An Empirical Evaluation Of Attention And Pointer Networks For Paraphrase Generation

Varun Gupta

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

July 2019

© Varun Gupta, 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Varun Gupta**

Entitled: **An Empirical Evaluation Of Attention And Pointer Networks For Paraphrase Generation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Examiner
Dr. Ching Y. Suen

_____ Examiner
Dr. Tristan Glatard

_____ Supervisor
Dr. Adam Krzyżak

Approved by _____
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

_____ 2019

_____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

An Empirical Evaluation Of Attention And Pointer Networks For Paraphrase Generation

by Varun GUPTA

In computer vision, one of the common practice to augment the image dataset is by creating new images using geometric transformation, which preserves the similarity. This data augmentation was one of the most significant factors to win the Image Net competition in 2012 with vast neural networks. Similarly, in speech recognition, we saw similar results by augmenting the signal by noise, slowing signal or accelerating it, and spectrogram modification.

Unlike in computer vision and speech data, there haven not been many techniques explored to augment data in natural language processing (NLP). The only technique explored in text data is by lexical substitution, which only focuses on replacing words by synonyms.

In this thesis, we investigate the use of different pointer networks with the sequence to sequence models, which have shown excellent results in neural machine translation (NMT) and text simplification tasks, in generating similar sentences using a sequence to sequence model and of the paraphrase dataset (PPDB). The evaluation of these paraphrases is carried out by augmenting the training dataset of IMDb movie review dataset and comparing its performance with the baseline model. We show how these paraphrases can affect downstream tasks. Furthermore, We train different classifiers to create a stable baseline for evaluation on IMDb movie dataset. To our best knowledge, this is the first study on generating paraphrases using these models with the help of PPDB dataset and evaluating these paraphrases in the downstream task.

Acknowledgements

It is a pleasure to thank many people who made this thesis possible.

First of all, I would like to express my sincere gratitude to my supervisor Prof. Adam Krzyżak, for giving me the chance to work under his supervision. His continuous assistance, encouragement and brainstorming helped me throughout my thesis work. Without him, it wouldn't be possible to convert an idea into a thesis.

I also want to thank my thesis committee, Drs. T. Glatard, C. Y. Suen and A. Hanna for reviewing my work and giving insightful comments.

My sincere thanks also go to Iulian Vlad Serban, a Ph.D. student from the University of Montreal and CEO of Korbit.ai, who helped me a lot in giving right pointers and giving me a chance to work with him on beautiful project Korbit which taught me a lot of machine learning and NLP.

Last but not least, I would like to thank my family: my parents, grandparents, aunt, brother, sister, and my girlfriend who supported me to study in Canada financially, mentally and always kept me motivated in tough times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Definition of paraphrase as given in Cambridge dictionary: Paraphrase is defined as the repetition of something written or spoken using different words, often in a humorous form or in a more straightforward and shorter form that makes the original meaning clearer.

Restating and rephrasing a given sentence as the target or output sentence keeping its overall meaning preserved is known as paraphrase generation. Take, for example, the following two sentences:

Sentence 1 *The price of a resort vacation typically includes meals, tips and equipment rentals, which makes your trip more cost-effective.*

Sentence 2 *All-inclusive resort vacations can make for an economical trip.*

The two sentences above are paraphrases of each other because both of them have the same meaning, but are written differently. Understanding paraphrases has shown to be a beneficial task for many additional natural language processing (NLP) jobs, such as the expansion of questions and answers for better understanding, simplification of sentences, summarizing paragraphs, information retrieval, information extraction, restating utterances generated by a conversational agent ChatBots to map student vocabulary. It could also be very beneficial in sentiment analysis or to increase the amount of data for supervised learning [13], [47].

The criteria of semantical equivalence (i.e., the same or almost the same meaning) are challenging to define precisely and can vary from task to task. Paraphrase generation is a task in which we try to generate these similar pair of sentences. Although there are many kinds of research done in the production of paraphrases using different techniques, the study of paraphrase generation is far from enough. It is not always possible to use the same alignment table or vocabulary set for generating paraphrases in different contexts. For example, in the English literature context, we can convert *increase* to *expand*, but that might not be possible in the deep learning domain. Example of it could be *The model complexity* ***expands*** *with a more significant number of parameters*, which might not be a correct paraphrase in the deep learning context.

1.2 Applications

As mentioned earlier, the generation of paraphrases can be useful in a variety of tasks. For example, in the information extraction system paraphrases can be used to extract information which connects to specific events. Interpretations can be used in this type of system to generate or augment equivalence extraction pattern semantically.

The paraphrase is also used in text summarizing task to summarize sentences which are more critical than other sentences in a text while preserving the central meaning of the text [34], [37].

In conversational agent systems also known as chat-bots, paraphrases are used to convey the message which could be easily understood by the user or student by mapping the student vocabulary on the target sentence or by rephrasing the utterance generated by the agent.

Sentence compression is also one kind of an application where paraphrase can be very helpful. In sentence compression, the condition is, that output sentence must be shorter than source sentence and should have same meaning [14], [18].

In question answering systems, rephrasing the questions may lead to the generation of more questions and with answers that cannot be directly mapped to previous questions. This type of technique is also beneficial to improve the system's performance. To obtain most frequent questions (FAQs) we often uses paraphrases [2], [14], [45].

Machine translation problem can also be related to paraphrasing but across language when we try to generate similar text in a different language than source language.

1.3 Goal of the Thesis

In this work we try to assess classical based on attention network [3] and pointer network [17], [16], which yielded good results in neural machine translation (NMT) and text simplifications tasks using the sequence-to-sequence model. We investigate how well these models perform in generating paraphrases under supervision of the paraphrase database (PPDB). We further assess the use of these paraphrases in the downstream task, i.e., by producing similar sentences for training dataset of IMDb movie dataset and evaluate algorithms' performance on an original test dataset of IMDb and compare the results with baseline model. We show how downstream tasks can be used to assess the performance of these paraphrase generation models. Furthermore, we investigate different models to create a stable baseline for IMDb movie dataset.

1.4 Contributions

Four main contributions of this work are:

- Evaluation of seq2seq model with attention and different variants of pointer network on the PPDB dataset.
- Trying out different classification models in NLP, to create a stable baseline on IMDb movie review dataset for classification tasks.

- Evaluation of generated paraphrases in the downstream task by augmenting IMDb movie training dataset and comparing with solid baseline created without augmenting the dataset.
- Evaluation of the PPDB dataset for generating paraphrases.

1.5 Thesis Structure

The first chapter is a brief introduction to paraphrase generation task and points out how important it is to generate paraphrases. This chapter also explains the overall goal of the thesis and contributions of this work in natural language processing (NLP) field.

Chapter 2 presents the essential background and significant related work done in paraphrase generation. This chapter explains different kind of models used in generating paraphrases in the past, metrics used to evaluate these models, meaning of having similar descriptions and dataset used in this work.

Chapter 3, presents the explanation and training procedure for our paraphrase generation model on the paraphrase database (PPDB), the comments about different pointer network and attention model we are going to assess in this work.

Chapter 4, presents the training procedure of sentiment classifier on IMDb movie review dataset, performance of different models on sentiment classification task and creating a solid baseline for further comparison.

Chapter 5, presents augmentation of IMDb training data set with different paraphrase model trained on PPDB data set with the help of attention and pointer network. Furthermore, the chapter presents training of text classifier with this newly generated paraphrases and comparing results with baseline.

Chapter 6, presents the summary of the main results of this thesis and future work, which aims at the improvement of the evaluation of paraphrase generation in natural language processing (NLP).

Chapter 2

Related Work

2.1 Paraphrase Theory

A typical normal for human language is the ability to communicate the equivalent information in multiple ways. These dialects are known paraphrases, which in the literature have additionally been referred to as reformulations, restating and other diversity of phenomena.

There can be many variations of paraphrases, but in this work, we try to limit the generation of paraphrases, which can be carried out by linguistic and semantic knowledge to produce similar sentences. Here are some examples[1]

1. **S:** The couple wants to **purchase** a home.

   P: The couple wants to **buy** a home.

2. **S: It was** a Honda **that** John **sold to** Aman.

   P: John **sold** a Honda **to** Aman.

3. **S:** Aman **bought** a house **from** John.

   P: John **sold** a house **to** Aman.

4. **S:** The flood **last year** was a terrible catastrophe in which many people died.

   P: The flood **in 2006** was a terrible catastrophe in which many people died.

5. **S: I want some fresh air.**

[1] **Here 'S' represents the original sentence, and 'P' represents paraphrase of it. Furthermore, bold words are the primary information used in generating paraphrase.**.

P: Could you open the window?

In all the examples mentioned above, we only require linguistic, lexical, referential and structural knowledge to generate paraphrases. Example (1), is generated using knowledge of synonyms words which comes under the lexical category. (2) An example is generated using structural information, which comes under syntactic knowledge. This type of transformation is described in the theory of transformational grammar [19]. Example (3), is an illustration of alternation which can be carried out by syntactic transformation. Example (4), is an instance of referential paraphrase. Example (5), focuses more on pragmatic paraphrase because we can refer that from both the original sentence and paraphrase sentence, "the speaker wants to convey a message to the hearer to open window."

One thing common about all the above-generated paraphrases is that we do not need any domain knowledge or domain is common in both the original sentence and in its paraphrase sentence, i.e., 'English literature.' The things become more tricky when we try to generate paraphrases where the original sentence is in one domain, and we want to generate paraphrase in a domain other than the original domain. Let us see examples of this kind of paraphrases.

6. **S :** Nearest neighbor is good.

   P (Literature Domain) : The closest neighbor is good.

   P (Machine learning Domain) : The closest neighbor is good.

7. **S :** Martel Foresterie Urbaine provides tree cutting and stump removal services for trees in the Montreal area

   P (Literature Domain) : Martel Foresterie Urbaine provides tree pruning and stump removal services for trees in the Montréal area

   P (Machine learning Domain) : Martel Foresterie Urbaine provides tree pruning and stump removal services for trees in the Montréal area

As we can see from the above example when generating paraphrase in one domain for example in 'English literature' (described in sample 6) 'Nearest neighbour' is a synonym of the 'closest neighbour.' However, while generating paraphrase in machine learning domain, it might not be a good idea to convert 'Nearest neighbour'

to 'closest neighbour' as 'Nearest neighbour' holds an individual or reserved meaning in a machine learning context. This means context or domain knowledge is also required in generating paraphrases. Sample 7 also describes one more kind of this paraphrase where 'tree cutting' is converted to 'tree pruning,' which has a special meaning in machine learning domain and hence it is not a correct paraphrase.

In this work, we attempt to focus on evaluating past methods on generating similar sentences using linguistic, lexical, referential and structural knowledge.

2.2 Similarity Measures

Before evaluating the existing methods first, we should clarify the conditions which should be fulfilled by the constructed sentence to be considered as a paraphrase. Here are a few criteria for assessing paraphrases.

1. **Syntax Level:** The first minimal requirement for generating paraphrase from the source sentence is to have a valid syntax in the given language. In other words, it should follow all the syntax rules defined by the given language while generating a paraphrase in natural language generation.

2. **Semantics Level:** The second minimal requirement which must be followed by the generated paraphrase in natural language generation is its meaning and interpretation of output or target sentence. The output sentence must be meaningful.

3. **Lexical Level:** Lexical level is a way to convert characters or words into tokens, which can be identified by machines (numbers). These tokens dimensions capture the several likeness measures of a word, a context of a word and other things. There can be many ways to convert these characters or words into these tokens, for example, n-grams, similarity coefficients, alter remove, etc. This type of measure is useful to find the similarity or to generate more interpretations from a given source sentence. However, in our case, while generating similar sentences, it should also be checked for contextual meaning of source sentence.

4. **Same Meaning:** This main property of paraphrase sentences is to have the same meaning in a given context. To better understand how two sentences can have the same meaning, let us describe two key terms **Connotation** and **Denotation**

   Connotation is the emotional and imaginative association surrounding a word." The connotations for the word snake could include evil or danger.

   "Denotation is the strict dictionary meaning of word. For example, if we look up the word snake in a dictionary, we will discover that one of its denotative meanings is "any of numerous scaly, legless, sometimes venomous reptiles having a long, tapering, cylindrical body and found in most tropical and temperate regions." For example,

   - **Connotation** That man is a cold-blooded **snake**
   - **Denotation** There are a plethora of **snake** species living in Louisiana.

   In other words, the description of the notion in the world is known as denotation, while the absolute meaning, such as nuance and style, is called connotation [15], [22].

   To understand these key terms better, let us consider an example:

   (a) **S** Your cat looks slim.

   P Your cat looks skinny.

   Sentences (S) and a (P) are instances of sentences having different connotations. Both (S) and (P) are well-structured sentences, but (P) is less desirable. This kind of sentences is accepted as a paraphrase throughout our evaluation only if, the word vector of the original and generated word have similar embeddings in a given reference.

   The similarity between two sentences can also be measured on these two constituents.

(a) **Corpus Based** Measuring the sentence similarity using information gain from the large corpus for examples features like Explicit Semantic Analysis (ESA), Latent Semantic Analysis (LSA), Pointwise Mutual Information (PMI) [7], [1], [36].

(b) **Knowledge-based** This measure sentence or word similarity using semantic nature of word for example WordNet, Wikipedia, DBPedia etc. Normally this knowledge is useful for general sentence [5], [6].

In this work, we assess some of the existing methods which have performed well on NMT and text simplification by generating paraphrases in the given context.

2.3 Paraphrase Generation

Paraphrase generation task is substantially similar or is a particular case of neural machine translation (NMT) task in a way, given the source sentence we need to generate an output sentence which has the same meaning. The only anomaly in a paraphrase generation and NMT is that in the former case, output sentence is also in the same language as the source sentence.

Paraphrase generator models are given an input interpretation as a source sentence, and they produce more than one (depending on the beam size) similar interpretations which are then given a score based on some criteria. In general, there are two techniques to generate paraphrases.

2.3.1 Approaches to Paraphrase Generation

In this subsection, we go through the various strategies which are used for paraphrase generation.

Bootstrapping

This method does not need any machine translation. We generate paraphrases using templates. This technique can only be used when the input and output sentences are templates and by applying it on a large monolingual corpus. We start with retrieving the sentences in the corpus that contains seed pairs which match to the template we wish to generate. Filtering techniques are used to filter out candidate paraphrases, which are not useful enough. Next, after obtaining these templates, we look into the corpus again for the sentences which match these new templates. More seed values are extracted from new sentences, and more iterations are used to generate more templates and more seeds. This process is repeated until no new seed can be obtained or limitation on iteration is reached.

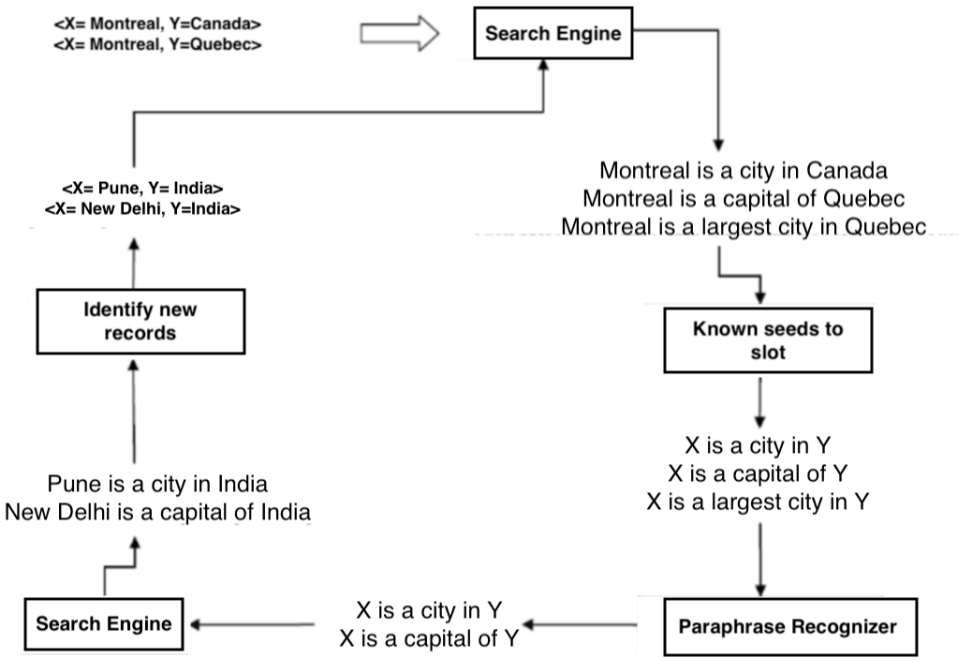

FIGURE 2.1: Generating paraphrases of "X is a city in Y" by bootstrapping [2], [33]

In this method, if the slot values can be identified reliably, then one can obtain initial seed slot values automatically by retrieving direct sentences that match the original templates. There are many well-known methods for bootstrapping; one of them is template extraction anchor set extraction (TEAS). TEAS has been used in many information extraction patterns [2], [27], [52]. There are some other methods which

require corpora annotated with instances of particular types of events to be extracted [21], [48], [8]. One model of this approach is shown in Figure 2.2, [4].

