METHODS OF SENTENCE EXTRACTION, ABSTRACTION AND ORDERING FOR AUTOMATIC TEXT SUMMARIZATION

MIR TAFSEER NAYEEM
Bachelor of Science, Islamic University of Technology, 2011

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

METHODS OF SENTENCE EXTRACTION, ABSTRACTION AND ORDERING FOR AUTOMATIC TEXT SUMMARIZATION

MIR TAFSEER NAYEEM

Date of Defence: September 5, 2017

Dr. Yllias Chali Supervisor	Professor	Ph.D.
Dr. Wendy Osborn Committee Member	Associate Professor	Ph.D.
Dr. John Anvik Committee Member	Assistant Professor	Ph.D.
Dr. Howard Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

I dedicate this thesis to my mother who inspired me at each and every step of my life and to my late father who could not see this thesis completed.

Abstract

In this thesis, we have developed several techniques for tackling both the extractive and abstractive text summarization tasks. We implement a rank based sentence selection which can retain the most important and non-redundant contents to form the summary. For ensuring a pure sentence abstraction, we propose several novel sentence abstraction techniques which jointly perform sentence compression, fusion and paraphrasing at the sentence level. We also model abstractive compression generation as a sequence-to-sequence (**seq2seq**) problem using an encoder-decoder framework, which is also a novel inclusion according to the state-of-the-art text summarization systems. We propose simple yet effective solutions to several common problems in neural **seq2seq** models such as redundant repetition and unknown token replacement. Our sentence level models improve the informativity as well as the grammaticality of the generated sentences. Furthermore, we applied our sentence abstraction techniques to the multi-document text summarization. We also propose a greedy sentence ordering algorithm to maintain the summary coherence for increasing the readability. We introduce an optimal solution to the summary length limit problem. For the sentence level tasks, we conduct our experiments on human generated abstractive compression datasets and evaluate our system on several newly proposed Machine Translation (MT) evaluation metrics. In the case of the document level summary, we conduct experiments on the Document Understanding Conference (DUC) 2004 datasets using ROUGE toolkit. Our experiments demonstrate that the methods bring significant improvements over the state-of-the-art methods. At the end of this thesis, we also introduced a new concept called "**Reader Aware Summary**" which can generate summaries for some critical readers (e.g. Non-Native Reader).

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Yllias Chali for the continuous support, invaluable advice and encouragement. His guidance helped me to explore research challenges and thinking about scientific problems profoundly. The door to Prof. Chali's office was always open whenever I ran into a trouble spot or had a question about my research or writing. I am really grateful to him.

I would like to thank my M.Sc. supervisory committee members Dr. Wendy Osborn, and Dr. John Anvik for their time and effort spent on thesis committee meetings. They provided very valuable feedback and helpful suggestions to my research.

I am very grateful to Dr. Shafiq Joty who is always generous to share his brilliant thoughts and suggestions for my work. He has been and will always be an inspiration in my career.

I also must thank University of Lethbridge for the financial and travel support. I am also thankful to Natural Sciences and Engineering Research Council (NSERC) of Canada discovery grant for providing me a TITAN X GPU machine to conduct my experiments without which it would not have been possible for me to carry out the whole work.

I am forever thankful to my close friends, you should know that your support, friendship and encouragement was worth more than I can express on paper.

Last but not the least, I would like to thank my family: my mother and my brothers for their continuous and unparalleled love, help and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

"Text Summarization is the process of distilling the most important information from one or more texts to produce an abridged version for a particular task and user." (Section 23.3 of Jurafsky and Martin (2008))

Today's world is all about information, most of it online. The internet contains billions of textual documents and is growing at an exponential rate. Search engines such as Google, Bing, Yahoo usually return thousands of pages for a single query. As a result, we are facing a challenging problem of information overload. Moreover, the internet provides large collections of text on a variety of topics. This accounts for the redundancy in the texts and difficulty to obtain concise information. Users get so over-whelmed reading large amounts of text that they may skip reading many important and interesting documents. These concerns have initiated interest in the development of automatic summarization systems. Such systems are designed to take a single article, a cluster of news articles, a broadcast news show, user reviews or an email thread as input, and produce a concise and fluent summary of the most important information while being non-redundant, coherent and grammatically readable. A good summary can largely benefit individuals and organizations to achieve better decisions. For example, individual customers can consult reviews about the products written by other users to decide whether to buy or not. On the other hand, companies can leverage these reviews about their products to improve the products and the marketing strategy. However, it is almost impossible to read all the reviews given their sheer number.

1.2 Contributions of this Thesis

- We implemented an ILP (Integer Linear Programming) based sentence selection along with TextRank (Mihalcea and Tarau, 2004) scores and key phrases for extractive multi-document summarization. We further model the coherence using a greedy algorithm to increase the readability of the generated summary.

- We designed a novel abstractive sentence generation model which jointly performs sentence fusion and paraphrasing using the skipgram word embedding model. Furthermore, we applied our model to the abstractive multi-document summarization and got competitive results.

- We designed a neural paraphrastic compression model which jointly performs compression and paraphrasing in a single sentence in order to produce abstractive compression using a **seq2seq** encoder decoder model. According to the state of the art, this is the first neural model to jointly tackle compression and paraphrase in a single sentence. We propose simple yet effective solutions to several common problems in neural **seq2seq** models such as redundant repetition and unknown token replacement.

