data. The application accepts question pair from user and output a result along with score indicate similarity between two questions. In this experimentation, the threshold of 0.5 is selected to be the dividing point which means that the application will classify question pair with 0.5 score or above to be duplicated question and vice versa.

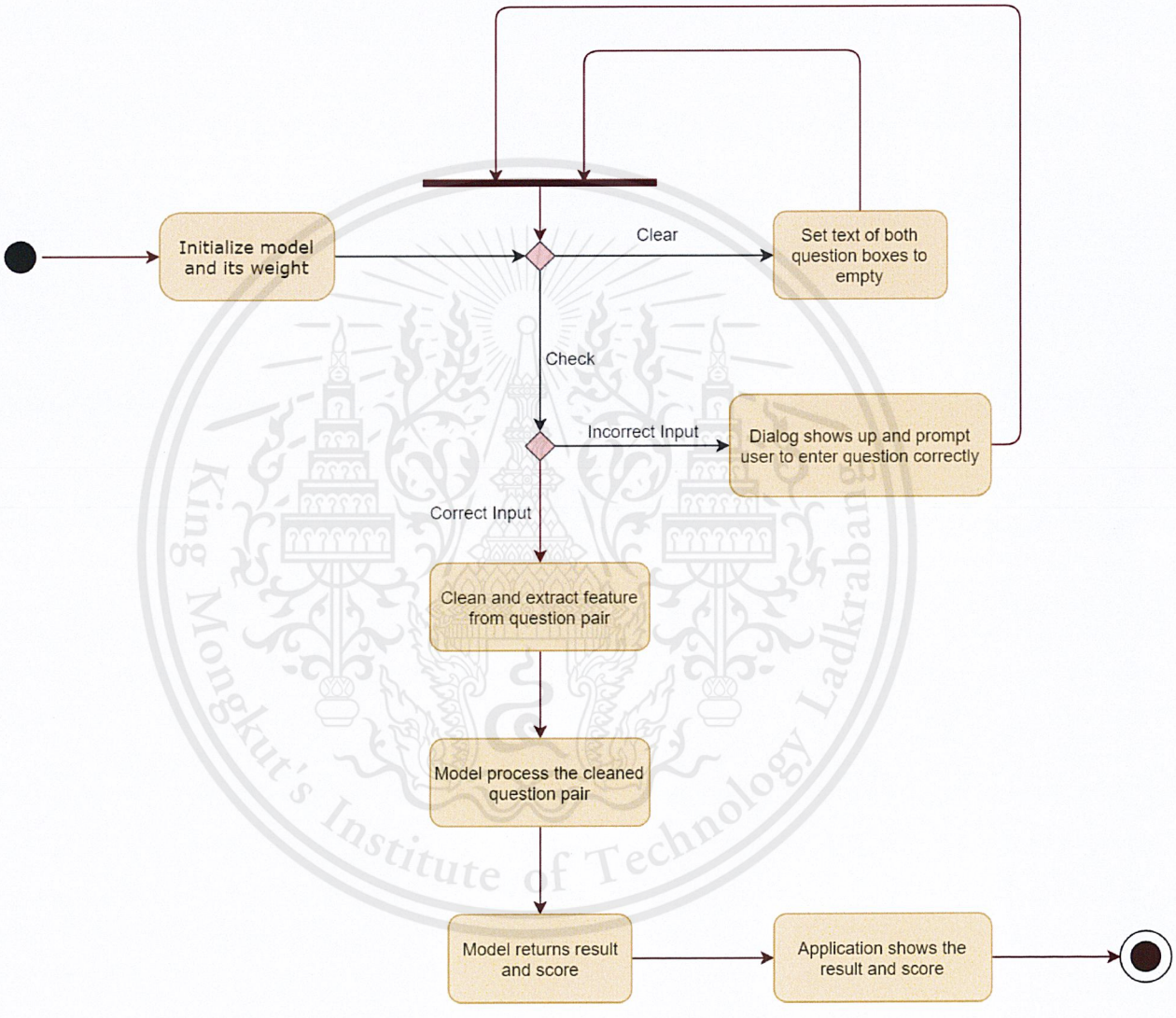

Figure 5.12: Activity diagram showing operation flow of the application

Figure 5.12 shows the operation flow of the application. The application first initialized the model and its weight. After that, the user can input two questions and click check button to get the result or click a clear button to clear the question in question boxes and clear the result text.