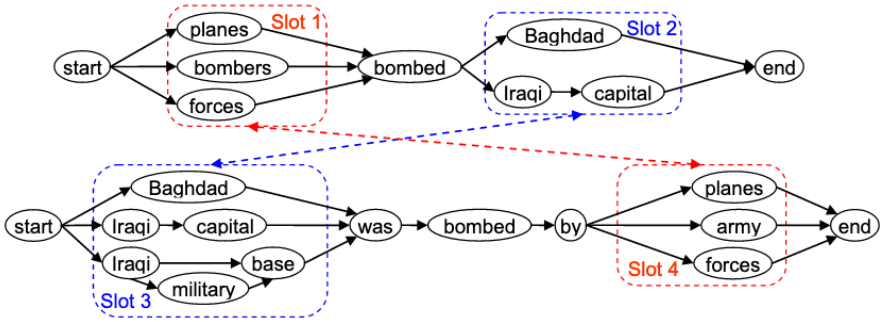

FIGURE 2.2: An example of bootstrapping approach used by Barzilay and Lee [4]. They proposed a strategy to produce paraphrases through the utilization of monolingual parallel corpora, particularly from the news articles covering a similar occasion around the same time. The training comprises of utilizing multi arrangement to create sentence-level rewords from an un-annotated corpus

Statistical Machine Translation (SMT)

As mentioned earlier, the paraphrase generation can be seen as a particular case of the machine translation problem. In general, most of the generation tasks drew an idea from statistical machine translation (SMT), which are based on a large corpus. In SMT, we can define this problem as follows.

Let's take S as a input sentence, whose words are w_1, w_2, w_3 .... $w_{|S|}$ and N is an instance of one candidate translation or in our case it is candidate for good paraphrase which has words a_1, a_2, a_3 .... a_i. If we have more than one instance of such N, our aim is to find the best N^* from the list of N, which has maximum probability of being a translation or paraphrase of S (Source) sentence. This can be represented in form of equation as follows

$$N^* = argmax P(N|S) = argmax \frac{P(N)P(S|N)}{P(S)} = argmax P(N)P(S|N) \tag{2.1}$$

In Equation (2.1), using the conditional probability formula $argmax P(N|S)$ can be further expanded as shown below. The source sentence, i.e., S is fixed, so, $P(S)$

is fixed across all the translations N, hence can be removed from the denominator. $P(N|S)$ is the probability of translation given source sentence. $P(N)$ is the language model which is used to find out the probability of being a correct sentence of output sentences. Also, $P(S|N)$ is probability of translation or paraphrase model [12].

In the candidate sentence, each word probability is dependent on its precedence word. So, the total probability of P(N) becomes:

$$P(N) = P(a_1) * P(a_2|a_3) * P(a_3|a_1, 1_2).....P(a_N|a_{N-2}, a_{N-1}) \tag{2.2}$$

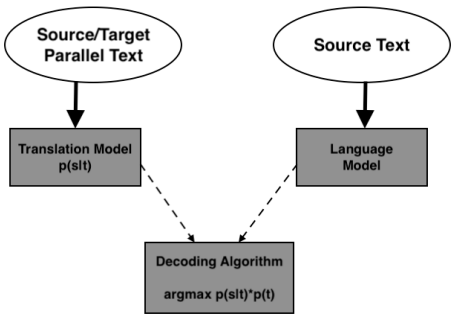

FIGURE 2.3: SMT model

This language model has a smoothing mechanism. Smoothing mechanism is needed to handle cases, where n-grams that are unique or does not exist in the corpus, which can lead the language model where the probability of the whole sentence is 0, i.e., P(N)=0. A diagram of the SMT model is shown in 2.3. There is some progress seen in utilizing long transient memory (LSTM) models to produce paraphrases in this approach [43]. The model consists of encoder and decoder, both utilizing varieties of the stacked remaining LSTM.

To begin with, the encoding LSTM takes a one-hot encoding of all the number of words in a sentence as info and produces a final hidden vector, which can be seen as a portrayal or representation of the full input sentence. The second part of the model, i.e. Decoder LSTM then accepts the concealed vector as initial input and produces new sentence by generating word by word, ending in a finish of-sentence token (generally <EOS>). The encoder and decoder are prepared to take an expression and replicate the one-hot conveyance of a comparing word by limiting perplexity

utilizing straightforward stochastic angle drop. New paraphrases are created by contributing another expression to the encoder and passing the yield to the decoder. More about LSTM is given in section 2.5

These models are ubiquitous and are very generic for a wide variety of different generation tasks in natural language processing, for example, in question answering, paraphrase generation, text summarizing. Also, this is the state-of-the-art in most of the generation task. In this work, we have used these models for generating paraphrases of the input sentence.

Parsing

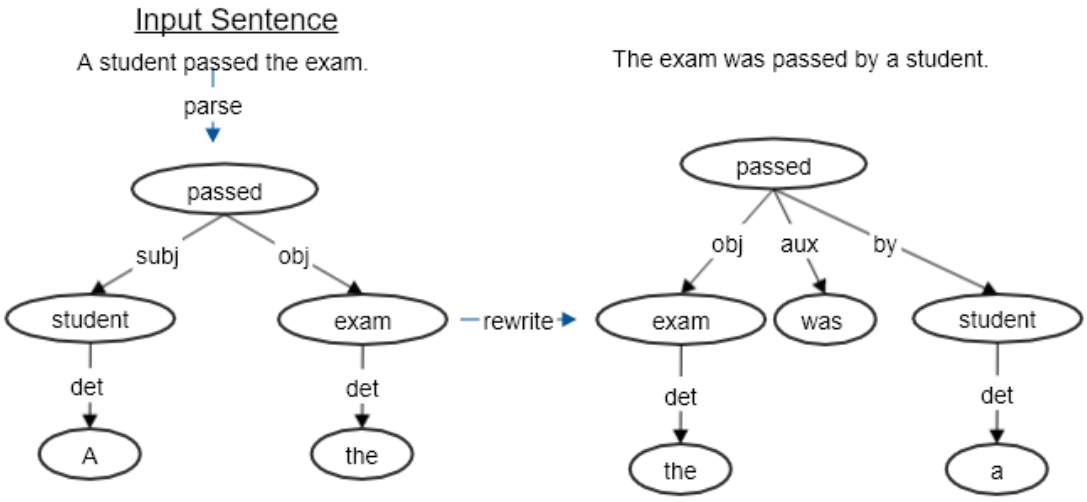

FIGURE 2.4: An example of parsing based approach for generation of paraphrase

Syntactic transfer in machine translation may also be used [35] to generate paraphrases. In this approach, we first need to parse the input expression. Then to generate output paraphrase sentence, these parse tree or expression are modified in a way which preserves the meaning of the syntax. There are some errors induced while parsing the input sentence. An example of this approach is shown in 2.4.

Phrase-based Machine Translation

One more way to generate paraphrases is to convert source sentence to a different language, also known as the pivot language and then translate back to the original

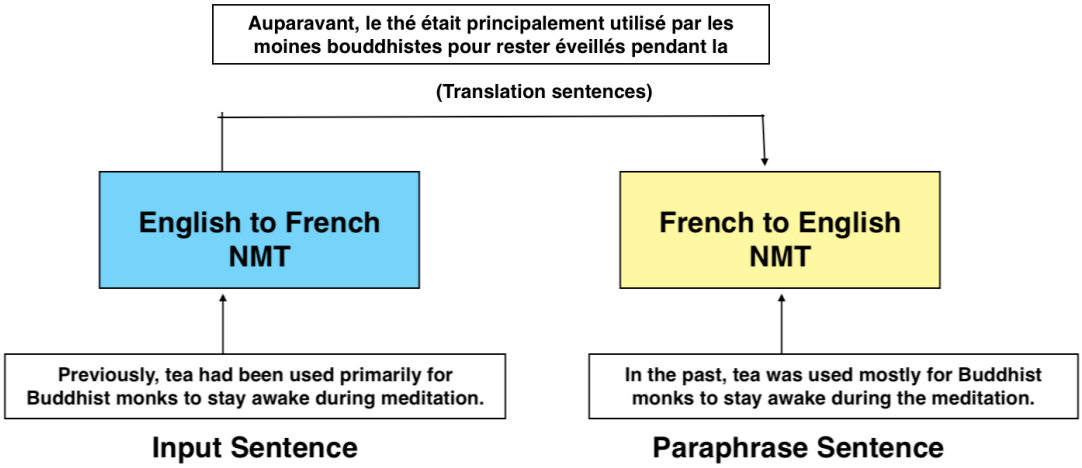

FIGURE 2.5: An example of Phrase-based Machine Translation for generation of paraphrase

language. The target sentence usually is very different from the source sentence, primarily when different translation models are used [14]. The main advantage of this approach is that there are many research advancements done in this translation systems whose model can be directly used as a black box. It also has one main disadvantage, namely cumulative error for both the translations. An example of this approach is shown in 2.5.

2.3.2 Creation of Paraphrase Dataset

As stated earlier in section 2.3, SMT models need a large amount of corpus data to train the model before generating output candidate sentences. To make this large corpus, we need a tremendous amount of data. This data is fetched from a different news article, social media sites, newspapers, and other text sources. Next, from this data, we create sentence pairs which then get annotated by a human being manually or by models which are good at paraphrase identification task. These sentences are then used to create word alignment table for generation task as in phrase-based SMT system [25]. This alignment table includes synonyms which could be in the same language or a different language depending on the generation task. Phrase pairs that frequently occur in the aligned sentences may be assigned higher probabilities [44].

2.4 DataSet

2.4.1 The paraphrase database

In this work, we have used the paraphrase database (PPDB) [40], [41] for training the paraphrase generation model. PPDB is a database containing millions of paraphrases in 16 unique languages. The objective of PPDB is language enhancement by making frameworks increasingly sensitive to language inconstancy and obscure words. The total PPDB asset is openly accessible under the Creative Commons Attribution 3.0 United States License. The paraphrases are generated by extracting lexical, phrasal, and syntactic approaches from large bilingual parallel corpora and computing the similarity scores for the pair of paraphrases using Google n-grams and the Annotated Gigaword corpus. This data set is created following basic idea, that if two English strings e1 and e2 translate to the same foreign string f known as a pivot language, they should have the same meaning and should be a paraphrase of one another. It collects many different features between sentence pair like n-gram based features for words (to the left and right of the given phrase), Lexical, lemma-based, part of speech (POS) and named entity unigrams and bigrams; Dependency link features and Syntactic features. Then all these are aggregated, over all the occurrences of e, to obtain distributional signature $s_{e,i}$. Then the cosine similarity score is calculated between two paraphrases by the formula:

$$sim(e_1, e_2) = \frac{s_{e,1}, s_{e,2}}{|s_{e,1}||s_{e,2}|}$$

here, $\overrightarrow{s_{ei}}$, is The PPDB dataset comes in a plain text file in the following format.

$$LHS|||PHRASE|||PARAPHRASE|||(FEATURE = VALUE)*|||ALIGNMENT|||ENTAILMENT$$

Here PHRASE is an expression which is considered as a source sentence, PARAPHRASE is a paraphrase of PHRASE and considered as a target sentence, LHS is the constituent or CCG-style slashed constituent label for the paraphrase pair. ENTAILMENT is an automatically assigned entailment relation (e.g., Equivalence for pairs

like couch/sofa, or forward entailment for pairs like dog/animal) [39]. This dataset consists of 169.9 million sentence pairs.

2.4.2 IMDb Dataset

We use IMDb dataset [32] for augmenting its training samples and then we do the sentiment analysis on the sentences generated by our different paraphrase generation models. The IMDb dataset is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. This dataset provides a set of 25,000 highly polar movie reviews for training and 25,000 for testing. This dataset consists of movies URL, reviews, and labels, whether it is positive or negative. In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. In the labelled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets.

2.5 Recurrent Neural Networks

Recurrent neural networks (RNN) has shown lots of extraordinary feats of deep learning in the last couple of years. In many generation tasks, RNN is the state of the art model. Many examples can be seen which are built using RNN like Apple's Siri, Google voice search, Google translators, etc. The main difference which makes RNN suitable to machine learning issues that include sequential tasks is its internal memory, which can store results from the last precedent stage and can use it at later stages for inference. In this section, we discuss its functionality, how it works, advantages and disadvantages.

Recurrent Neural Networks (RNN) are incredible and powerful neural-like systems and have a place with the most promising algorithm out there right now since these models contain internal memory.

Initially, at the time when RNN's were developed in the 1980s, they did not get immediate notoriety because of limited data and limited computational power at that time. They got popularity late with the expansion of computational power and a vast amount of information that we have these days and the development of LSTM in the 1990s.

RNN internal memory is responsible for storing valuable things about all the precedent inputs they have received, which helps or enables them to be very accurate while predicting the next step. This helped RNN to become state of the art algorithm in most of the sequential data problems in machine learning like audio, video, text, speech as they were better in capturing the context and deep understanding of sequential data.

Now the question is when to use RNN? According to Lex Fridman (MIT):

"Whenever there is a sequence of data, and that temporal dynamics that connects the data is more important than the spatial content of each frame."

So the question is what leads to RNN when we have a feedforward neural network and how RNN works?

In a feed-forward network, information flows from the input layer to the hidden layer through activation units and then the output is given. Every node in the feed-forward network is only visited once for every data, and information moves straight through the network. There was no feedback (loops), due to which, there is no way for the accountability of previous information while predicting the next step. To solve this, RNN was developed.

As quoted earlier, the main difference in RNN is that they have internal memory, which can store the results from last precedence state. In RNN the information cycles through the loop as seen in Fig 2.6. So while deciding step y_{t+1}, it also takes the accountability of whatever the model has learned until the current step that is still y_t. For example, in the feed-forward network if we try to send the sentence *'My name is John'* word by word, when the model is processing word *'name'* the model has already forgotten about the word *'My'* and same for other next words in a sentence. However, in case of RNN while processing word *'name'* it will still have the

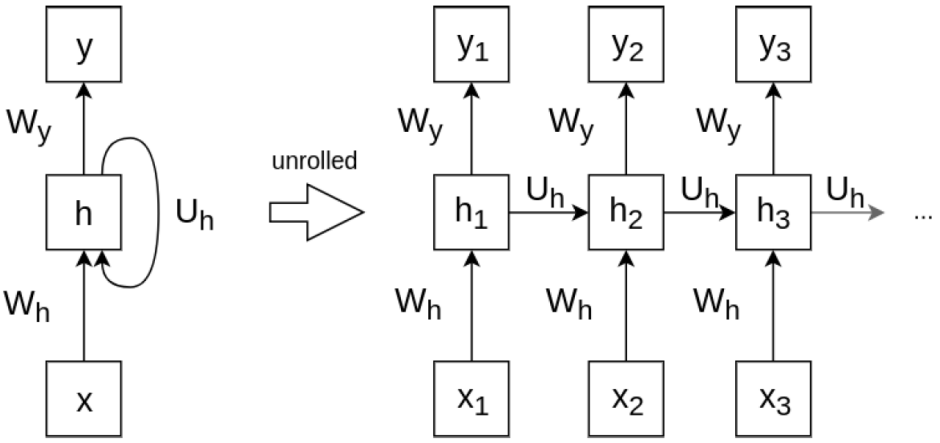

FIGURE 2.6: Here in left hand side is the diagram of RNN unit and on right hand side is RNN in unrolled state. The weight vector W_h , W_y and U_h are shared across all the time and are fine tuned using backpropogation and h_i is the the last output that is y_{t-1}. In this graph we are trying to predict y_t which will be equal to x_{t+1}

embeddings from the last word that is *'My'* and the output at this time will be the combination of embeddings of both *'My'* and *'name'*.

Looking at the Fig 2.6 we can see at every time step t RNN takes two inputs the current input and the past output from last precedence state, which makes it suitable for sequential data processing because sequence contains essential information which must be used by the model for predicting next output. RNN has two weight vectors which are applied to input vector and previous output vector respectively. Both these weight vectors are tweaked at training time using gradient descent and backpropagation.

Feedforward neural network is only able to map one input to one output. However, RNN can map one-to-one, one-to-many, many-to-one (voice recognition) and many to many (Translation), as shown in Fig 2.7.