- We further proposed to use our neural paraphrastic compression at the document level. To the best of our knowledge, this is the first model to generate multi-sentence abstractive summary using a **seq2seq** encoder decoder model in a multi-document setting. Furthermore, we also introduced a new concept called "**Reader Aware Summary**" which can generate summaries for some critical readers (e.g. Non-Native Reader). Finally, we designed an optimal solution for the classical summary length limit problem which was not addressed in the past text summarization research in particular.

1.3 Overview of Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we provide an overview of automatic text summarization. We will also provide a brief introduction of the deep learning techniques especially used in text summarization. In Chapter 3, we will present our model for extractive coherent summarization in multi-document setting. Chapter 4 is devoted to our paraphrastic fusion model at the sentence level as well as the document level summary generation. Chapter 5 & 6 describe our novel approaches for the abstractive compression generation using a neural **seq2seq** encoder decoder model at the sentence, multi-document and reader level. Chapter 7 concludes and proposes directions for future research.

Chapter 2

Background

2.1 Automatic Text Summarization : A Recent Overview

The task of automatic document summarization aims at finding the most relevant information in a text and presenting them in a condensed form. A good summary should retain the most important contents of the original document or a cluster of document, while being coherent, non-redundant and grammatically readable. There are two types of summarizations: abstractive summarization and extractive summarization. Abstractive methods, which are still a growing field are highly complex as they need extensive natural language generation to rewrite the sentences. Therefore, the research community is focusing more on extractive summaries, which selects salient (important) sentences from the source document without any modification to create a summary. Summarization is classified as single-document or multi-document based upon the number of source documents. The information overlap between the documents from the same topic makes the multi-document summarization more challenging than the task of summarizing single documents.

2.1.1 Extractive Summarization

Over the past few decades, several extractive approaches have been developed for automatic summary generation that implements a number of machine learning, graph-based and optimization techniques. LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004) are graph-based methods of computing sentence importance for text summarization. The RegSum system (Hong and Nenkova, 2014) employs a supervised model for predicting word importance. Treating multi-document summarization as a submodular maximization problem has proven successful by (Lin and Bilmes, 2011). Instead of greedily adding sentences to form a summary. The most widely used practice is to formulate the problem as integer linear programming (ILP). Therefore, concept-based ILP (Gillick and Favre, 2009) has been proposed where the goal is to maximize the sum of the weights of the concepts (usually implemented as bigrams) that appear in the summary. Unfortunately, none of the above systems cares about the coherence of the final extracted summary.

In the very recent works using a neural network, Cheng and Lapata (2016) proposed an attentional encoder-decoder and (Nallapati et al., 2017) used a simple recurrent network based sequence classifier to solve the problem of extractive summarization. However, they are limited to single document settings, where sentences are implicitly ordered according to the sentence position in the original document. Parveen and Strube (2015); Parveen et al. (2015) proposed graph-based techniques to tackle coherence, which is also limited to single document summarization. Moreover, a recent work (Wang et al., 2016) actually proposed a multi-document summarization system that combines both coherence and informativeness but this system is limited to syntactic linkages between named entities.

2.1.2 Abstractive Summarization

Abstractive summarization is generally much more difficult. It involves sophisticated techniques for meaning representation, content organization, sentence compression, sentence fusion and paraphrasing. There has been a huge interest in compressive document summarization that tries to compress original sentences to form a summary (Clarke and Lapata, 2006, 2008; Filippova, 2010) as a first intermediate step towards abstractive summarization. Compression summarization techniques include sentences which are compressed from original sentences without further modifications other than word deletion. Sentence compression involving two or more sentences is called MSC (Multi-Sentence Compression). Most of the previous MSC approaches rely on syntactic parsing to build a dependency tree for each related sentence in a cluster for producing grammatical compressions (Filippova and Strube, 2008). Unfortunately, syntactic parsers are not available for all the languages. As an alternative, word graph-based approaches that only require a Parts-of-Speech (POS) tagger and a list of stopwords have been proposed first by (Filippova, 2010). A directed word graph is constructed in which nodes represent words and edges represent the adjacency between words in a sentence. Compressed sentences are generated by finding k-shortest paths in the word graph. Boudin and Morin (2013) improved Filippova's approach by re-ranking the fusion candidate paths according to keyphrases to generate more informative sentences. However, grammaticality is sacrificed to improve informativity in these works.

Banerjee et al. (2015) proposed an abstractive multi-document summarization system using Filippova's sentence fusion approach (Filippova, 2010) combined with Integer Linear Programming (ILP) sentence selection. They chose random k-shortest paths generated from word graph and fed them into the ILP objective function. They used a 3-gram (Tri Gram) language model to ensure the linguistic quality of the compressed sentences. Following Banerjee's work, several recent approaches have been proposed with slight modifications. Multiword Expressions (MWE) was exploited in (ShafieiBavani et al., 2016) to produce more informative compressions. Recently, Tuan et al. (2017) use syntax factor along with Banerjee's model to generate compressions. However, all the above mentioned systems try to produce compressions by copying the source sentence words and no paraphrasing is involved in the process.