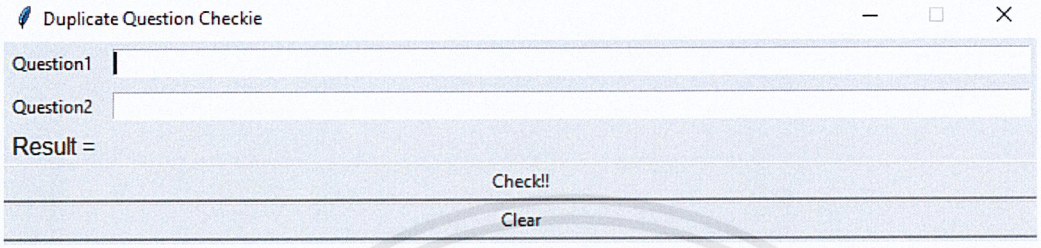

Figure 5.13: Screenshot at the start of the application

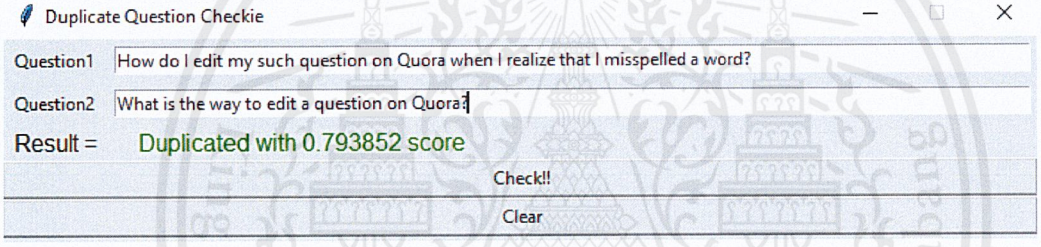

Figure 5.14: After clicking Check!! button, green text result means duplicated question

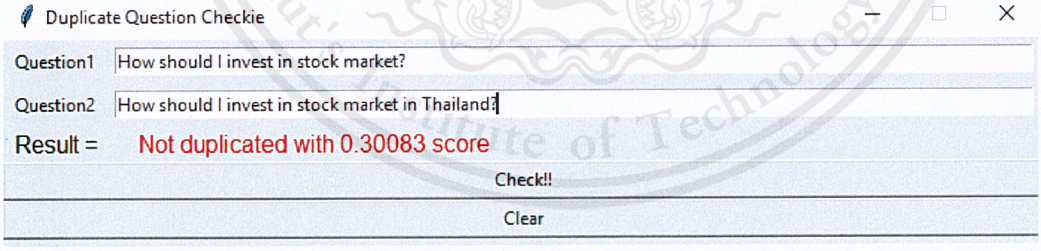

Figure 5.15: Red text indicate not duplicated question

Figure 5.13, 5.14, and 5.15 shows the screenshots of the application. After entering question pair and then clicking check button, the model will predict and returns the result which the application shows under the question box. The green color of result text indicates that question pair the user has

entered is duplicated while red color indicates the opposite. In addition to the classification result, the application also shows the score of the result. The score ranging between 0 to 1 shows how similar the questions are for the model with higher score indicates high similarity between two questions.

5.5 Summary

Comparing the selected type of neural network, the network that uses the Siamese architecture achieves higher validation accuracy and does not overfit as much. The model that does not use the Siamese architecture has a high training accuracy but low testing accuracy. The Siamese deals with overfitting by processing the same type of input uniformly. This is logical because the input is the same type. It is odd that the training time of Siamese-LSTM is higher than LSTM however Siamese-FFN is lower than FFN. The LSTM model takes more time to train due to its complexity but also does achieve better results.

The variance of Siamese LSTM doesn't increase the model predictive power as much as the time taken to train the model. It might be more suitable in some situations to use the model with lower accuracy. There is a trade-off between accuracy and training time, the accuracy of Bi-directional only increases by 0.6% from the original Siamese LSTM while its training time doubles.