There are many different variations of RNN, and in the purest form, RNN is known as Vanilla RNN. All the variations have there own advantages and disadvantages. The original RNN which was introduced in NLP has this form [46].

$$h_t = f_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.3}$$

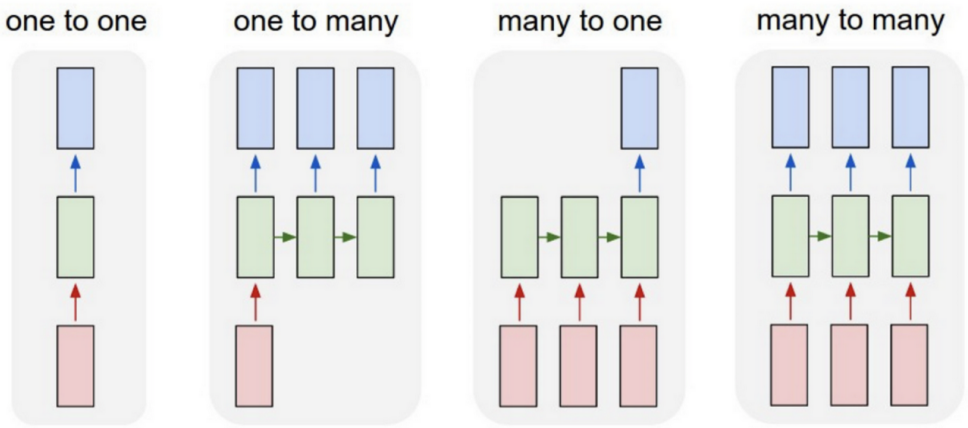

FIGURE 2.7: Different types of configuration in RNN.

$$y_t = f_y(W_y h_t + b_y). \tag{2.4}$$

Here x_t is the input vector, h_t is the hidden state, y_t is the output vector, W, U are trainable parameter matrices, and b is a bias which is also a trainable vector. Functions f_h and f_y are non-linear activation functions such as sigmoid or softmax. The corresponding structure is shown in Fig 2.6.

In RNN the total output is generally equal to the total number of words assuming we give input word by word. Which means the target sentence length must be the same as of source sentence. In reality, this constraint doesn't work for example in paraphrase generation most of the time the target sentence has different length as of source sentence. So for these sorts of generation task Encoder-Decoder neural network approach works best.

Although RNN works well for this kind of task, there are two main issues with vanilla RNN.

1. **Exploding Gradients** when the calculation relegates a moronically high significance to the weights, without any good reason. Because of this, the value of weights become very high, and the network becomes unstable.

2. **Vanishing Gradients** when the value of gradient becomes too small that model stops learning or takes too much time. This was the major problem in the early '90s, which was later solved by long short term memory (LSTM) [20]

The main idea for **Long short term memory** (LSTM) network was to store relevant information from past experiences, unlike storing everything like in Vanilla RNN. LSTM only needs to remember essential experiences from the past, which also makes LSTM remember longer sentences than Vanilla RNN. The LSTM unit is used to build RNN for better performance.

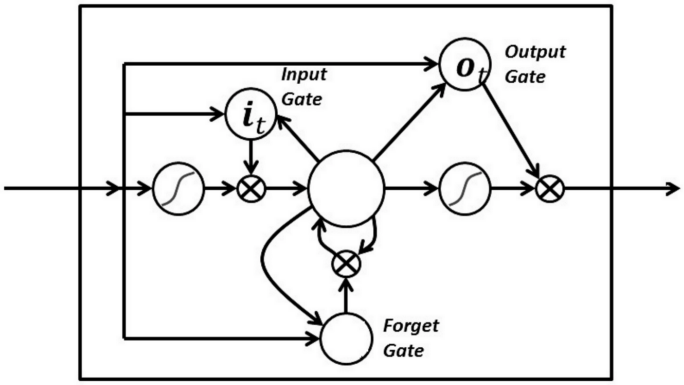

FIGURE 2.8: LSTM

As seen in Fig 2.8 LSTM has three gates input gate, output gate and forget gate, which helps LSTM to remember only the necessary information and forget whatever is not relevant to the task. The gates in LSTM are analog, in the form of sigmoids, meaning that they range from 0 to 1. The fact that they are analog enables them to do backpropagation with it.

Another unit, i.e., **Gated Recurrent Unit** (GRU) which was introduced in [11], also aims to solve the problem of vanishing gradient problem. This varies from LSTM network in the way, how different gates were implemented in this unit. It only has two gate update and reset gate in comparison to LSTM, which has three gates.

An example of GRU is shown in Fig 2.9. Here, the update gate (Equation 2.5) and the reset gate (Equation 2.6) are calculated at each time step to store only the necessary information and forget whatever is not relevant to the task.

$$z_t = \sigma(W^Z x_t + U^Z h_{t-1}) \tag{2.5}$$

$$r_t = \sigma(W^r x_t + U^r h_{t-1}) \tag{2.6}$$

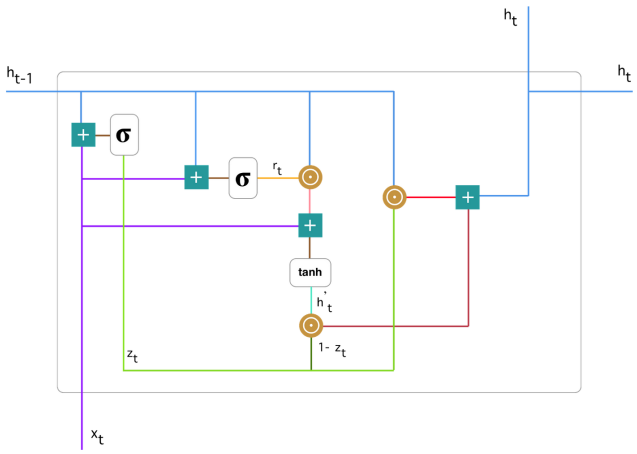

FIGURE 2.9: GRU

Here, x_t is a input vector, U^Z, W^Z, U^r, W^r are weight vectors learned by doing back-propogation at training time, h_{t-1} is a output from the last step and t denotes the time at current step and σ represents sigmoid activation function.

$$h'_t = tanh(Wx_t + r_t \odot Uh_{t-1}) \tag{2.7}$$

The equation 2.7 refers to the current memory content (h'_t) where r_t value is for re-setting the information and represents how much we should reset from the past information. In equation 2.7, the symbol $\odot$ represents the Hadamard element-wise product.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{2.8}$$

The last, equation 2.8 refers to the final memory content after applying update gate. Here, z_t is value for update gate, and the rest is the same as before.

Uses of these gates LSTM, GRU helps solve vanishing gradients in RNN because they always keep the gradients high enough and therefore the training is relatively short and the accuracy high.

2.6 Convolutional Neural Networks

Convolutional Neural network (CNN) is a deep neural network which is most commonly used in computer vision for images recognition, images classification. Objects detection, recognize faces. The idea of CNN was inspired by the study done on the visual cortex of mammals and how they perceive the world using a layered architecture of neurons in the brain. These models can be thought of as a group of neurons which specialize in the detection of different objects. Yann leCun [29] was first to take this idea to develop CNN. The model diagram is shown in Fig 2.10. Usually, ConvNets are used to look for features in an image. Thus we do not need to provide features implicitly. CNN understands the right features by itself as it goes deep.

The ConvNets are made up of three main elements.

1. **Convolution layer** This layer is normally responsible for capturing low-level features of object or image. A filter of height 'h' and width 'w' is taken (in general filter is just a matrix or vector.), and, this filter is rolled over the matrix of the given image. More concretely, at a given position of the convolution filter, we take the element-wise multiplication of each filter cell value with the corresponding image pixel value that overlaps the filter cell, and then take the sum of that. The formula is shown below:

$$h_{i,j} = \sum_{k=1}^{m} \sum_{l=1}^{m} w_{k,l} x_{i+k-1,j+l-1}$$

2. **Pooling layer** This layer makes CNN little bit translation invariant in terms of the convolution output. There can be many different techniques for pooling, but the most common is max-pooling. We apply max pooling layer to every vector we got by applying a convolutional layer only to consider the features which are important for the task and avoid the extra computational cost by neglecting features which are not useful. This can be calculated using the following formula.

$$h_{i,j} = max\ x_{i+k-1,j+l-1,} \forall 1 \leq k \leq m\ and\ 1 \leq l \leq m$$

3. **Fully-connected layer** This layer learns the features learned by different convolutional filters in the model to build a global representation of the holistic image. The neurons units in the fully connected layer get activated based on whether various entities represented by convolution features are present in the inputs. As the fully connected neurons get activated for this, it produces different activation patterns based on what features were present in the input images.

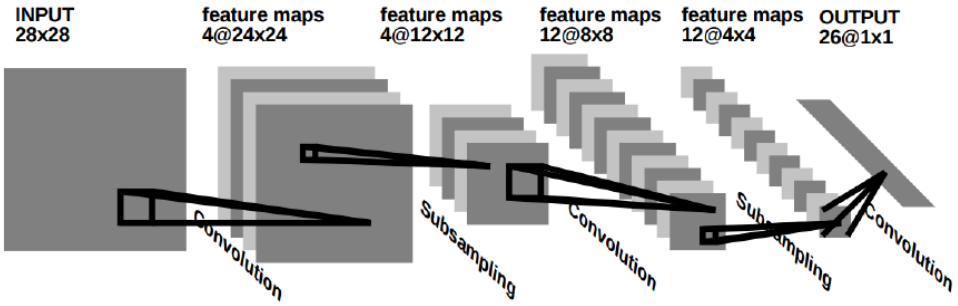

FIGURE 2.10: CNN model for image processing. [28]

In this work, we use a convolutional neural network for the classification of sentences generated by paraphrase generation model on IMDb movie dataset.

2.7 Evaluation Metrics

In this work, we evaluate the performance of our paraphrase generation models with measures like perplexity, accuracy, Bilingual Evaluation Understudy (BLEU) and Metric for Evaluation of Translation with Explicit Ordering (METEOR) [38], [26]. Accuracy is merely a measure of the number of correctly classified instances in a given epoch, irrespective of their relevancy, giving equal importance to positive samples and negative samples. As in paraphrase generation model, it is a difficult task to decide when a target sentence is an accurate paraphrase of the given source sentence without individual evaluation who have expertise in this domain. However, individual evaluation sometimes causes a delay in the evaluation process, and it has its disadvantages like biases, is time-consuming and also needs experts in the same domain.

Perplexity measures the cross-entropy between the empirical distribution (the distribution of things that appear) and the predicted distribution (what your model likes) and then divides by the number of words and exponentiates after throwing out obscure words. The perplexity of a discrete probability distribution p is defined as,

$$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)} \tag{2.9}$$

Here, H(p) is the entropy (in bits) of the distribution and x ranges over events.

In our instance, the paraphrase generation model of an unknown probability distribution p will be proposed based on the training samples that are drawn from p. Given a proposed probability model q, one may evaluate q by asking how well it performs in predicting test samples $x_1, x_2, ..., x_N$, drawn from probability distribution p. The perplexity of the model q is defined as,

$$b^{-\frac{1}{N} \sum_{i=1}^{N} \log q(x_i)} \tag{2.10}$$

Here, b is customarily 2. Better models q of the unknown distribution p tend to assign higher probabilities $q(x_i)$ to the test events. Thus, they will have lower perplexity. Which means they are less surprised by seeing the test sample.

The exponent above may be regarded as the average number of bits needed to represent a test event x_i if one uses an optimal code based on q. Low-perplexity models do better job of compressing the test sample, requiring few bits per test element on average because $q(x_i)$ tends to be high. The above exponent can be treated as cross-entropy.

$$H(\bar{p}, q) = -\sum_x \bar{p}(x) \log_2 q(x) \tag{2.11}$$

Here, $\bar{p}$ denotes the empirical distribution of the test sample.

Bilingual Evaluation Understudy (BLEU) is a metric used for comparing the output translation with a given candidate reference sentences. Its score varies from 0.0 to 1.0, depending on the quality of the translation. If the translation is perfect, then it is

given a score of 1.0, and when a translation is not relevant, then it is given a score of 0.0. This metric is not limited to only translation task but can also be used in other natural language processing tasks like paraphrase generation, text summarization and other language generation tasks.

BLEU was proposed in [38]. Some of the benefits of using BLEU as a metric are as follows: it is easy to calculate, highly correlated to human evaluation, independent of language and easy to understand. It is adopted widely as a metric in different NLP tasks.

The BLEU score is calculated by summing up all the matching n-grams in candidate sentence and n-grams in references sentences where n-gram (where n represent the number of words considered at a time) can be unigram (1-gram), bigram (2-gram) and so on. The main thing to note here is that this comparison is made regardless of word order.

As stated in [38]:

"The primary programming task for a BLEU implementor is to compare n-grams of the candidate with the n-grams of the reference translation and count the number of matches. These matches are position-independent. The more the matches, the better the candidate translation is."

To show how to calculate BLEU score, let us take one reference sentence and one paraphrase sentence, where we are evaluating the BLEU score of paraphrase sentence given the reference sentence.

1. **Reference Sentence:** Brad came to dinner with us.
2. **Paraphrase Sentence:** Brad ate dinner with us.

The one gram-BLEU score of the candidate solution in the above example will be 0.667 as we have four unigrams in paraphrase sentence which are also present in the reference sentence, i.e. Brad, dinner, with, us. Moreover, we have six total number of words in the reference sentence.

Metric for Evaluation of Translation with Explicit Ordering, or in short METEOR, is also a viral measurement for evaluating machine translation output. It was first proposed in [26].

The METEOR is calculated by taking harmonic mean of unigram recall and precision, with recall weighted higher than precision. It first creates the alignment table of unigram words in the candidate sentence and reference sentence with constraint, every unigram in candidate sentence must map to zero of one unigram in reference sentence. This process is repeated consecutively to create a final alignment table by removing unigrams formed at the previous step.

The problem with BLEU measurement is that it does not take account of orders of the words in the sentence. However, METEOR considers the order of words and features like stemming and synonyms matching. It internally uses wordnet. METEOR has shown a better correlation with human judgment than BLEU metrics on the same dataset. The correlation of METEOR with human judgment went up to 0.964 on some dataset in comparison to 0.817 with BLEU measurement.

While performing a binary classification task, we measure the performance of our classifier with measures like **accuracy** 2.12, **F1-Score** 2.15, **sensitivity or recall** 2.14, and **precision** 2.13.

$$AC = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.12}$$

$$PR = \frac{TP}{TP + FP} \tag{2.13}$$

$$RC = \frac{TP}{TP + FN} \tag{2.14}$$

$$FS = \frac{2 * PR * RC}{PR + RC} \tag{2.15}$$

Where,

- True Positives (TP):- Examples which are positive and correctly classified by the classifier.
- True Negative (TN):- Examples which are negative and correctly classified by the classifier.
- False Positive (FP):- Examples which are negative and wrongly classified as positive by the classifier.
- False Negative (FN):- Examples which are positive and wrongly classified as negative by the classifier.

In this work, we use perplexity as a measure on test data set to compare different paraphrase generation models and to use the best one to generate paraphrase for IMDb movie data set.

2.8 Summary

In this chapter first, we reviewed the literature of paraphrase theory, what paraphrase is, and different type of paraphrases. Then we look at some of the different types of similarity measures we can use to detect paraphrases, i.e., similarity measures like syntax, semantics and lexical. After that, we discussed the previous and current approaches for generating paraphrases in natural language processing like bootstrapping, statistical machine translation, phrase base, and parsing methods. We also looked at how neural approaches later become the standard way of generating the paraphrases because of their advantages.

One of the most notable disadvantages of a neural approach for generating paraphrases was the requirement of a large corpus. Then we studied how these large corpora are made for training these big neural models. Lastly, we went through the basic theory of RNN's and CNN's, how they work, and what are the evaluation metrics we use to evaluate different models.

Chapter 3

Encoder-Decoder models for Paraphrase Generation

3.1 Encoder-Decoder RNNs for NLP

Encoder-decoders are the neural network approaches, which are genuinely ongoing models for deep learning in NLP. These models in some cases outperform classical statistical machine translation methods. The Encoder-Decoder architecture has become an effective and standard approach for both neural machine translation (NMT) and sequence-to-sequence (seq2seq) prediction tasks which involve task like paraphrase generation, question answering and language modeling. Encoder-decoder mainly consists of two parts. First, encoder to encode input sentence into a context vector and second, decoder which decodes the context vector to output sequence. The key advantage of seq2seq model is the capacity to train a solitary end-to-end model right on the source and target sentences, and the capacity to deal with sentences of variable length to yield sequence of content. They were first presented autonomously as RNN encoder-decoder [11] and sequence-to-sequence [49] for machine interpretation. This group of encoder-decoder models are regularly alluded as seq2seq, regardless of their particular execution. The seq2seq model tries to learn the conditional distribution given as:

$$p(y_1, y_2, ..., y_{T'} | x_1, x_2, ..., x_T) \tag{3.1}$$

where, y is the output sequence conditioned on the input sequence x or source sequence. $y_{T'}$ denotes the word generated by the model at time step T', T' is the length of the output sentence and T is the length of the input sentence. T' and sequence length T are not necessarily same. A seq2seq model first encodes the entire variable x input with its encoder RNN into a fixed size vector c known as context vector. Then, a decoder RNN generates output $y_1, ...; y'_T$ conditioned on previous predictions and context vector c:

$$p(y_1, y_2, ..., y_{T'}|x_1, x_2, ..., x_T) = \prod_{t=1}^{T'} p(y_t|y_1, y_2, ..., y_{t-1}, c) \tag{3.2}$$

There are two different ways to define dependency of output sequence y on context vector c. [49] proposes to condition y on c at the first output from the decoder. And [11] proposes to condition every generation of $y_{T'}$ on the same context vector c, thus forming the basis to our model. For the simplicity, we modify equation for vanilla RNN version to get the hidden state s at time step t, denoted by s_t. Modifying 2.3 leads to:

$$s_t = f_h(W_s x_t^d + U_s s_{t-1} + Cc + b_s) \tag{3.3}$$

Here and elsewhere C is a parameter matrix.