Recently end-to-end training with encoder-decoder neural networks has achieved huge success in the case of abstractive summarization. These systems have adopted techniques such as encoder-decoder with attention neural network models (Bahdanau et al., 2015; Luong et al., 2015) from the field of machine translation to model the sentence summarization task. Rush et al. (2015) was the first to use neural sequence-to-sequence learning in the headline generation task from a single document, the classical (DUC 2004, Task-1). Unfortunately, this line of research under the term sentence summarization (Rush et al., 2015), which can generate only a single sentence, is somewhat misleadingly called text summarization in some follow-up research works. Nallapati et al. (2016) from IBM also uses the same deep learning framework for the same task. However, it has extremely interesting additional techniques to improve performance like copying words from the source document. Chopra et al. (2016) extended (Rush et al., 2015) to keep the CNN (Convolutional Neural Networks) encoder but replace the decoder with Recurrent Neural Networks (RNN). Their experiments show that the CNN encoder with RNN decoder model performs better than (Rush et al., 2015). There are some limitations to the above mentioned models. One is that the source documents are quite small (about 1 paragraph or 500 words in the training dataset of the Annotated English Gigaword (Napoles et al., 2012)) and the produced output is also very short (about 75 characters). Rush et al. (2015) uses the first sentence of a document to pair with the headline of that document. It is argued that the first sentence is enough to capture the gist of a document, but it is not perfect. Like the headlines, their model produces some ungrammatical sentences.

In the case of **seq2seq** models, summarization tasks can benefit from copying words from input sentences shown in (Gu et al., 2016; Gulcehre et al., 2016). Gu et al. (2016) propose **CopyNet** to model the copying mechanism in response generation, which also applies for sentence summarization task. Gulcehre et al. (2016) propose a switch gate to control whether to copy from a source sentence or generate from the target decoder vocabulary. Unfortunately, there is still much to be done to adapt such encoder-decoder architectures to document summarization instead of single sentence generation. Encoding for generic documents, which typically contains multiple paragraphs or a collection of related documents, still lacks satisfactory solutions. Very recently, there are a few attempts (See et al., 2017; Paulus et al., 2017) which can generate multiple sentences from a single document. In this thesis, we will propose a model which can eventually generate multiple sentences from a collection of related documents.

2.1.3 Automatic Summary Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)[1] is an automatic tool to determine the quality of a machine generated summary by comparing it against a reference or a set of reference summaries (typically human-produced) (Lin, 2004). There are 4 different ROUGE metrics - namely ROUGE-N (1,2,3,4), ROUGE-L, ROUGE-W, and ROUGE-S.

- **ROUGE-N** applies co-occurrence statistics to evaluate a summary which measures unigram (one word), bigram (two word), trigram (three word) and higher order n-gram overlap.
- **ROUGE-L** measures the longest matching sequence of words using LCS (Longest Common Sub-sequence).
- **ROUGE-W** assigns different weights to consecutive in-sequence matches in LCS.
- **ROUGE-S** is any pair of word in their sentence order which allows for arbitrary gaps. This can also be called skip-gram co-occurrence. For example, skip-bigram measures the overlap of word pairs that can have a maximum of two gaps in between words[2]. As an example, for the phrase "cat in the hat" the skip-bigrams would be "cat in, cat the, cat hat, in the, in hat, the hat".

Among these above mentioned measures, **ROUGE-N** is used the most in multi-document summarization research. It counts the number of overlapping n-grams between the system summary and human written reference summaries. We can define ROUGE-N as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in R} \sum_{g_n \in S} Count_{match}(g_n)}{\sum_{S \in R} \sum_{g_n \in S} Count(g_n)}$$

where n is the length of the n-gram, g_n and $Count_{match}(g_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries (Lin,

[1]ROUGE package link: http://www.berouge.com

[2]http://www.rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/

2004). When multiple reference summaries are used for evaluation, pairwise summary-level ROUGE-N between a candidate machine generated summary s and every human produced reference r_i from the reference set $R = \{r_1, r_2, \dots, r_n\}$ is computed. The final ROUGE-N score is then obtained by taking the maximum of the summary-level ROUGE-N scores as follows:

$$\text{ROUGE-N}_{multi} = argmax_i\ (\text{ROUGE-N}(r_i, s))$$

Let us assume, we have the following system and reference summaries:

System Summary (machine generated): A man with a helmet painted red is riding a blue motorcycle.

Reference Summary (human produced): A man with a helmet is riding a blue motorcycle.

If we consider just the individual words (uni-gram), the number of overlapping words between the system summary and reference summary is 10. However, to get a good quantitative value, we can actually compute the precision and recall using the overlap of words. Recall in the context of ROUGE simply means how much of the reference summary the system summary is capturing. If we are only considering the individual words (uni-gram), the recall can be computed as:

$$\text{ROUGE-1 (recall)} = \frac{num\_of\_overlapping\_words}{total\_words\_in\_reference\_summary} = \frac{10}{10} = 1.0$$

This means that all the words in the human produced reference summary has been captured by the machine generated system summary. However, a machine generated summary (system summary) can be extremely long, capturing all words in the human produced reference summary. But, much of the words in the system summary may be useless, resulting a summary with redundant and repetitive information. This is where precision comes into play[3]. In terms of precision, what you are essentially measuring is, how much of the system summary was in fact relevant or needed? For the same example, precision is measured as:

$$\text{ROUGE-1 (precision)} = \frac{num\_of\_overlapping\_words}{total\_words\_in\_system\_summary} = \frac{10}{12} = 0.83$$

This simply means that 10 out of the 12 words in the system summary were relevant. Let us assume, we had the following system summary instead of the previous example:

System Summary 2 (machine generated): A man with a helmet painted red is riding a blue motorcycle down the road.