Chapter 6

Conclusion and Discussion

This study is proposed to find the most effective method to classify duplicate question pairs. Mainly utilizes various neural network models and Natural Language Processing techniques. The system is trained on Quora's open dataset of labeled question pairs. The dataset contains only English questions. The selected neural network models in the experimentation are Feedforward model and LSTM model. With Siamese neural network architecture as a supplement on the model as it is effective in a task that involves finding a similarity or a relationship between two comparable things.

Several NLP techniques are used to deal with text data before feeding it into neural network model. Cleaning techniques include tokenization, lemmatization, uncapitalization, word replacements on abbreviation, and stop-words removal. Word2Vec is used to map words into meaningful vector representation based on the word relationships. Then, finding the differences between each word by using Cosine Similarity. The dataset, which comes with an unbalanced class amount between duplicated class and not duplicated class, is resampled to an equal amount of classes to prevent the model being bias.

From the experiment using only Word2Vec as the input, the Siamese architecture improves both the training time and the predictive power of both the LSTM and the Feedforward model. The Siamese LSTM have the highest accuracy at 80.18%.

After more features are added, Siamese LSTM still result in the highest accuracy at 82.29%. It is also found that vanilla Feedforward model is overfitted on the training data since the loss increased while training. But the

Siamese Feedforward doesn't have this problem which shows that Siamese architecture help generalize the model. Oddly applying Siamese architecture to neural network model increases the predictive power of the LSTM model but also increases the training time.However, applying the architecture on Feedforward model decreases the training time and the accuracy also decreases.

When comparing between Feedforward and LSTM model, the LSTM performs better than Feedforward overall except for the training time. After more features are added, the model performs more similarly. The best performance in terms of accuracy in experimentation is Siamese Bi-LSTM model with the accuracy of 82.80 %. This study concludes that Siamese architecture helps reduces overfitting in the neural network models.Moreover, Siamese architecture apply on LSTM perform better that other neural network models in terms of various performance measures such as accuracy, loss, precision, and recall

The study can be improved further by including more techniques in both Natural Language Processing and Neural Network model. Embedding vector at a higher level such as sentence embedding (Doc2Vec) can be experimented to see whether it improves model performance. More text mining features can be included. The model can be experimented with more features or fewer features to find the most efficient features in classifying duplicate questions. Extending the system to support Thai language can be done when there are more Thai language resources.

References

[1] E.D. Liddy. (2001). *Natural Language Processing.* [Online]. Available: http://surface.syr.edu/cnlp/11/

[2] E. Burns. (2017). *natural language processing (NLP).* [Online]. Available: http://searchbusinessanalytics.techtarget.com/definition/natural-language-processing-NLP

[3] *AI - Natural Language Processing.* [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_natural_language_processing.htm

[4] D. Thakur. *Lexical Analysis - Compiler Design.* [Online]. Available: http://ecomputernotes.com/compiler-design/lexical-analysis

[5] P.D. Blinov, M.V. Klekovkina, E.V. Kotelnikov and O.A. Pestov. (2013). *Research of lexical approach and machine learning methods for sentiment analysis.* [Online]. Available: http://www.dialog-21.ru/media/1226/blinovpd.pdf

[6] A. Voutilainen, *The Oxford Handbook of Computational Linguistics*, 2nd edition. Great Clarendon Street: Oxford University Press, 2014.

[7] H. Schmid. (1995). *Improvements In Part-of-Speech Tagging With an Application To German.* [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=22ABE06B1D41BBE15F16EBA1C08A1CFF?doi=10.1.1.52.2255&rep=rep1&type=pdf.