By modifying fig 2.6 we obtained 3.1 with encoder and decoder functionality, and the full setup is known as encoder-decoder or seq2seq model.

Work was done in [49] shows that performance of seq2seq model while generating text can be improved by giving the input sentence in reverse order. The framework accomplished a bilingual evaluation understudy (BLEU) score of 34.81, which is a decent score contrasted with the standard score achieved with a statistical machine translation system of 33.30. This is the first case of a neural machine interpretation framework that defeated a phrase-based statistical machine translation baseline on a large scale problem. However, this work has not accounted for reverse order of sentences.

The encoder-decoder framework has one disadvantage, which is as the length of a sentence increases the performance of seq2seq model decreases [11], [10].

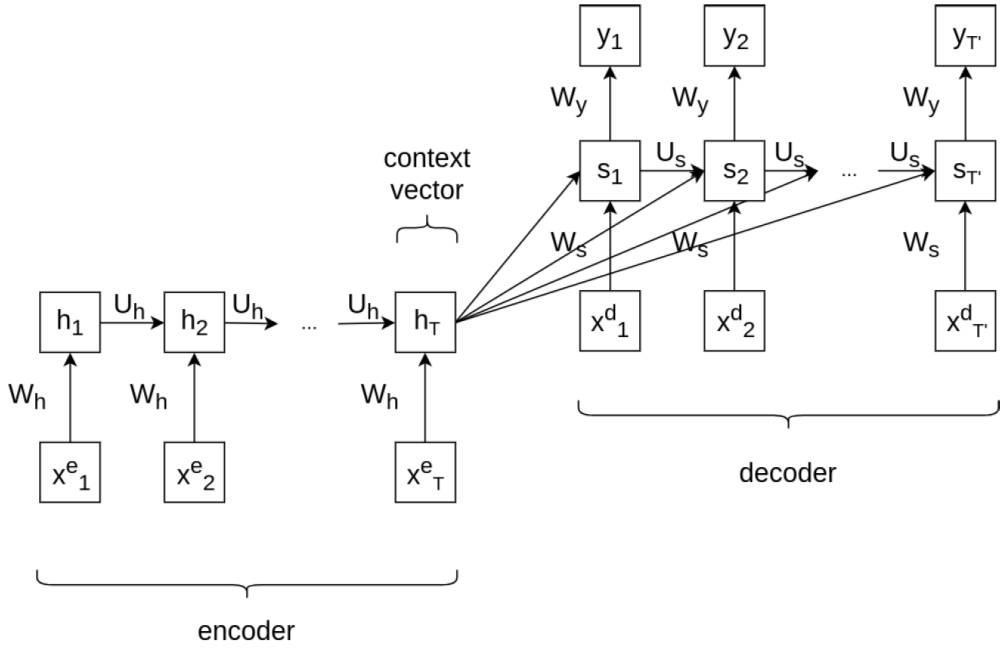

FIGURE 3.1: seq2seq model with x^e as a input and h_t as a hidden state of encoder RNN. And for the decoder x^d as a input, s_t as a hidden state and y_t as a output. Note that x^d_t will be equal to y_{t-1}. The h_t represents the context vector which will be last hidden unit from encoder and given as a output to decoder. This context vector will be used while generating the first word in decoder network.

Fig 3.2 shows, how the performance of seq2seq model decreases with an increase in the total number of words in a sentence. Here, we can see how the BLEU score decreases as the length of the input sentence increases. We present attention mechanism in next section 3.2, which helps the decoder network only to focus particular part of the source sentence while generating output. This method significantly improves the performance of seq2seq network on long sentences [3], [31].

We also use other variations of RNN like long short term memory (LSTM) or Gated Recurrent unit (GRU) for better performance on long sentences.

3.2 Encoder-Decoder with Attention

In this section, we present the attention mechanism to improve the poor performance of seq2seq model on longer sentences [3], [31].

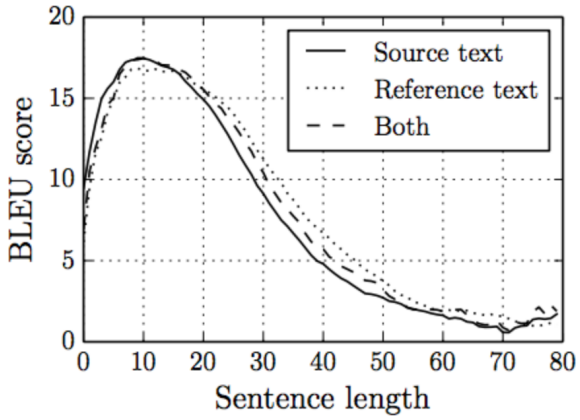

FIGURE 3.2: Performance of encoder-decoder network as the sentence length increases.

The problem occurs while generating the word in decoder network. It looks at the entire input sentence every time while generating a new word. The basic concept of attention is to only focus on a particular part of the sentence. Each time the model predicts an output word, It only uses parts of an input where the most relevant information is concentrated instead of an entire sentence. In other words, it only pays attention to some input words.

In Fig 3.3, we can see that the basic structure of the model is the same as shown earlier in fig 3.1. However, the main difference after adding attention mechanism in seq2seq model comes when generating the next word in the decoder network. In seq2seq model with attention mechanism, we determine the hidden state of the decoder at the current time by taking the previous output, previous hidden state and context vector. Further, note that here we are not using the single context vector c for generating all the words in the decoder network, but a separate context vector c_i for each target word $y_{T'}$.

The encoder first encodes input sentence represented by its word embedding sequence x, into a single context vector c (which is a combination of all the hidden units in encoder and represented by $c = q(h_1, ..., h_{T_x})$) and a hidden state $h_t = f(x_t, h_{t-1})$. Typically decoder network predicts the sequence by predicting one word at a time denoted by y_t, where each y_t output is conditioned on previous outputs $y_1, ..., y_{t-1}$

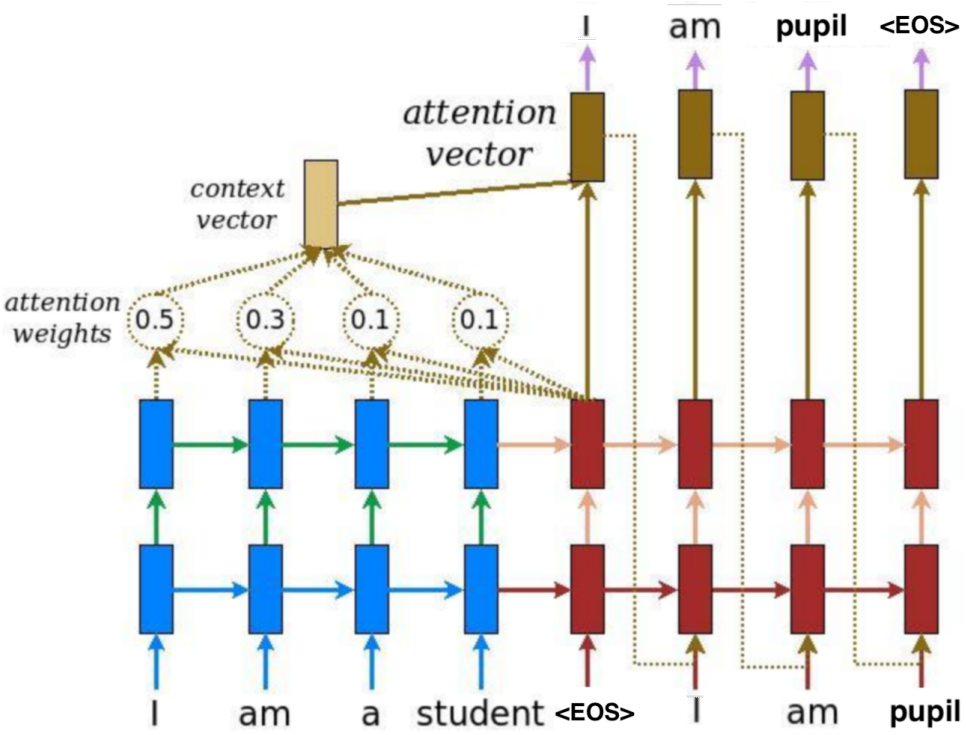

FIGURE 3.3: An illustration of the attention mechanism (RNNSearch) proposed by Bahdanau, 2014. Instead of converting the entire input sequence into a single context vector, we create a separate context vector for each output (target) word. These vectors consist of the weighted sums of encoder's hidden states.

and the context vector c, maximizing the following joint probability:

$$p(y) = \prod_{t=1}^{T'} p(y_t|y_1, ..., y_{t-1}, c) \tag{3.4}$$

In the context of RNNs, the conditional probability of each y_t in the joint probability of Equation 3.4 is modeled as a nonlinear function g with input y_{t-1} context vector c and hidden state s_{t-1}:

$$p(y_t|y_1, ..., y_{t-1}, c) = g(y_{t-1}, s_{t-1}, c) \tag{3.5}$$

Then [3] proposes to use unique vector c_t for each decoding time step, redefining the decoder conditional probability for each word y_t as:

$$p(y_t|y_1, ..., y_{t-1}, x) = g(y_{t-1}, s_{t-1}, c_t) \tag{3.6}$$

where the context vector c_t is a weighted sum over all input hidden states $(h_1, ..., h_T)$:

$$c_t = \sum_{j=1}^{T} a_{tj} h_j \tag{3.7}$$

Here, attention weights a_{tj} are calculated as:

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T_x} exp(e_{tk})} \tag{3.8}$$

$$e_{tj} = a(s_{t-1}, h_j) \tag{3.9}$$

Here the scoring function 3.9, is a pair-wise scoring function which is used for scoring the relation between decoder hidden state s_{t-1} and encoder's hidden state h_j. This scoring is learned jointly while training the whole seq2seq model with the help of a feedforward network.

There are many different kinds of attention mechanism, out of which in this work we have tried two different variations proposed in [31] and [3]. Moreover, we have used seq2seq model with an attention mechanism to compare with seq2seq model with pointer network in generating similar sentences.

3.3 Encoder-Decoder with Pointer network

Encoder-Decoder network also suffers from two other problems, which are reproducing factual details or unknown words or rare word inaccurately, and also they tend to repeat themselves by generating the same words again and again. The second problem can be taken care of by attention and coverage mechanism [50].

Typically, when we create a seq2seq model when we have to define the maximum number of vocabulary length, which is represented by their word embedding. Usually, this vocabulary length varies from 10,000 - 50,000, containing the top maximum number of most frequent words. Further, note that an increase in the maximum length of vocabulary also increases the computation of seq2seq model and make the training process slower. All the other words or token which are not incorporated under vocabulary are marked as '<UNK>' which means unknown words, all these

tokens have the same word embedding. Therefore whenever decoder is generating an output word of embedding <UNK> token, then decoder outputs <UNK> as a token. This is known as unknown words problem and can be very problematic in the case of paraphrase generation. To solve this unknown words problem, we use pointer network in seq2seq model.

There are many variations of implementation of these pointer network and copy mechanism which can be implemented in seq2seq model for better performance, some of them are discussed in [51], [17], [47], [9], [16]. However, In this work, we have evaluated [16] and [17] under the supervision of PPDB dataset. The reason for using these two model is because of there capability of competing between a word from common vocabulary and input sentence vocabulary based on some variables which makes them suitable for generating better similar sentences according to the given context.

In this work, we have compared attention model described in previous section 3.2 with the pointer network implementation specifically purposed in [17] and in [16] on generating similar sentences task.

3.3.1 COPYNET Network

CopyNet was proposed in [16] to incorporate copying mechanism in seq2seq model. This mechanism has shown good results on text summarization tasks on different datasets. The model architecture is shown in Fig 3.4.

The model uses bidirectional RNN as an encoder which transforms or encodes the variable length of the sentence into a fixed size of context vector. It has the same setup as proposed in [3]. The difference comes how model copies words from the input sentence in a decoder network.

As mentioned in work, the model uses the same canonical RNN decoder, which was earlier proposed in [3] with the following differences.

1. **Prediction** It has two modes, namely generate mode and copy mode. Generate mode generates a word from fixed sized model vocabulary and copy mode

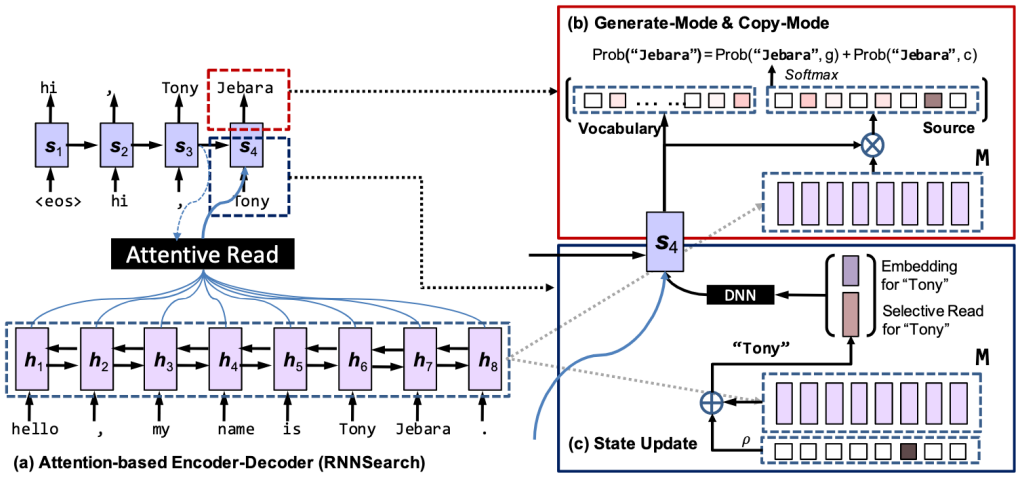

FIGURE 3.4: Architecture of COPYMET network.

generates a word from the vocabulary of the input sentence. While predicting the next word, it uses mixed probabilities from both the mode.

2. **State Update** It uses word embedding of the previously generated word by the decoder network and also the corresponding location specific hidden state in a series of hidden state of the encoder.

3. **Reading** Also, besides of attentive read to the combination of hidden layers of encoder network, It also has "selective read" to the combination of hidden layers of encoder network.

The model has two sets of vocabulary $V = V_1, ..., V_n$ which also includes <UNK> for out of vocabulary (OOV) words and source vocabulary $A = a_1, ..a_N$ which includes unique vocabulary from input sentence. Source vocabulary makes COPYNET to copy OOV words in the output sentence.

At time t, the decoder RNN state is represented by s_t. The probability of a generated word y_t is given by

$$p(y_t|s_t, y_{t-1}, c_t, H) = p(y_t, g|s_t, y_{t-1}, c_t, H) + p(y_t, c|s_t, y_{t-1}, c_t, H) \tag{3.10}$$

where H is a combination of hidden states of the encoder network, c_t is a context vector at t. g stands for generative mode and c stands for copy mode, these probabilities

are calculated as follow:

$$p(y_t, g|.) = \begin{cases} \frac{1}{F} e^{\Psi_g(y_t)} & y_t \in V \\ 0 & y_t \in A \cap \bar{V} \\ \frac{1}{F} e^{\Psi_g(UNK)} & y_t \notin V \cap A \end{cases}$$

$$p(y_t, c|.) = \begin{cases} \frac{1}{F} \sum_{j:x_j=y_t} e^{\Psi_c(x_j)} & y_t \in A \\ 0 & \text{otherwise} \end{cases}$$

Where F is a normalization term, and $e^{\Psi_c(.)}$ and $e^{\Psi_g(.)}$ are scoring functions for copy mode and generate mode respectively. Because of the shared normalization term both generate mode and copy mode probabilities are competing through softmax function 3.10. The scoring functions are calculated as follow:

$$\Psi_g(y_t = v_i) = v_i^T W_o s_t, v_i \in V \cap UNK \tag{3.11}$$

and,

$$\Psi_c(y_t = x_j) = \sigma(h_j^T W_c), x_j \in X \tag{3.12}$$

where X is an input sentence, x_j is a word at j position, v_i and W_0 are one-hot indicator vector for word v_i from the vocabulary. σ is a nonlinear activation function.