The Precision now becomes:

$$\text{ROUGE-1 (precision)} = \frac{10}{15} = 0.66$$

Now, the precision score has decreased, this is because we have quite a few redundant words in the system summary. The precision is really crucial when we try to generate summaries that are concise in nature. Therefore, it is always best to compute both the **Precision** and **Recall** and then report the **F-Measure**. If the system summaries are forced to be concise through some constraints (such as length limit constraint), then we could consider using just the recall since precision is of less concern in this scenario. In this thesis, we report only the limited length recall measure in our experiment. Moreover, we also report the performance in terms of **ROUGE-SU4**, where **S** means skip-bigram (match 2 non contiguous words with other words in between) which allows rephrasing and sentence reorganization. As the ROUGE score is supposed to evaluate abstractive summaries, its a good measure. For other in between words, we use **U4** which means maximum of 4 unigram words are allowed within a skip-bigram.

[3]http://text-analytics101.rxnlp.com/2017/01/how-rouge-works-for-evaluation-of.html

2.2 Word Embedding

This section introduces the concept of word embedding, which is a vector representation of words. It is a popular method used in many natural language processing applications, such as document classification, text summarization and question answering.

2.2.1 One-Hot Vectors

Building the above applications requires us to measure the similarity between two words, sentences or even paragraphs. One-hot vector is a representation of all the words through a vector space model. For each word, its vector representation has the corresponding entry in the vector as 1 (presence), and all other entries as 0 (absence). The lengths of one-hot vectors match the size of the dictionary or in more technical terms, vocabulary. Cosine similarity[4] on one-hot vectors cannot capture semantic information when documents say the same thing in completely different words. For instance, consider these two following news headlines:

- Obama speaks to the media in Illinois
- The President greets the press in Chicago

These have no content words in common (except for the stopwords such as *the* and *in*, which don't carry much information for a semantic similarity measurement), so according to the one-hot vectors representation, their cosine distance would be maximal. To calculate their semantic similarity accurately, we need further information, which we can learn from the large amounts of data through machine learning models (Kusner et al., 2015). The Figure 2.1 taken from (Kusner et al., 2015) visualizes the word to word similarity of the example headlines.

[4]https://en.wikipedia.org/wiki/Cosine_similarity

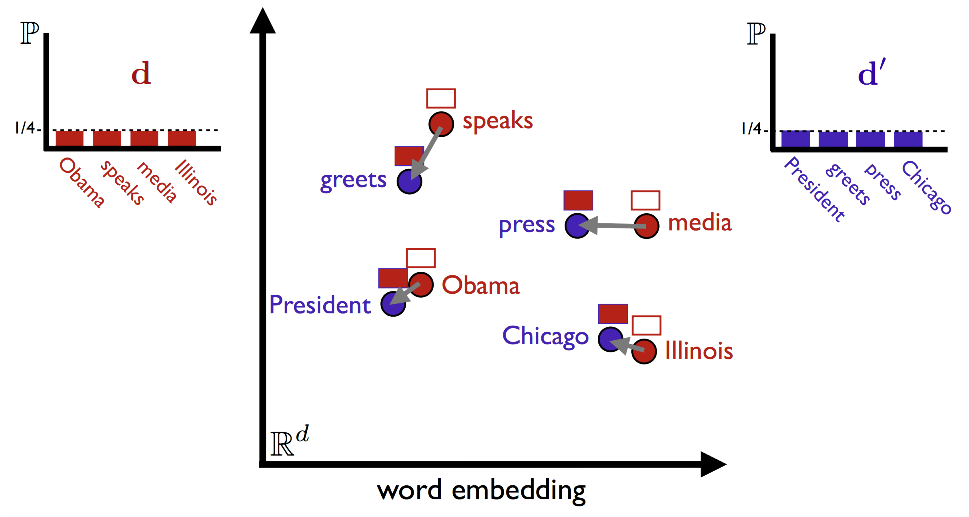

Figure 2.1: Visualization of word to word similarity of all non-stop words from both headlines are embedded into a word2vec space.

2.2.2 Word2Vec Embedding

Vector space models have been used in distributional semantics since the 1990s for estimating continuous representations of words, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Latent Semantic Analysis (LSA) (Landauer et al., 1998) are two such examples. The term word embedding was originally introduced by Bengio et al. (2003) by training a neural language model. The language models build the joint probability $P(w_1,\ldots,w_T)$ of a sentence, where w_i is the i^{th} word in the sentence. The language model assigns higher probabilities to grammatical and meaningful sentences, and lower probabilities to meaningless sentence constructions. Consider that we are searching something in the internet, from Figure 2.2[5], if we write “How long is a”, the search engine would suggest the word “football”. This is because the probability of “How long is a football” is very high according to the language model among all the words in the target vocabulary.

Word2vec (Mikolov et al., 2013b,a) is particularly the most popular of the word embedding models for learning word embeddings from large amount of texts to create high-dimensional (50 to 300 dimensional) representations of words in an unsupervised manner.