[8] R. Schank. (1978). *Conceptual Information Processing.* [Online]. Available: https://benjamins.com/#catalog/journals/sl.2.3.11wil/details

[9] C. Brun and C. Hagège. 2003. *Normalization and paraphrasing using symbolic methods.* [Online]. Available: https://dl.acm.org/citation.cfm?doid=1118984.1118990

[10] E. Clark, K. Araki, *Normalization in Social Media: Progress, Problems and Applications for a Pre-Processing System of Casual English.* [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042811024049

[11] M. Kaufmann. (2010). *Syntactic normalization of Twitter messages. In International Conference on Natural Language Processing.* [Online]. Available: https://pdfs.semanticscholar.org/2962/0eaa36ad1caefb871039a7ed4b4891d96666.pdf

[12] A. L. Samuel. (1959). *Some Studies in Machine Learning Using the Game of Checkers.* [Online]. Available: http://ieeexplore.ieee.org/document/5392560/?reload=true

[13] W. Yin, H. Schutze. *Convolutional Neural Network for Paraphrase Identification* [Online]. Available: https://aclanthology.info/pdf/N/N15/N15-1091.pdf

[14] A. Zell. (1994). *Simulation Neuronaler Netze (Simulation of Neural Networks in German).* [Online]. Available: https://www.researchgate.net/publication/34448793_Simulation_Neuronaler_Netze

[15] J. A. Botha, E. Pitler, J. Ma, A. Bakalov, A. Salcianu, D. Weiss, R. McDonald and S. Petrov. (2017). *Natural Language Processing with Small Feed-Forward Networks.*[Online]. Available: https://arxiv.org/abs/1708.00214

[16] C. Olah. (2015). *Understanding LSTM Networks.* [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[17] T. Addair. *Duplicate Question Pair Detection with Deep Learning.* [Online]. Avaliable: https://web.stanford.edu/class/cs224n/reports/2759336.pdf

[18] E. Dadashov, S. Sakshuwong and K. Yu. *Quora question Duplication.* [Online]. Available: https://web.stanford.edu/class/cs224n/reports/2761178.pdf

[19] N. S. Sarwan *An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec.* [Online]. Available:https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[20] C. Mccormick *Word2Vec Tutorial - The Skip-Gram Model.* [Online].Available:http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[21] D. Robinson. (2017). *Convert Statistical Analysis Objects into Tidy Data Frames.* [Online]. Available: https://CRAN.R-project.org/package=broom.

[22] J. Pennington, R. Socher and C. D. Manning. *GloVe: Global Vectors for Word Representation.* [Online]. Available: https://nlp.stanford.edu/pubs/glove.pdf

[23] R. Belavkin. *Feed-Forward Neural Networks.* [Online]. Available: http://www.eis.mdx.ac.uk/staffpages/rvb/teaching/BIS3226/hand11.pdf

[24] C. Olah. *Understanding LSTM Networks.* [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[25] *Model Fit: Underfitting vs. Overfitting.* [Online]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

[26] *Stemming and lemmatization.* [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[27] J. Silge and D. Robinson. (2018). *Term Frequency and Inverse Document Frequency (tf-idf) Using Tidy Data Principles.* [Online]. Available: https://cran.r-project.org/web/packages/tidytext/vignettes/tf_idf.html

[28] *TensorFlow API r.8 Mean Square Error.* [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/losses/mean_squared_error

[29] R. DiPietro. (2016). *A Friendly Introduction to Cross-Entropy Loss.* [Online]. Available: https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

[30] H. Gupta. (2017). *One Shot Learning with Siamese Networks in PyTorch.* [Online]. Available: https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e

[31] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger and R. Shah. (1994). *Signature Verification Using A "Siamese" Time Delay Neural Network.* [Online]. Available: https://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf

[32] *Layer wrappers - TimeDistributed.* [Online]. Available: https://keras.io/layers/wrappers/

[33] G. Chevalier. *LSTMs for Human Activity Recognition.* [Online]. Available: https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition

[34] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov . (2012). *Improving neural networks by preventing co-adaptation of feature detectors.* [Online]. Available: https://arxiv.org/pdf/1207.0580.pdf

[35] M. D. Zeiler. (2012). *ADADELTA: An Adaptive Learning Rate Method.* [Online]. Available: https://arxiv.org/abs/1212.5701

[36] D.Gupta. (2018). *Activation Functions and When to Use Them?.* [Online]. Available: https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/

[37] A. Krizhevsky, I. Sutskever and G.E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* [Online]. Available:https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[38] R. Vogler. (2015). *Comparison of String Distance Algorithms.* [Online]. Available: https://www.joyofdata.de/blog/comparison-of-string-distance-algorithms/