COPYNET updates decoder RNN state at every time step t, using previous hidden state s_{t-1}, predicted word y_{t-1} and context vector c as follows:

$$s_t = f(y_{t-1}, s_{t-1}, c) \tag{3.13}$$

However, if y_{t-1} is copied over to the output sentence then the decoder RNN states are updated by changing y_{t-1} to $[w(y_{t-1}); S(y_{t-1})]^T$. Where $w(y_{t-1})$ is the word embeddings of y_{t-1} and $S(y_{t-1})$ is the weighted sum of hidden states in H or in encoder RNN network corresponding to y_t.

$$S(y_{t-1}) = \sum_{r=1}^{T} ptrh_r \tag{3.14}$$

$$ptr = \begin{cases} \frac{1}{K} p(x_i, c|s_{t-1}, H) & x_i = y_{t-1} \\ 0 & \text{otherwise} \end{cases}$$

Here k is a normalizing term. Pointer network (ptr) is only concentrated on one location from source sentence. Although, $S(y_{t-1})$ helps decoder to copy over subsequence from source sentence and is named as "selective read."

The COPYNET network is fully differentiable and can be trained end to end, exactly like seq2seq model. It minimizes the negative log-likelihood as an objective loss function, as shown below.

$$J = -\frac{1}{N} \sum_{k=1}^{N} \sum_{t=1}^{T} \log[p(y_t|y_1, y_2, ...y_{t-1}, X)] \tag{3.15}$$

3.3.2 Pointer Softmax Network

The Pointer Softmax Network (PS) was proposed in [51]. The idea is to use attention mechanism and attention weights to select a word or token from the input sequence as the output instead of using it to blend hidden units of an encoder to a context vector at each decoder step. This setup was able to copy a word from the input sentence to the output sentence, which is not present in seq2seq model vocabulary or is not seen by the model in the training process. This approach shows the improvement in two tasks, i.e., neural machine translation on the Europarl English to French parallel corpora and text summarization on the Gigaword dataset.

Fig 3.5 shows the architecture of PS. The model learns two main things 1) To predict whether the pointing mechanism is required at each time step 't' 2) To point to the correct location of the word in source sentence which needs to be copied over to target sentence. The model uses two different softmax output layers, first, is shortlist softmax layer, and the second one is the *location softmax layer*. The first one is the softmax layer, which is used in the attention mechanism over all the vocabulary to

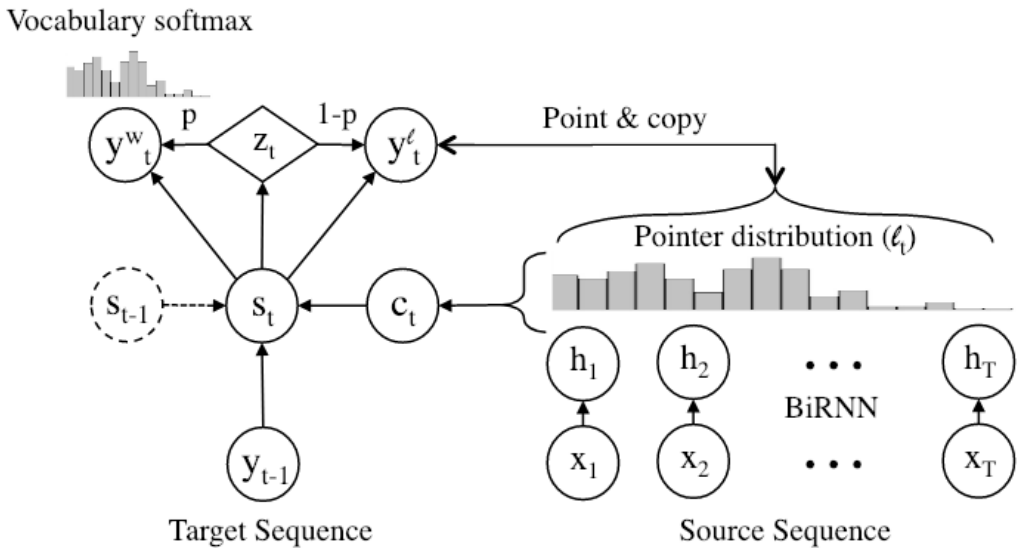

FIGURE 3.5: Illustrations of the PS architecture. At each timestep, l_t,c_t and w_t for the words over the limited vocabulary (shortlist)is generated. z_t is a switching variable that will decides whether to use vocabulary word or to copy a word from the source sequence.

generate a word from the model vocabulary. The second softmax is a location softmax, is an output layer which gives the location of the word from source sentence which needs to be copied over. Location softmax is a pointer network in seq2seq model where each of the output dimension corresponds to the location of a word in the context sequence. Consequently, the output dimension of the location softmax varies according to the length of the given source sequence. The decision of whether the pointer word should be used or shortlist word, a switching network is used. The switching network is a multilayer perceptron network which takes the representation of source sequence and previous hidden state from decoder RNN as a input and outputs the value of binary variable z_t which will indicate whether to shortlist softmax layer(when $z_t == 1$) or location softmax layer(when $z_t == 0$). When the word is not in shortlist softmax and not even in location softmax layer, this switching network chooses to shortlist softmax and gives <UNK> as a next token.

$$p_\theta(y, z|x) = \prod_{t=1}^{T'} p_\theta(y_t, z_t|y_{<t}, z_{<t}, x) \tag{3.16}$$

Here, x is a source sequence, z is a binary variable to indicate which softmax layers probabilities we should use, and y is an output sequence. y_t can be a word either

from location softmax layer or shortlist softmax layer. The probability of y_t from being in shortlist softmax layer is given by:-

$$p(w_t, z_t|(y,z)_{<t}) = p(w_t|z_t = 1, (y,z)_{<t}) * p(z_t = 1|(y,z)_{<t}) \tag{3.17}$$

Accordingly, the probability of y_t from being in location softmax layer.

$$p(l_t, z_t|(y,z)_{<t}) = p(l_t|z_t = 0, (y,z)_{<t}) * p(z_t = 0|(y,z)_{<t}) \tag{3.18}$$

In both the above equation 3.17, and 3.18 we have removed the conditioning of x, which is a source sequence while predicting y.

The probability of switching network is given by following equations.

$$p(z_t = 1|(y,z)_{<t}, x) = \sigma(f(x, h_{t-1}; \theta)) \tag{3.19}$$

$$p(z_t = 0|(y,z)_{<t}, x) = 1 - \sigma(f(x, h_{t-1}; \theta)) \tag{3.20}$$

Using 3.17 and 3.18, 3.16 can be rewritten as following:

$$p(y|z,x) = \prod_{t \in T'} p(w_t, z_t|(y,z)_{<t}, x) * \prod_{t \in T'} p(l_t, z_t|(y,z)_{<t}, x) \tag{3.21}$$

At training time, the switching network is trained along with the full network. To train the network and to calculate loss, we use standard cross entropy error. Furthermore, while testing the model equations, 3.17 and 3.18 choose a word from the shortlist and a location softmax layer.

3.4 Experiments

3.4.1 Dataset

In the first part of the experiment, we compared seq2seq with attention model, pointer softmax model (PS) and COPYNET network, described in 3.2 and 3.3 for paraphrase generation. We train our model on the PPDB dataset described in 2.4.

Note that for higher precision, we have used medium size of PPDB dataset which has almost 9,103,492 pair sentences, which means their score for being a paraphrase is high. Before training the model, we preprocessed the PPDB dataset. We only took the sentence pair where the maximum number of words in a sentence is 50. We also removed all the punctuation signs from the source and target sentences. After removing the punctuation signs, the words like "I'm" becomes "I m," "I haven't" becomes "I havent" and we consider them as two different words. After doing the preprocessing, we partition the dataset into three different categories with 80 percent of samples in the training set and 10-10 percentage of samples in testing and validation set. This same dataset was used for both models for training, testing and at validation time.

3.4.2 Models

In this analysis we compared three different models described in 3.2 and 3.3 for paraphrase generation.

The first model described in 3.2 only uses attention to generate paraphrases. The model is trained on the preprocessed PPDB dataset, where preprocessing steps are described in section 3.4.1. We have used word-level embedding in the model to represent the whole sentence. We first created a vocab of most 30,000 frequent words in training samples and represented them with a unique index. This vocab is also augmented with four extra tokens that are <UNK> for representing unknown words or the words which are not covered in the vocabulary of the model. <PAD> this is used to add extra spacing to a sentence to make it equal to the length of source sentence if the target or source sentence is smaller than length 50, <SOS> and <EOS> representing the start of sentence and end of sentence, respectively. Both <SOS> and <EOS> were added before the sentence starts and at the end of the sentence respectively. Using these unique indices, the words in source text sentence is converted into the list of an integer, which is then fed as an input to the seq2seq model described in 3.2.

Furthermore, the weights and bias parameters are learned by the model by back-propagating the error at training time. This model was then tested and validated

using test samples and validation dataset with different hyper-parameters like the number of hidden units, type of RNN cell used in the encoder-decoder model, batch size, a different type of attention. The output from the decoder is an index value of a generated token which is then converted back to a word by matching it from the vocabulary of the model and then combined to form one complete sentence. We have used the beam size of 3, which means we pick the top 3 sentences with the highest probability.

We followed the same preprocessing in the COPYNET and the pointer softmax model described in 3.3 with a different variation of coping mechanism in seq2seq model. These model further had different hyper-parameters, which were described in section 3.3. Therefore while training the PS model with the encoder-decoder network, a separate multi-layer perceptron model was also trained and was used for binary classification. We used standard binary cross-entropy as a loss function for backpropagating the error in the model. We have also tried this model with different hyper-parameters of the model described in the previous section.

All three models were fine-tuned on hyperparameters and then compared against each other for paraphrase generation by finding the loss in dev dataset, BLEU and METEOR scores. To save time in training we have fixed some parameters like we used teacher forcing while training encoder-decoder model, dropout ratio was fixed to 0.2, Vocab size was used for most 30000 frequent words in training time, and batch size is also set to 250. Note that to our best knowledge, these models were compared only on summarizing of paragraphs previously and not on paraphrase generation of sentences.

As the code for all these model were not made publicly available, we have implemented all these models in Pytorch. We trained our model on GPU provided by Helios Calcul Quebec, which has 15 compute nodes, each of which has eight K20 GPU's from Nvidia, and 6 compute nodes and eight nVidia K80 boards each. Each K80 board contains two GPU, for a total of 216 GPUs for the cluster.

Seq2Seq Model with Attention			
Hidden Layer	Number of Layers in RNN	Type of RNN Cell	Valid Perplexity
128	1	GRU	27.1790
128	1	LSTM	27.3762
256	1	GRU	28.1144
512	1	GRU	26.5589
128	**2**	**GRU**	**26.5401**
128	2	LSTM	26.7232

TABLE 3.1: Results of seq2seq with Attention model with different hyper parameters on PPDB test dataset. Smaller perplexity indicates better performance

Seq2Seq Model with Pointer softmax network			
Hidden Layer	Number of Layers in RNN	Type of RNN Cell	Valid Perplexity
128	1	LSTM	29.9218
128	**1**	**GRU**	**26.5936**
256	1	GRU	27.4747
512	1	GRU	26.8019
128	2	GRU	28.2140

TABLE 3.2: Results of seq2seq with Pointer softmax model with different hyper parameters on PPDB test dataset. Smaller perplexity indicates better performance

3.5 Results and Analysis

It took us about one day to train one model with only the attention mechanism and one epoch. This time increases while training the model for pointer softmax model as this model also consist of training of the separate model, i.e., switching network. The complexity was also high to create pointer vocabulary and calculating the probabilities of both source and model vocabulary. Due to limited time we had, to compare models, we first tried out the attention model, PS and COPYNET model with only one epoch and with different hyper-parameters. It was done to find out the best configuration for the model to proceed with based on the first iteration. We trained every model for 15 epochs, and after looking at the training and validation perplexity, we can conclude that the model converges after 15 iterations.

Tables 3.1, 3.2 and 3.3 exhibit the validation perplexity on the PPDB dataset. Furthermore, figures 3.8, 3.7 and 3.9 reveal the curve of train and valid perplexity. Looking at the results, COPYNET outperforms the other two models by a small margin.

Seq2Seq Model with COPYNET network			
Hidden Layer	Layers in RNN	Type of RNN Cell	Valid Perplexity
128	1	LSTM	26.6721
128	1	GRU	26.7842
256	1	GRU	26.9701
512	1	GRU	26.6891
128	**2**	**GRU**	**25.9625**
128	2	GRU	26.3713

TABLE 3.3: Results of seq2seq with COPYNET Pointer network with different hyper parameters on PPDB test dataset. Smaller perplexity indicates better performance

Model Comparison		
Model	BLEU-Score	METEOR-Score
$Seq2Seq_{attn}$	0.4538	0.3035
$Seq2Seq_{attn+COPYNET}$	**0.4547**	**0.3464**
$Seq2Seq_{attn+PS}$	0.2922	0.3219

TABLE 3.4: BLEU and METEOR score on test dataset of PPDB dataset with attention, COPYNET and Pointer softmax. The higher the scores indicate better performance.

Switching layer in the pointer softmax model did not help much in the generation of paraphrases. In the table, 3.4 we show the performance of all different models on test dataset consisting of 25,000 examples. The BLEU and METEOR score was slightly better for COPYNET network as compared to other models.

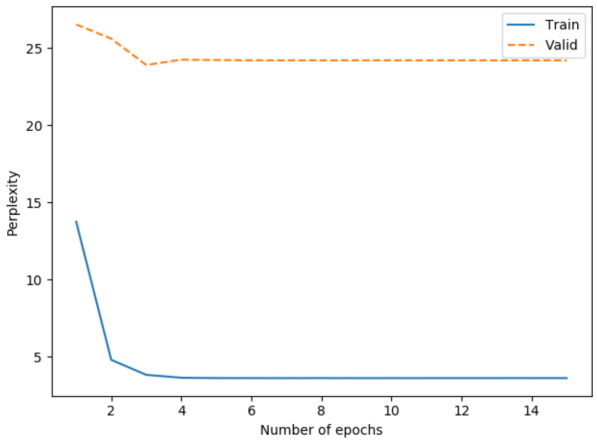

FIGURE 3.6: Plot showing perplexity of seq2seq model with attention model on PPDB training and test data set.

Seq2seq model with attention mechanism and with COPYNET network model both shows the best performance at iteration 3, and they have minimum validation perplexity at this point, i.e., 23.9142 and 23.6172 respectively. On the other hand, the

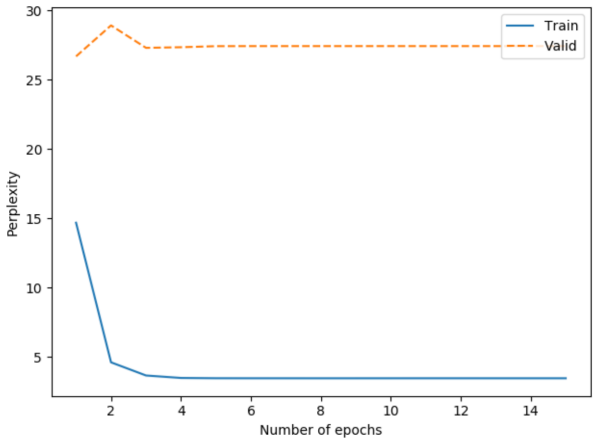

FIGURE 3.7: Plot showing perplexity of seq2seq model with pointer softmax (PS) network model on PPDB training and validation data set.

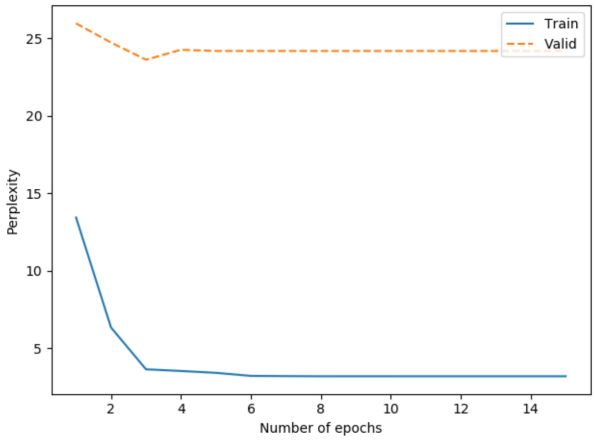

FIGURE 3.8: Plot showing perplexity of seq2seq model with COPYNET pointer network model on PPDB training and validation data set.

pointer softmax model gave the best result at one iteration, where we got minimum validation perplexity as 26.6837. The final hyper-parameter we chose for attention and COPYNET network were **dropout ratio=0.2, teacher forcing ratio=1.0, vocab size=30,000, batch size=250, number of hidden layers=128, number of layers in RNN = 2, RNN cell type=GRU, attention type = Luong attention.** Also, the hyper-parameter of pointer softmax network model was **dropout ratio=0.2, teacher forcing ratio=1.0, vocab size=30,000, batch size=250, number of hidden layers=128, number of layers in RNN = 1, RNN cell type=GRU, attention type = Luong attention**.