[5]Figure collected from http://book.paddlepaddle.org/04.word2vec/, this figure is under https://creativecommons.org/licenses/by-sa/4.0/

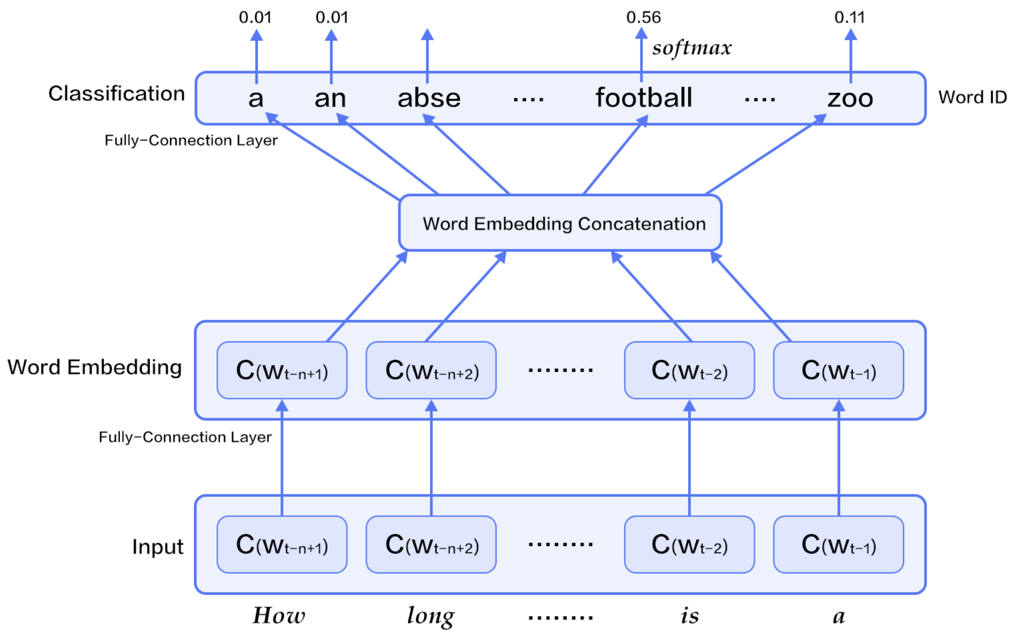

Figure 2.2: N-gram neural language model.

Word2Vec embeds words in a continuous vector space where semantically similar words are placed as nearby points to each other as illustrated in the Figure 2.3. It was recently shown that the word vectors capture many linguistic regularities. For example, vector arithmetic operations [vector("Paris") - vector("France") + vector("Italy")] results in a vector that is very close to vector("Rome"), and [vector("king") - vector("man") + vector("woman")] is close to vector("queen")[6]. Mikolov et al. (2013b) defined two architectures for learning word embeddings, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model.

Continuous Bag-of-Words model (CBOW): Unlike a language model that can only base its predictions on past words, the CBOW model predicts the current word based on the N words both before and after it. When N=2, the model is as the Figure 2.4 (left).

Skip-gram model: Instead of using the surrounding words, skip-gram uses the centre word to predict the surrounding words as can be seen in Figure 2.4 (right).

[6]https://code.google.com/archive/p/word2vec/

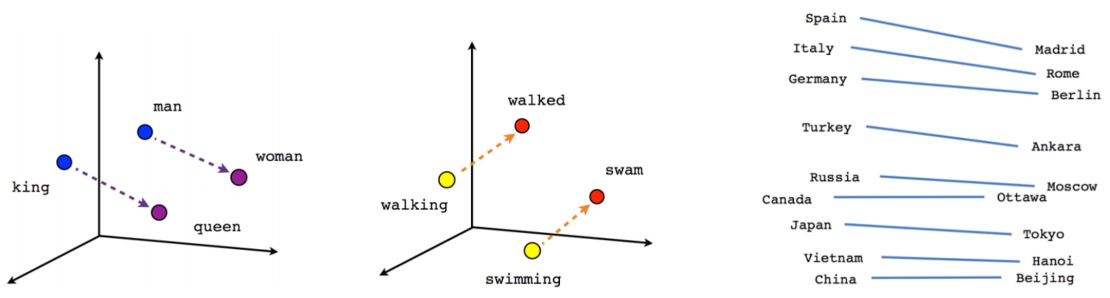

Figure 2.3: Visualization of semantic relationships, e.g. male-female, verb tense and even country-capital relationships between words (Mikolov et al., 2013b).

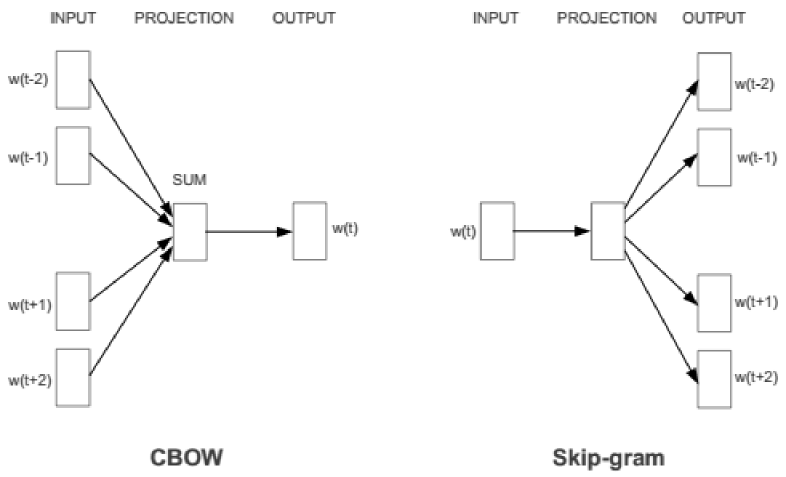

Figure 2.4: CBOW model (left) and Skip-gram model (right) from (Mikolov et al., 2013b).

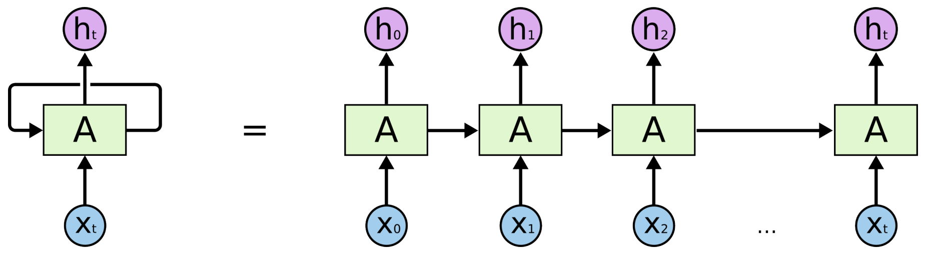

Figure 2.5: An unrolled recurrent neural network (Image is taken from the source: http://colah.github.io/posts/2015-08-Understanding-LSTMs)

2.2.3 GloVe Embedding

In contrast to word2vec, GloVe[7] (Pennington et al., 2014) takes advantage of two primary families of word vectors, i.e. global matrix factorization methods (e.g. LSA (Landauer et al., 1998)) and local context window based methods (e.g. skip-gram (Mikolov et al., 2013b)). Moreover, word2vec is a "predictive" model, whereas GloVe is a "count-based" model. GloVe builds a co-occurrence matrix for the entire corpus first, then factorizes it to yield matrices for word vectors and context vectors.

2.3 Recurrent Neural Network (RNN)

In a traditional neural network, we assume that all the inputs and outputs are not dependant on each other. But for many tasks it is not a good idea. If we want to predict the next word in a sentence we need to know which words came before it. Recurrent neural networks are generally good for data where there is a relation between previous inputs and the current input in a sequence. As Natural Language Processing (NLP) is a classical problem on sequential data, the RNNs have shown great success in many NLP tasks in the last few years, such as language modeling, syntax parsing, image captioning, dialog generation, machine translation, summarization and question answering.

As shown in Figure 2.5, by unfolding an RNN at the t-th time step, the network takes two inputs: the t-th input vector $\vec{x_t}$ (Normally, the embedded input word goes through an RNN as $e(\vec{x_t})$ at every time step) and the hidden state from the last time-step $\vec{h_{t-1}}$. From

[7]https://github.com/stanfordnlp/GloVe

those, it computes the hidden state of the current time-step $\vec{h_t}$. This process is repeated until all inputs are processed in sequence. Considering the RNN as function f, the formulation is:

$$\vec{h_t} = f(\vec{x_t}, \vec{h_{t-1}})$$

2.3.1 Long Short Term Memory (LSTM)

One of the essential properties of RNNs is that they are able to connect previous information to the present situation. Sometimes, we only need to look at recent information to describe the present situation. For example, consider a language model trying to predict the last word based on the previous ones in a sentence “How long is a football ***match***”. We actually do not need any further context, the next word is going to be ***match***. In such cases, where the gap between the relevant information and the place that it is needed is small, RNNs can learn to use the past information. In contrast, we try to predict the last word of the sentence “I grew up in Bangladesh, I can speak fluent ***Bengali***”. Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need some context of ***Bangladesh***, which is further back from the last word. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information[8].

Long Short Term Memory networks usually called LSTMs (Hochreiter and Schmidhuber, 1997) are a special kind of RNN, capable of avoiding the long-distance dependencies problem (Bengio et al., 1994). They work exceptionally well, and are widely used on a large variety of NLP problems recently.

In comparison to the structure of a RNN, an LSTM includes a memory cell c, an input gate i, a forget gate f and an output gate o. These gates and memory cells have the ability to avoid the long term dependencies problem. We can formulate the LSTM denoted as a function f, as follows:

[8]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

$$h_t = f(x_t, h_{t-1})$$

f contains following formulations from (Hochreiter and Schmidhuber, 1997),

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{h-1} + W_{ci}c_{t-1} + b_i) \tag{2.1}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{h-1} + W_{cf}c_{t-1} + b_f) \tag{2.2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{h-1} + b_c) \tag{2.3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{h-1} + W_{co}c_t + b_o) \tag{2.4}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.5}$$

In the above equations, i_t, f_t, c_t, o_t stand for input gate, forget gate, memory cell and output gate respectively. W and b are model parameters, **tanh** is the hyperbolic tangent, and $\odot$ denotes an element-wise product operation as shown in Figure 2.6.

2.3.2 Gated Recurrent Unit (GRU)

GRU (Cho et al., 2014b) is related to a LSTM, but both uses a different gating mechanism to prevent long-distance dependencies problem. GRUs are relatively new, have a less complex structure, train faster, computationally more efficient and perform better than a LSTM on less training data (Chung et al., 2014). GRU also controls the flow of information like the LSTM unit, but without having to use a memory unit, and combines the forget and input gates into a single "update gate". GRU just exposes the full hidden content without any control (Cho et al., 2014b).