We show some of the examples of paraphrases generated by different models below. Note, that these source sentences were picked randomly and were not in the PPDB

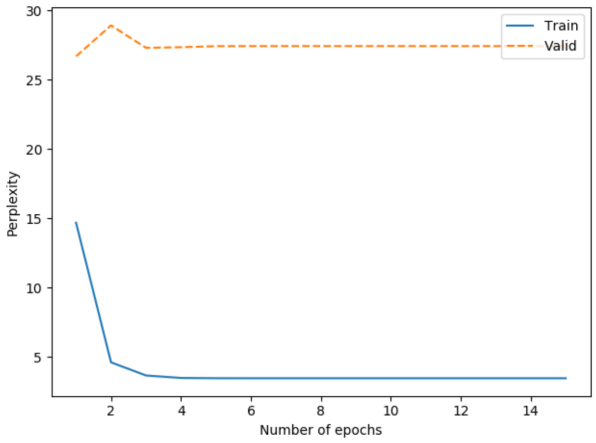

FIGURE 3.9: Plot showing perplexity of seq2seq model with pointer softmax network model on PPDB training and validation data set.

test dataset Below, 'sentence' represents the original sentence which was given as an input to the model. 'Paraphrase(Attn)' represents the paraphrase generated by seq2seq model with attention model. 'Paraphrase(COPYNET)' presents the paraphrase generated by COPYNET network and 'Paraphrase(PS)' shows the paraphrase generated by pointer softmax network.

1. **Sentence**: Niagara Falls is viewed by thousands of tourists every year.

   Paraphrase(Attn): Ontario is recognized by 1000 <UNK> <EOS>

   Paraphrase(COPYNET): Niagara Falls is recognized by 1000 tourists <EOS>

   Paraphrase(PS): Ontario is recognized by 1000 visitors <EOS>

2. **Sentence**: Economy is a big problem for the Bush administration

   Paraphrase(Attn): <UNK> government problem is <UNK> <EOS>

   Paraphrase(COPYNET): Bush government problem is economy <EOS>

   Paraphrase(PS): George's government problem is big <EOS>

3. **Sentence**: Language is complex and the process of reading and understanding language is difficult for many groups of people

   Paraphrase(Attn): Language is difficult for <UNK> and <UNK> <eos>

   Paraphrase(COPYNET): Language is difficult for reading and understanding <eos>

   Paraphrase(PS): Speech is complex for understanding for many people<eos>

4. **Sentence**: I don't know.

Paraphrase(Attn): I m not aware of <EOS>

Paraphrase(COPYNET): I m not aware of <EOS>

Paraphrase(PS): I m not aware of <EOS>

Glancing at the examples above, we can conclude that seq2seq model with COPYNET generated better paraphrases given the original sentence, which goes aligned with our results shown earlier.

3.6 Summary

In this chapter, initially, we show how encoder-decoder networks become popular in natural language processing generation tasks. We show that what were the significant disadvantages of these model and how these issues were solved using attention and different variation of pointer network. Then we talk about COPYNET and pointer softmax (PS) model.

We performed experiments on seq2seq model with attention and two different variations of pointer network under the supervision of PPDB dataset and compared their results using metrics like BLEU and METEOR score. In this experiment, COPYNET outperforms pointer softmax pointer network by a little margin. Then we show some examples of paraphrases generated by these models. From these examples and results, it can be concluded that COPYNET pointer network yields best paraphrases among compared models.

Chapter 4

Convolutional neural network for text classification

4.1 Convolution neural network for NLP

Recurrent neural networks are efficacious at predicting sequential data, as described in section 2.5. However, it also involves a lot of complex calculations for predicting the sequence. That is why it might be an overkill to use RNN in text classification, which can also be solved by considering only the symbolic words in the text. Convolutional neural networks (CNN) are viral, particularly in computer vision problems like object recognition, classification of objects. Since these models also became familiar and widely used in natural language processing for text classification, Implicit discourse relation recognition [24], [30], and other tasks. CNN has been successful in numerous text classification tasks. As discussed in the [24], the author shows that even with a little tuning of hyperparameters the CNN becomes state of the art in 4 out of 7 tasks by showing outstanding results on multiple benchmarks. As an illustration of the text classification task, the architecture of the convolutional neural network which explains how CNN can be applied in text classification is shown in fig 4.1.

In fig 4.1, we show an example of a convolutional network on a 7-word sentence and with word embedding of 5 dimensions. CNN consists of 3 main components which are listed below other than feature map (Input vector of a given sentence, generally it is in a matrix form of word length in sentence X dimension of word embedding).

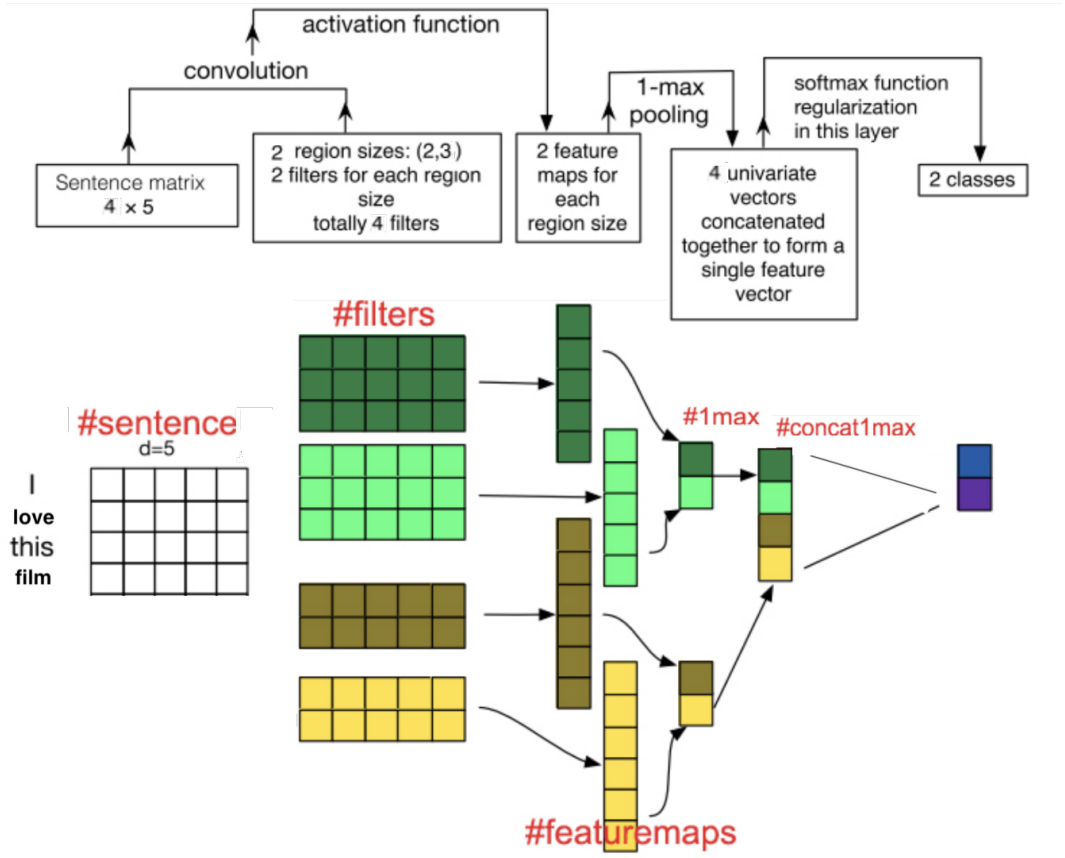

FIGURE 4.1: An architecture of CNN model for text classification task.

1. **Filters** The first step after getting the feature map as an input to the model is applying filters on the feature map. Filters are used to capture some fixed kind of patterns which might be helpful for the classification task. Usually, in the field of computer vision, they have 2D spatial orientation like in image processing. However, in the text, we have the only dimension, which is the single order of words in a sentence. We use filters to capture essential words like 'good,' 'bad,' which might be helpful to classify the text. Filters can be vector or matrix depending on the region size h, which is the product of how many words or n-grams we want to see at time step t and the width of filters is the same as of dimension of word embeddings. Note that each filter is scrolled through the whole sentence until it reaches the end, and a new output vector is calculated by applying bias term and activation function. Fig 4.2, pictures this method.

2. **Pooling Layer** Second step is to take out only the important features from the vectors or matrices we received after applying filters on the input feature map.

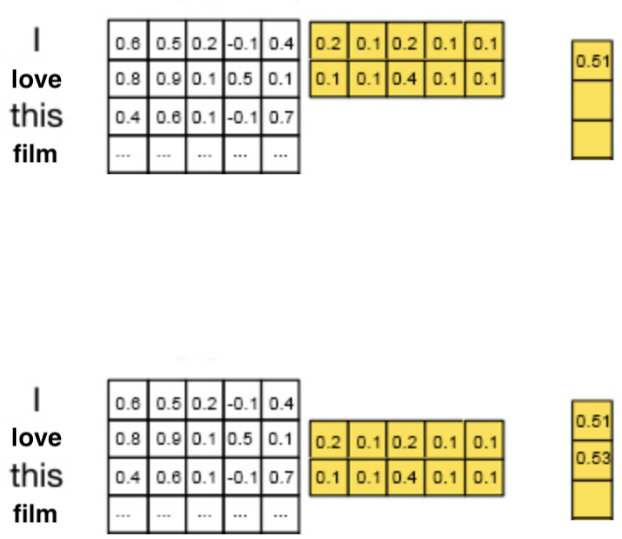

FIGURE 4.2: Applying filter on feature map in CNN. Feature map is a matrix created by converting words into word embedding, the length of this feature map is equal to the number of total words in a sentence and width of feature map is the length of word embedding of a single word in a sentence. In pooling layer, we apply filter matrix of length h (h=2 in image above) and width equal to the size of word embedding to extract important features from the feature map. This step is required to reduce computation later.

For this, we apply 1-max pooling layer. As the filter vector have words equal to the *h* length, and to decrease the calculations we only might want to extract the important word for instance if filter captures 'like the' then we only want to extract 'like' by applying the max-pooling layer.

3. **ConcatMax Layer** Third step is to a concatenate all the words embedding of words which are extracted out by applying max pooling layer.

The output from the concatmax layer is then a fixed length vector which is passed either through a softmax layer or other model like neural network, support vector machine for classification of text. The error from the classification is than back-propagated through the model and in length of filter and bias term used while applying the filter to feature map.

In this work, we also use a convolutional network for classification of text generated by different paraphrase generation models, discussed in 3 under the supervision of

PPDB dataset.

4.2 Experiments

4.2.1 Dataset

For training the binary classifier for text classification, we used IMDb movie dataset described in section 2.4. The only preprocessing practiced while training the model is to remove punctuations and to make every word lowercase. Furthermore, we divide the training dataset into train data and validation dataset, and we kept the test dataset untouched. These three datasets are then used to train, validate and test the models, respectively.

4.2.2 Model

For the selection of a model for this experiment, we tried different varieties of models for text classification to select the best model for this task. We also implemented [23] and [24], in this section. Some of the steps were universal to all the models like creating a vocabulary list of most frequent 25000 words, and then map them with a unique index, preprocessing of sentences. The source sentence is then converted to these index values and given as an input to a model. Also preprocessing were the same in all models.

Recurrent Neural Network (RNN)

We first examined a single layer recurrent neural network (RNN) for the classification task. The model take a sequence of words $S = s_1, s_2, s_3, , , s_n$ as an input one at a time t along with the previous hidden state h_{t-1}, and produces a hidden state h_t for every word in a sequence. This process is repeated until all the words are passed into the model, and the model produces a final hidden state, h_T.

$$h_t = f(s_t, h_{t-1}) \tag{4.1}$$

The final hidden state h_T is passed to a linear layer l, also known as a fully connected layer, to predict the final classification of the text.

$$\bar{y} = l(h_T) \tag{4.2}$$

Please note that in this model, we use one-hot vector encoding to represent a word in a model. Furthermore, we used binary cross-entropy as a loss function with logits to bound output between 0 and 1. Fig 4.3, shows the architecture of this model.

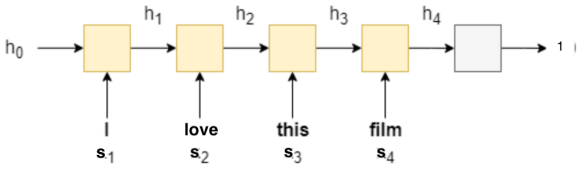

FIGURE 4.3: Single layer RNN model architecture for classification task. Here, h_i is the hidden state from previous encoding $t-1$ step and s_i is the input at time t.

Bi-directional Long Short Term Memory (LSTM)

Next, we improvised the vanilla RNN model used in the previous setup by using bidirectional LSTM model, as LSTM has shown better results when compared with vanilla RNN version. In this model, instead of using one hot vector encoding for word embeddings, we used pre-trained word embeddings obtained from Global Vectors for Word Representation (GLOVE) [42]. In the previous model, the hidden state h_t obtained from previous RNN unit is passed to the next RNN unit, only in the forward direction. In this model, we used bi-directional LSTM where the hidden state is passed from the first word to the last word h_t, and from the last word to the first word $\bar{h_t}$. Furthermore, we also added regularization in the model, called dropout. In dropout, we randomly set any neuron to zero in forwarding pass.

The sentiment of the text is predicted by passing the concatenation of final hidden state from both the forward hidden layer h_t, and backward hidden layer $\bar{h_t}$. The concatenation of both the hidden states is further passed through a linear layer as before.

$$\bar{y} = l(h_t, \bar{h}_t) \tag{4.3}$$

We used the same binary cross-entropy as a loss function with logits as before. Fig 4.4 shows the architecture of the bidirectional model.

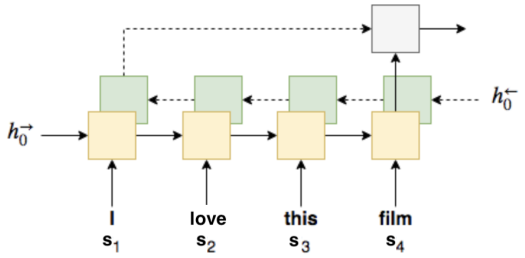

FIGURE 4.4: Bi-directional LSTM model architecture for classification task

FastText Model

We also tried fasttext model proposed in work [23]. Fasttext has shown promising results on different classification tasks, as shown in work. This model has very fewer hyperparameters as compared with previous models. We have also used pre-trained word embedding GLOVE. In this model, the text is first converted using word embeddings and then passed as an input to the model. Then the average of all these word embeddings (using average pooling filter) is calculated and passed through a linear layer for the prediction of sentence classification. Furthermore, we use the same loss function as before, i.e., binary cross entropy with logits. Fig 4.5 shows the architecture of the fasttext model.

Then we examine the convolutional neural network model for classification task as described in section 4.1. We used the same pre-trained word embeddings obtained from GLOVE and the same loss function to train and backpropagate error through the model.

Finally, all the models were first fine-tuned on validation dataset, and then their performance was evaluated on test dataset while choosing the best model for further classification experiment.

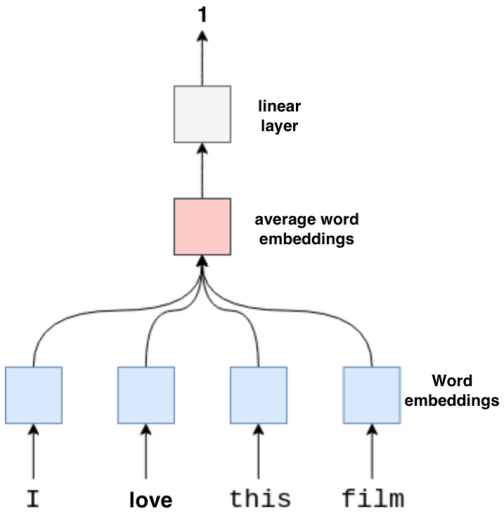

FIGURE 4.5: Fasttext model architecture for classification task

4.3 Result and Analysis

This experiment is significant to this work, and we needed to get the stable and high-grade model for the text classification because if we cannot manage to get reasonably higher accuracy on IMDb movie dataset, then it becomes useless to us to estimate the quality of generated paraphrases from seq2seq model. That is why, in this segment, we examined different methods to get reasonably high accuracy.

We made all the models in Pytorch, and as the models were less complex as compared to paraphrase generation model, we trained our models on 10 CPU on a google compute engine instance. We train every model for ten epochs.

Firstly, we train a single layer RNN model 4.2.2 using stochastic gradient descent (SGD) as an optimizer, and we were able to get the accuracy of **57.98%** with the hyperparameters **Batch size = 64, Most frequent words vocabulary = 25000, word embedding dimensionality = 100, hidden dimensionality = 100, learning rate = 1e-3, input dimensionality = 25000, output dimensionality = 1**. We were able to reduce the training loss up to **0.6930**. The confusion matrix of this model is shown in 4.1. The F1-Score, precision, sensitivity, and specificity were evaluated as **0.0554, 0.4881, 0.6421, 0.3267** respectively. The confusion matrix of this model is shown below.