A GRU layer is quite similar to a LSTM layer, the following equations are for a single GRU layer (Cho et al., 2014b):

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$

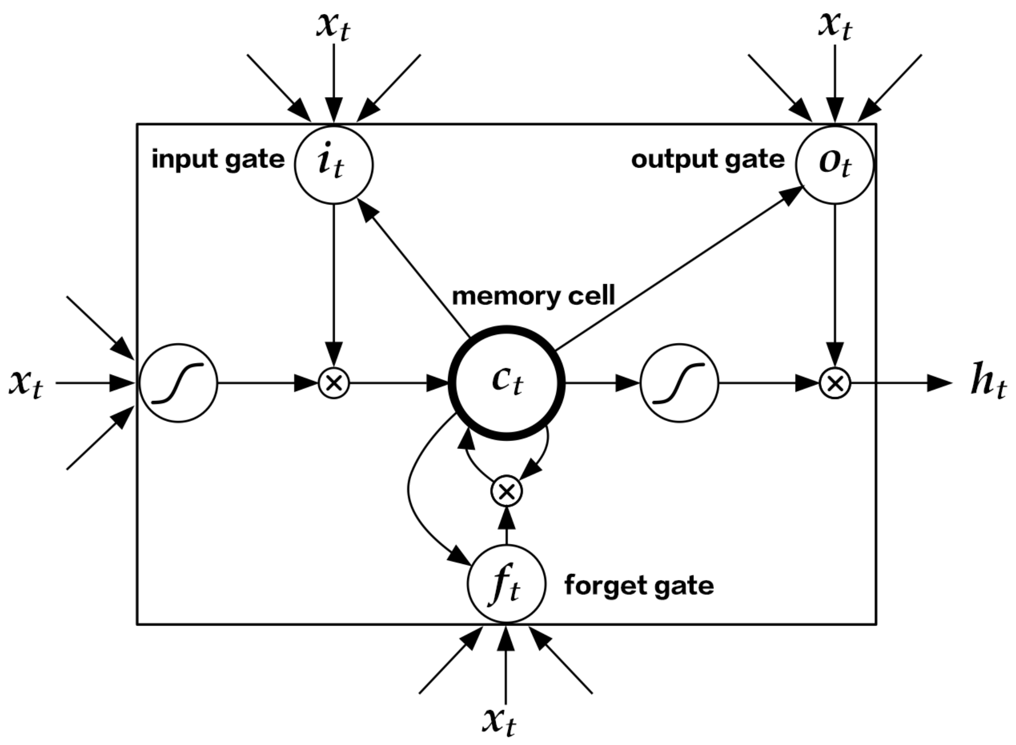

Figure 2.6: LSTM at time step *t* (Hochreiter and Schmidhuber, 1997)

$$r = \sigma(x_t U^r + s_{t-1} W^r)$$

$$h = tanh(x_t U^h + (s_{t-1} \odot r) W^h)$$

$$s_t = (1 - z) \odot h + z \odot s_{t-1}$$

In the above equations, a GRU has two gates, a reset gate r, and an update gate z. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep as shown in Figure 2.7. For all recurrent units the general formulation is,

$$h_t = Recurrent(x_t, h_{t-1})$$

where *Recurrent* is a unit which can be a simple RNN, GRU or LSTM.

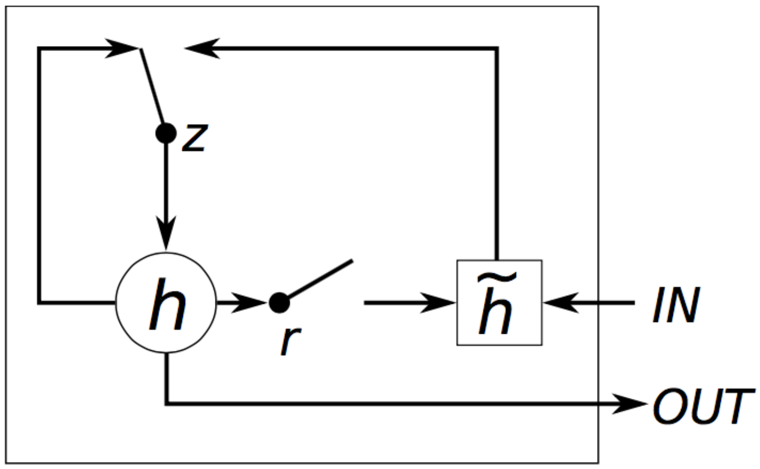

Figure 2.7: GRU Gating Mechanism (Chung et al., 2014)

2.4 Extensions of Recurrent Neural Network (RNN)

2.4.1 Bi-directional RNNs

RNNs can summarize the history of all the previous inputs seen up untill now. However, they can not see the future input tokens. Because most NLP tasks provide the entire sentence as input, sequential learning can benefit from having the future encoded tokens as well as the history seen so far.

The bidirectional recurrent neural network (bi-RNN) (Schuster and Paliwal, 1997) has been successfully used recently in speech recognition (Graves et al., 2013) and machine translation (Bahdanau et al., 2015). In a bi-RNN, a backward RNN layer (can be a simple RNN, LSTM or GRU) processes the sequence in the reversed direction with regards to its immediate forward RNN layer. The forward RNN encodes the source sequence in its original order $(x_1, x_2, \ldots, x_T)$ from left-to-right and generates a sequence of hidden states $(\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_T})$. The backward RNN encodes the source sequence in reverse order, from right-to-left $(x_T, x_{T-1}, \ldots, x_1)$ and generates $(\overleftarrow{h_1}, \overleftarrow{h_2}, \ldots, \overleftarrow{h_T})$. Then for each word x_i, its complete hidden state is the concatenation of the corresponding hidden states from the two RNNs, i.e., $h_i = \left[\overrightarrow{h_i^T}, \overleftarrow{h_i^T}\right]^T$. Therefore, RNN layers at time-step t can see both the history