Next, we train bi-directional LSTM model 4.2.2. We again use adam as an optimizer, with pre-trained GLOVE word embeddings, and with hyperparameters **learning**

TABLE 4.1: Confusion matrix for single layer RNN model.

		Actual value		
		Positive	Negative	Total
Predicted outcome	Positive	8027 (64.20%)	8416 (67.32%)	16443
	Negative	4473 (35.80%)	4084 (32.68%)	8557
	Total	12500	12500	*N*

TABLE 4.2: Confusion matrix for bi-directional LSTM model.

		Actual value		
		Positive	Negative	Total
Predicted outcome	Positive	10378 (83.02%)	4384 (35.092%)	14762
	Negative	2122 (16.97%)	8116 (64.92%)	10238
	Total	12500	12500	*N*

rate = 1e-3, batch size = 64, most frequent words vocabulary = 25000, input dimensionality = 25000, output dimensionality = 1, word embedding dimensionality = 100, hidden dimensionality = 100, bidirectional RNN = true, number of layers in RNN =2 and dropout = 0.5 . This time we were able to achieve an accuracy of **82.74%** with F1-score **0.7610**, precision as **0.7030**, sensitivity as **0.8030** and specificity as **0.6490**. Glove pre-trained word embeddings have shown much improvments in this model due to which we use same word embeddings in next models. The confusion matrix of this model is shown in 4.2.

Then we train fasttext model 4.2.2 which was the fastest model to train as it only average word embeddings without any complicated calculations as compared to other models. The accuracy on IMDb dataset improved by 5% and we were managed to get the **86.27%** accuracy on the test dataset. The training loss also reduces to 0.386. The parameter used for this model were **batch size = 64, input dimensionality = 25000, word embedding dimensionality = 100, output dimensionality = 1**. We also used pre-trained GLOVE word embeddings to convert the input text to dense vector. The values of other performance metrics like F1-score, precision, sensitivity and specificity were **0.6432**, **0.5954**, **0.6992** and **0.5249**, respectively. Table 4.3 shows the confusion matrix of this model.

Finally, we train the CNN model using the Adam optimizer and GLOVE pre-trained word embeddings. The hyperparameter of this model were **imput dimensionality = 25000, output dimensionality = 1, dropout = 0.5, number of filters = 100, filter**

TABLE 4.3: Confusion matrix for FastText model.

		Actual value		
		Positive	Negative	Total
Predicted outcome	Positive	8741 (69.92%)	5938 (47.50%)	14679
	Negative	3759 (30.07%)	6562 (52.49%)	10321
	Total	12500	12500	*N*

TABLE 4.4: Confusion matrix for CNN model.

		Actual value		
		Positive	Negative	Total
Predicted outcome	Positive	10081 (80.64%)	2249 (17.99%)	12330
	Negative	2419 (19.35%)	10251 (82.00%)	12670
	Total	12500	12500	*N*

sizes = [3,4,5], word embeddings =100. This model improved the test accuracy, and we managed to get **88.6600%**. Furthermore, we see a decrement in the training error from **0.6930** to **0.3070**. The F1-score, precision, sensitivity, and specificity were **0.8120**, **0.8175**, **0.8064** and **0.8200**, respectively. This model shows improvement in all performance metrics as compared to other evaluated models. The confusion matrix of this model is shown in table 4.4.

Table 4.5, 4.6 summarizes the performance of examined models in terms of different performance metrics. Then we plot validation and training loss curve for the CNN model in graph 4.7 w.r.t number of epochs. The receiver operating characteristic curve (ROC) of examined models is shown in fig 4.7.

This experiment also proves that ConvNets works better on text classification than RNN, which goes with the results found in [24].

Sentiment classification with different models		
Model	Test Loss	Test Accuracy
RNN	0.6930	47.8400
Bidirectional Multilayer RNN	0.3740	85.3500
FastText	0.3870	85.1600
CNN	**0.3070**	**88.6600**

TABLE 4.5: Results of different models on sentence classification in NLP.

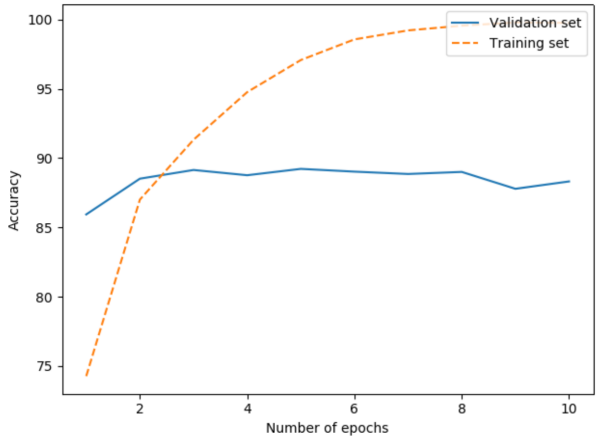

FIGURE 4.6: Training and validation accuracy wrt to number of epochs while training text classifier

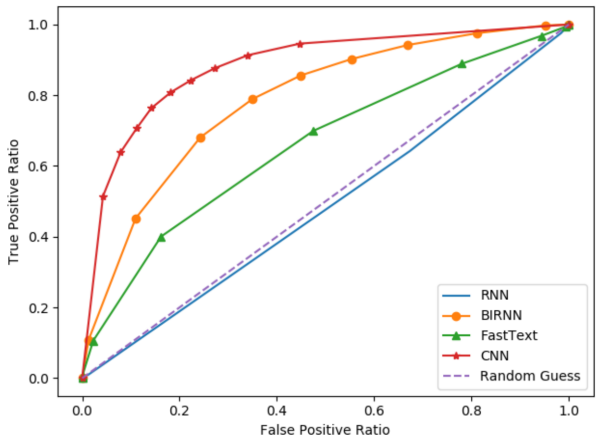

FIGURE 4.7: Receiver operating characteristic curve(ROC) of sentence classifier

4.4 Summary

In this chapter we initially show how CNN started to become popular in natural language processing (NLP) field after showing promising results in computer vision field. CNN shows promising and significant improved results of classification tasks when compared to RNN and become state-of-the art for classification tasks. CNN is also faster to train due to less complicated calculations when compared with RNN.

We compared different models for text classification to identify the one with reasonably high accuracy on IMDb sentiment classification. It was a crucial experiment and was also significant to our next set of an experiment which is to evaluate the performance of paraphrase generations models. The results of these experiments align with the results mentioned in paper [24]. Then we show the results of our

Sentiment classification with different models				
Model	F1-Score	Precision	Sensitivity	Specificity
RNN	0.5546	0.4881	0.6421	0.3267
Bidirectional Multilayer RNN	0.7610	0.7030	0.8030	0.6490
FastText	0.6432	0.5954	0.6992	0.5249
CNN	**0.8120**	**0.8175**	**0.8064**	**0.8200**

TABLE 4.6: F1-score, Precision, Sensitivity and Specificity of different models on sentence classification in NLP.

experiments on IMDb dataset by doing text classification on it.

Chapter 5

Evaluation of generated paraphrases in downstream task

In this chapter, we investigate the use of paraphrases generated by the different models under the supervision of PPDB dataset, described in chapter 3. The primary motive of this analysis is to see how much these generated paraphrases can help in downstream tasks like improving classifier to perform better on the original test dataset by augmenting the training dataset. Furthermore, we perceive which configuration of pointer and attention network in seq2seq model can produce better similar sentences which can help in the classification task performed on IMDb dataset.

5.1 Experiments

5.1.1 Dataset and Model

As suggested earlier in this chapter, to asses the use of generated paraphrases by the different seq2seq model under the supervision of PPDB dataset in a downstream task, we augment the original training dataset of IMDb movie dataset by generating one paraphrase of each sample in the dataset. Then we use the generated paraphrases along with original training dataset of IMDb movie review dataset to re-train the best classifier model we found in 4, i.e., convolutional neural network (CNN). Initially, we had 25,000 samples of a movie review in training dataset, but

after generating paraphrase of each movie review, we were able to augment the training dataset from 25,000 movie reviews to 50,000 movie reviews. Note that, we kept the original test dataset untouched for making these models performance comparable. So, test dataset had 25,000 movie reviews and was not used to generate paraphrases. Also, it is worth pointing out that we generated these paraphrases by trained models which we examined in chapter 3 result's section. We kept all the preprocessing steps same as before. We repeated the same process with all three different models examined in chapter 3 to get three preprocessed training dataset and only 1 test dataset.

While generating the paraphrases we used beam (n=3) to find the top 3 sentences and picked the one sentence which has a better language model score.

To avoid the correlation between the samples already present in the training dataset and generated paraphrases of IMDb movie reviews, we use a sample reweighting technique where we sample each of the correlated samples to make them equal to one.

5.2 Result and Analysis

First, we train the CNN model with paraphrases generated by seq2seq model with an only attention mechanism. We train the model for 30 iterations with hyperparameters **batch size=100, word embedding dimensionality = 100, length of vocabulary = 25000, number of filters = 100, filter size = [3,4,5], dropout =0.5** and Adam as an optimizer. We used pre-train glove word embeddings to convert input sentence to dense vector. With this configuration, we were able to get an accuracy of **77.8079%** with F1-score = **0.7856**, sensitivity = **0.8132**, specificity = **0.74296**, precision = **0.7598**. We received best model at 5 iteration as shown in Fig 5.1. The confusion matrix of this model is shown in table 5.1.

Then we train the CNN model with paraphrases generated by seq2seq model with COPYNET pointer network illustrated in 3.3.1. We tried it with the same hyperparameters which we used in the original CNN classifier in chapter 4, i.e., **batch**

TABLE 5.1: Confusion matrix of CNN Model trained on paraphrases generated by seq2seq model using only attention mechanism.

		Actual value		
		Positive	Negative	Total
Prediction outcome	Positive	10165 (81.32%)	3213 (25.70%)	13378
	Negative	2335 (18.68%)	9287 (74.29%)	11622
	Total	12500	12500	*N*

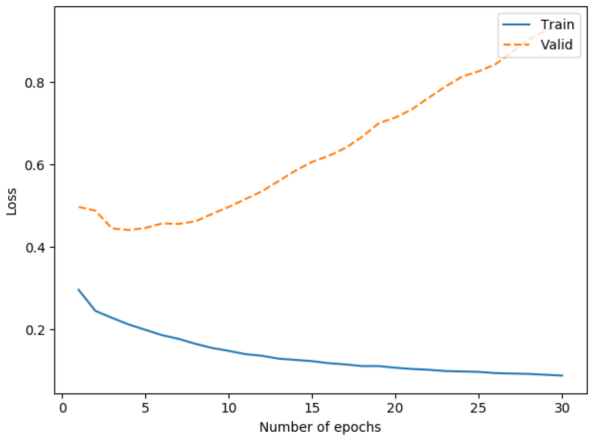

FIGURE 5.1: Training and validation loss error curve for seq2seq model with attention mechanism only and trained under the supervision of PPDB.

size=100, word embedding dimensionality = 100, length of vocabulary = 25000, number of filters = 100, filter size = [3,4,5], dropout =0.5 with Adam algorithms as an optimizer and binary cross-entropy as a loss function. Using this hyper-parameters, we got an accuracy of **78.3%**. The other performance metrics were f1-score = **0.79064562** , sensitivity= **0.81952**, specificity=**0.74648** and precision=**0.763736**. The training and validation error curve is shown in fig 5.2, where it can be seen that we get minimum validation error at iteration=3. The confusion matrix of this model is shown in table 5.2.

Next, we trained the CNN model with the paraphrases generated by the pre-trained seq2seq model with pointer softmax pointer network 3.3.2, and then we evaluated its

TABLE 5.2: Confusion matrix of CNN Model trained on paraphrases generated by seq2seq model with COPYNET pointer network.

		Actual value		
		Positive	Negative	Total
Prediction outcome	Positive	10244 (81.95%)	3169 (25.35%)	13413
	Negative	2256 (18.04%)	9331 (74.64%)	11587
	Total	12500	12500	*N*

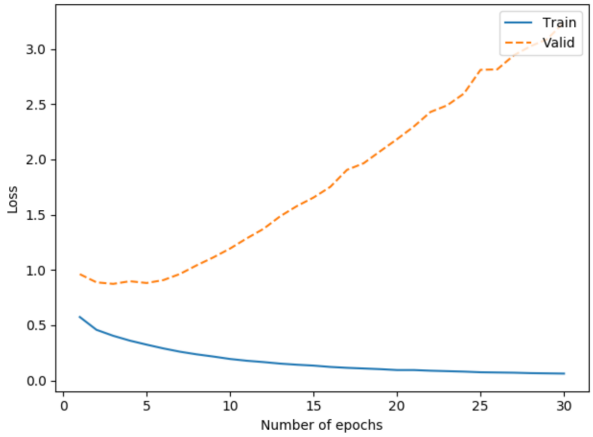

FIGURE 5.2: Training and validation loss error curve for seq2seq model with COPYNET as a pointer network trained under the supervision of PPDB.

TABLE 5.3: Confusion matrix of CNN Model trained on paraphrases generated by seq2seq model with Pointer Softmax pointer network.

		Actual value		
		Positive	Negative	Total
Prediction outcome	Positive	9888 (79.10%)	2884 (23.07%)	12772
	Negative	2612 (20.89%)	9616 (76.92%)	12228
	Total	12500	12500	*N*

performance on the original test dataset of IMDb dataset. We were able to get the test accuracy of **78.0159%** with f1-score, precision, sensitivity, and specificity as **0.7825**, **0.7741**, **0.7741**, **0.79104**, **0.76298** respectively. Fig 5.3 shows the training and validation loss error curve of this model. We trained this model for 30 epochs, and we got the best model at the fourth iteration. The parameters of this model were **batch size=100, word embedding dimensionality = 100, length of vocabulary = 25000, number of filters = 100, filter size = [3,4,5], dropout =0.5**. We use the same Adam algorithm as an optimizer and binary cross-entropy as a loss function. The accuracy of the model decreases from **88.66%** to **78.0159%** after adding the paraphrases, which shows the paraphrases generated by this model did not help in improving the test accuracy on IMDb movie review dataset. Table 5.3 shows the confusion matrix of this model.

Fig 5.4 abstracts all the performance metrics of each model and compare this with the CNN model where we did not add paraphrases while training the model.

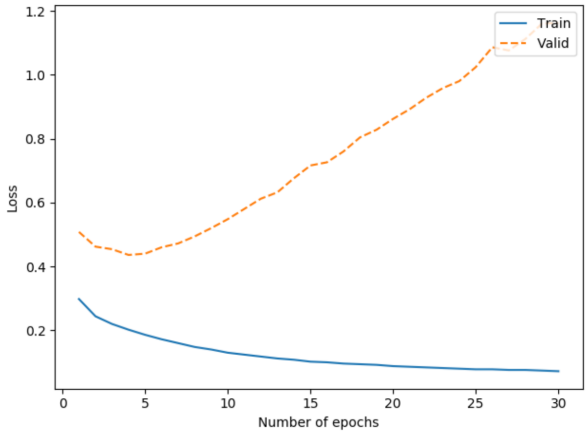

FIGURE 5.3: Training and validation loss error curve for seq2seq model with pointer-softmax as a pointer network trained under the supervision of PPDB.

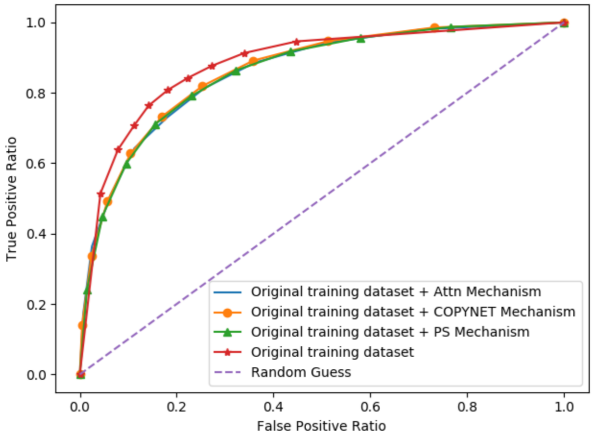

FIGURE 5.4: Receiver operating characteristic curve (ROC) for model having different training dataset. Here, original training dataset is the original training dataset which comes in IMDb movie review and consist of 25,000 samples, and consider as a baseline model for our experiment.

Fig 5.4 shows the receiver operating characteristic curve (ROC) of the analyzed models. Looking at the curve, we can infer that paraphrases generated using three distinct configurations in the Encoder- Decoder network did not help in improving the test accuracy on IMDb movie review test dataset. Moreover, all the performance metrics value decreases, which are summarized in table 5.4. These models have shown significant improvements in other NLP tasks but did not help in generating paraphrases under the supervision of PPDB dataset. The poor performance of these models can be either because of training them on PPDB dataset or model itself. Most

Text Classification					
Train Dataset	Test Accuracy	F1-Score	Precision	Sensitivity	Specificity
IMDb Original Train dataset	**88.6600**	**0.8120**	**0.8170**	0.8060	**0.8200**
IMDb Original Train dataset + paraphrases generated by seq2seq model with attention mechanism	77.8079	0.7856	0.7598	0.8132	0.7429
IMDb Original Train dataset + paraphrases generated by seq2seq model with COPYNET pointer network	78.3000	0.7906	0.7637	**0.8195**	0.7464
IMDb Original Train dataset + paraphrases generated by seq2seq model with pointer softmax pointer network	78.0100	0.7825	0.7741	0.7910	0.7629

TABLE 5.4: Evaluation of performance of paraphrases generated by Encoder-Decoder network in downstream task.

of the paraphrases in the test dataset were not at all complete sentence and not useful at all. Below we show one example of paraphrase generated by these models for the training dataset of IMDb movie review dataset.

1. **Sentence** Hello . This movie is ..... well ..... okay . Just kidding ! ITS AWESOME ! It's NOT a Block Buster smash hit . It's not meant to be . But its a big hit in my world . And my sisters . We are rockin' Rollers . GO RAMONES ! ! ! ! This is a great movie ..... For ME !

2. **Attention mechanism** hey guys . this s all right . just kidding . it's great . it is not a little <unk> . it's not only possible . but it s a big deal . and my <UNK> . we're <unk> . get going . that s a good idea .

3. **COPYNET** hey guys . this s all right . just kidding . it's great . it is not a little block buster . it's not only meant . but it s a big hit . and my dear . we re rockin . get going . that s a good movie .

4. **Pointer Softmax** hello this film s okay . kidding . awesome . it s not a stop . but a great deal s big . and my . we re rockin Rollers . ramones . this is a great deal for me .

After examining the full training dataset's paraphrases, generated by these models, we concluded that the main reasons for the unsatisfactory performance of these models are as follows:

1. The PPDB dataset does not have the complete sentences and only consists of lexical (single word to single word), phrasal (multiword to single/multiword), and syntactic (paraphrase rules containing non-terminal symbols) paraphrases, which in practice does not help Encoder-Decoder network to perform better on bigger or complete sentences. Furthermore, the IMDb movie review dataset includes reviews with more than 250 words where the performance of Encoder-Decoder network reduces to a great extent.

2. The PPDB dataset is an automatically extracted database, due to which it has many words which even does not make sense like 'rrb,' 'br,' '<html>' and others. In the dataset, we also notice most of the time, the word 'film' is getting converted into 'idea' or 'way,' which is not useful at all.

3. In Encoder-Decoder with pointer softmax pointer network, we observed that the same word is getting repeated many times in the same sentence. Furthermore, it generates many <UNK> tokens when the switching network chooses a word from attention mechanism, and the highest probability is given to a <UNK> token.

4. Encoder-Decoder with COPYNET pointer network performed a little bit better than other models, but the sentences generated by the model were not grammatically correct, which can be because of supervision of PPDB dataset.

5.3 Summary

In this chapter, we conducted experiments related to the evaluation of paraphrases generated by the different pointer network in encoder-decoder models described in chapter 3 in the downstream task, i.e., in text classification. We did this by augmenting the training dataset of IMDb movie review dataset and evaluating its performance on the original test dataset.

The results were not stimulating enough, and we observed a 10-11% decrease in test accuracy of IMDb movie review dataset. We further analyze these generated paraphrases and mention our analysis in result and analysis section of chapter 5.

Chapter 6

Conclusions and Future Work

6.1 Summary

In this work, we examined the performance of the encoder-decoder (also recognized as seq2seq model) model with attention and pointer network on generating paraphrases under the supervision of the medium size PPDB dataset. Furthermore, we evaluated the quality of generated paraphrases by using generated paraphrases for IMDb movie dataset in the classification task. In chapter 3, we first discussed the different variety of pointer network which performed significantly better in text summarization and text simplification tasks. We trained these model under the supervision of the PPDB dataset and analyze its performance of test dataset from the PPDB dataset. We summarize all results in section 3.5, here we use BLEU and METEOR as performance metrics to evaluate these models performance. Then in chapter 4, we investigated and trained different models for text classification on IMDb movie review dataset to create a stable and robust baseline on text classification. Here we also present, how convolution neural network outperforms other models for the text classification on IMDb movie review dataset.

As BLEU and METEOR have their limitations for evaluating the similarity between two sentences, we further generated paraphrases of the training dataset of IMDb movie dataset in chapter 5. We use these generated paraphrases to augment the training dataset of IMDb movie review dataset and retrain convolutional neural network with augmented training dataset to analyze the quality and how useful are these paraphrases in downstream tasks. We present our results in section 5.2. The

results were not promising enough, and we saw a decrement of 10-11% in test accuracy on IMDb movie test dataset after adding these paraphrases, which states that these paraphrases were not significant at all.

After analyzing the dataset of paraphrases generated by these models, we inferred that the unsatisfactory performance of these models could be due to the following reasons:

1. Wrong generalization of relevant terms in IMDb movie review dataset. For instance, pre-trained seq2seq on the PPDB database learns to map from 'best' to 'highest,' which caused to generate paraphrase for the sentence 'It was the best movie I have ever seen' to 'Highest movie so far,' which further classified by the CNN as a negative review. It can be because of the generation of paraphrases for the different domain.

2. The switching network in pointer softmax did not help much and most of the time, end up by repeating the same words again and again. The model also produces many <UNK> token when the probability of generating word using attention vocabulary was higher than pointer vocabulary. For example for movie review "This is a great horror film for people who do not want all that vomit-retching gore and sensationalism' the generated paraphrase was 'That is a great <unk> movie <eos>.' The second example which shows the repetitive generation of the same word is 'it's a lot of course <eos> this film on course of course <eos> it's time of course <eos> <unk> of <unk> <unk> <eos> so the house s right there of course <eos> i have seen a lot <unk> <eos> this film while much to <eos>'.

3. The problem also comes from the supervision of the PPDB dataset as it only consists of lexical (single word to single word), phrasal (multiword to single/multiword), and syntactic (paraphrase rules containing non-terminal symbols) paraphrases, due to which most of the time the generated sentences are not grammatically correct. Furthermore, the IMDb movie review dataset consists of very long sentences, which further decreases the quality of these paraphrases significantly.

4. As mentioned before, COPYNET pointer network performed a little bit better compared to other models, but because of the supervision of the PPDB dataset, the generated sentences were not well structured and complete.

6.2 Contributions

This thesis presented the following contributions:

1. Implementing and evaluation of seq2seq model with attention and different variants of pointer network under the supervision of PPDB dataset.
2. Implementing different classification models in NLP like single layer RNN, bidirectional RNN, Fasttext and CNN, to create a stable baseline on IMDb movie review dataset for classification tasks.
3. Evaluation of generated paraphrases in the downstream task by augmenting IMDb movie training dataset and comparing with solid baseline. We also show how the performance of downstream tasks is highly correlated with higher METEOR score or quality of generated paraphrases.
4. Evaluation of the PPDB dataset for generating paraphrases by looking at the examples generated by the encoder-decoder model.

6.3 Future Work

In chapter 3, we examined the different configurations of the encoder-decoder model with attention mechanism and the different pointer network. The results are shown in section 3.5 are not satisfactory enough, but due to lack of time and no availability of code online, we were not able to try out all the different variations of implementation of pointer network in seq2seq model. The max number of epocs we examined for each model was 15 in which we were able to see the convergence of the model, but it can be carried out till 50-100 to see further improvement in obtaining the best model. We believe these results can be improved if one can try other implementation

of copy decoder and with the tuning of all other hyper-parameters in the encoder-decoder network. The future work might also include trying PPDB dataset with a smaller version or removing short lexical or phrase sentences due to which we were not able to get complete sentences., which we think can improve the quality of generated paraphrase. One can also try other paraphrases dataset to see the improvement in the quality of generated paraphrases. One could also try coverage with attention to not repeat words which were already used to generate paraphrases, which can further improve the quality of generated paraphrases in COPYNET and Pointer softmax pointer network.

In chapter 4, we examined many different variations of a text classifier, and we achieved significant improvement in test accuracy with CNN. The results were consistent with other papers and results. Here we attempted to put our genuine efforts to get reasonable high accuracy for creating a strong and stable baseline. The results are represented in section 4.3. Here, one can try other datasets or another downstream task which we can further use to evaluate the quality of generated paraphrases.

In chapter 5, we evaluate the quality of generated paraphrases in the downstream task, i.e., in text classification. According to our analysis, the best way of evaluating the generated paraphrases is to evaluate their performance in the downstream task. All the metrics we have till now, such as BLEU, TER, METEOR have their limitations, which can give wrong indications about the performance of the model and the quality of generated paraphrase. So one should always try to do analysis using the downstream task. As shown in the chapter 3, the model performance is not very bad, but after using paraphrases in the downstream task, the quality of these paraphrases decreases to a greater extent. We show how the performance of downstream tasks is highly correlated with higher BLEU and METEOR score or quality of generated paraphrases. However, in the future, these results can also be tested on different downstream task with the different dataset for supervision. Our analysis can also be used further to improve the quality of generating paraphrases in new models.

To our best knowledge, this evaluation of generating paraphrases in downstream

task with COPYNET and pointer softmax network in seq2seq model is performed for the first time. Our results can be used as the baseline to further improve the quality of test accuracy on IMDb movie review dataset by augmenting the training dataset. Our analysis can also be used to the future development of paraphrase generation model and to improve the attention and pointer network configuration.

Bibliography

[1] N. Aggarwal, K. Asooja, and P. Buitelaar, "Pushing corpus based relatedness to similarity: Shared task system description", in *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, ser. SemEval '12, Association for Computational Linguistics, 2012, pp. 643–647.

[2] I. Androutsopoulos and P. Malakasiotis, "A survey of paraphrasing and textual entailment methods", *Journal of Artificial Intelligence Research*, vol. 38, pp. 135–187, 2010.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", in *Proceedings of The International Conference on Learning Representations*, 2015, pp. 1–15.

[4] R. Barzilay and L. Lee, "Learning to paraphrase: An unsupervised approach using multiple-sequence alignment", in *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2003, pp. 16–23.

[5] S. Biggins, S. Mohammed, S. Oakley, L. Stringer, M. Stevenson, and J. Priess, "Two approaches to semantic text similarity", in *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, ser. 12, Association for Computational Linguistics, 2012, pp. 655–661.

[6] D. Buscaldi, R. Tournier, N. Aussenac-Gilles, and J. Mothe, "Irit: Textual similarity combining conceptual similarity with an n-gram comparison method", in *The First Joint Conference on Lexical and Computational Semantics – Volume 1:*

Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics, 2012, pp. 552–556.

[7] D. Bär, C. Biemann, I. Gurevych, and T. Zesch, "Ukp: Computing semantic textual similarity by combining multiple content similarity measures", in *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, Association for Computational Linguistics, 2012, pp. 435–440.

[8] M. E. Califf and R. J. Mooney, "Bottom-up relational learning of pattern matching rules for information extraction", *Journal of Machine Learning Research*, vol. 4, pp. 177–210, 2003.

[9] Z. Cao, C. Luo, W. Li, and S. Li, "Joint copying and restricted generation for paraphrase", vol. *Proceedings of the Thirty-First Association for the Advancement of Artificial Intelligence conference on Artificial Intelligence*, 2017, pp. 3152–3158.

[10] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", in, vol. *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Association for Computational Linguistics, 2014, pp. 103–115.

[11] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", vol. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2014, pp. 1724–1734.

[12] A. Chorianopoulou, "Investigation paraphrasing algorithms with application to spoken dialogue systems", *Diploma thesis, Technical University of Crete Chania, Greece*, 2013.

[13] L. Dong, J. Mallinson, S. Reddy, and M. Lapata, "Learning to paraphrase for question answering", Association for Computational Linguistics, 2017, pp. 886–897.

[14] P. A. Duboue and J. Chu-Carroll, "Answering the question you wish they had asked: The impact of paraphrasing for question answering", in *Proceedings of*

the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics , Companion Volume: Short Papers, Association for Computational Linguistics, 2006, pp. 33–36.

[15] P. G. Edmonds, "Semantic representations of near-synonyms for automatic lexical choice", PhD thesis, 1999.

[16] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2016, pp. 1631–1640.

[17] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2016, pp. 140–149.

[18] Z. S. Harris, "Distributional structure", *Word*, vol. 10, pp. 146–162, 1954.

[19] Z. S. Harris, *Transformational Theory*. Springer Netherlands, 1970, pp. 533–577.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, pp. 1735–80, Dec. 1997.

[21] S. B. Huffman, "Learning information extraction patterns from examples", in *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, S. Wermter, E. Riloff, and G. Scheler, Eds., Springer, 1996, pp. 246–260.

[22] D. Inkpen, "Building a lexical knowledge-base of near-synonym differences", *Computational Linguistics*, vol. 32, pp. 223–262, 2004.

[23] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, *Bag of tricks for efficient text classification*, 2016.

[24] Y. Kim, "Convolutional neural networks for sentence classification", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2014, pp. 1746–1751.

[25] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation", in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. 03, Association for Computational Linguistics, 2003, pp. 48–54.

[26] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments", in *Proceedings of the Second Workshop on Statistical Machine Translation*, ser. 07, Association for Computational Linguistics, 2007, pp. 228–231.

[27] "Learning dictionaries for information extraction by multi-level bootstrapping", English, in *Proceedings of the National Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, 1999, pp. 474–479.

[28] Y. LeCun and Y. Bengio, *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Massachusetts Institute of Technology Press, 1998, ch. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", in *Intelligent Signal Processing*, S. Haykin and B. Kosko, Eds., Institute of Electrical and Electronics Engineers Press, 2001, pp. 306–351.

[30] Z. Lin, H. T. Ng, and M.-Y. Kan, "Automatically evaluating text coherence using discourse relations", in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, Association for Computational Linguistics, 2011, pp. 997–1006.

[31] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation", vol. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2015, pp. 1412–1421.

[32] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis", in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2011, pp. 142–150.

[33] P. Malakasiotis, "Paraphrase and textual entailment recognition and generation", *PhD thesis, Department of Informatics, Athens University of Economics and Business*, 2011.

[34] I. Mani, *Automatic Summarization*. John Benjamins Publishing Company, Jan. 2001, vol. 3, pp. 29–36.

[35] K. R. McKeown, "Paraphrasing questions using given and new information", *Computational Linguistic*, vol. 9, no. 1, pp. 1–10, 1983.

[36] R. Mihalcea, C. Corley, C. Strapparava, *et al.*, "Corpus-based and knowledge-based measures of text semantic similarity", in *The Association for the Advancement of Artificial Intelligence*, vol. 6, 2006, pp. 775–780.

[37] R. Mitkov and E. Hovy, *Text Summarization*. Oxford University Press, Sep. 2012, ch. 32, pp. 583–598.

[38] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation", in *Proceedings of the 40th annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[39] E. Pavlick, J. Bos, M. Nissim, C. Beller, B. Van Durme, and C. Callison-Burch, "Adding semantics to data-driven paraphrasing", in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, 2015, pp. 1512–1522.

[40] E. Pavlick and C. Callison-Burch, "Simple ppdb: A paraphrase database for simplification", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics, 2016, pp. 143–148.

[41] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification", in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Association for Computational Linguistics, 2015, pp. 425–430.

[42] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation", vol. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2014, pp. 1532–1543.

[43] A. Prakash, S. A. Hasan, K. Lee, V. V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural paraphrase generation with stacked residual LSTM networks", in

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, The COLING 2016 Organizing Committee, 2016, pp. 2923–2934.

[44] C. Quirk, C. Brockett, and W. Dolan, "Monolingual machine translation for paraphrase generation", in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004, pp. 142–149.

[45] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu, "Statistical machine translation for query expansion in answer retrieval", in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ser. 07, 2007, pp. 464–471.

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research", *Nature*, vol. 323, J. A. Anderson and E. Rosenfeld, Eds., pp. 696–699, 1988.

[47] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks", vol. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2017, pp. 1073–1083.

[48] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert, "Crystal: Inducing a conceptual dictionary", *Proceeding 14th International Joint Conference on Artificial Intelligence*, pp. 1314–1319, Jun. 1995.

[49] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[50] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Coverage-based neural machine translation", *Computing Research Repository*, vol. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–85–430, 2016.

[51] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks", in *Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2692–2700.

[52] F. Xu, H. Uszkoreit, and H. Li, "A seed-driven bottom-up machine learning framework for extracting relations of various complexity", in *Proceedings 45th*

Annual Meeting of the Association for Computational Linguistics 2007, Association for Computational Linguistics, Jun. 2007, pp. 584–591.