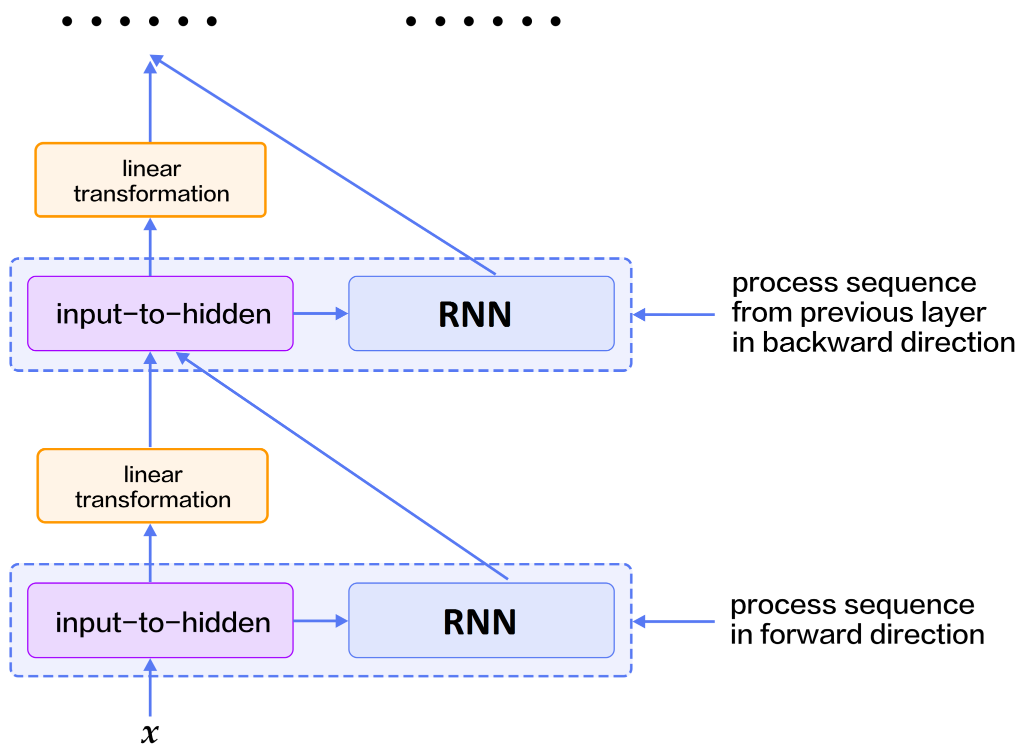
linear transformation
input–to–hidden
RNN
process sequence from previous layer in backward direction
linear transformation
input–to–hidden
RNN
process sequence in forward direction
x

Figure 2.8: Bi-Directional Recurrent Neural Network (bi-RNN) (Schuster and Paliwal, 1997)

and the future. Figure. 2.8 illustrates the bidirectional recurrent neural networks.

2.4.2 Stacking multiple RNNs

An additional important modification we can do for providing more expressive power to the model is by stacking multiple RNNs, LSTMs, GRUs, or any other neural network layer, to process the data. The output of the first layer will become the input of the second and so on. Figure 2.9 shows an example of stacked RNNs. For example, in a 3-layer stacked RNN, the calculation at time step t would look as follows:

$$h_{1,t} = \mathbf{RNN_1}(x_t, h_{1,t-1})$$

$$h_{2,t} = \mathbf{RNN_2}(h_{1,t}, h_{2,t-1})$$

$$h_{3,t} = \mathbf{RNN_3}(h_{2,t}, h_{3,t-1})$$

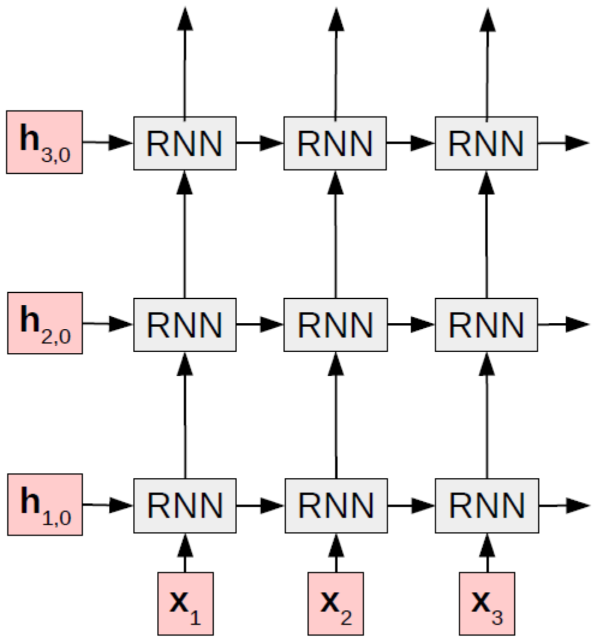

Figure 2.9: 3-layer stacked RNN (Neubig, 2017)

where $h_{n,t}$ is the hidden state for the nth layer at time step t of a RNN. Similarly, we could substitute RNN with LSTM(.), GRU(.), or any other recurrent unit (Neubig, 2017).

2.5 Neural Machine Translation (NMT)

Machine translation (MT) is the process of translating from the source language to the target language. We call the input language to the machine translation system the source language, and the output language as the target language. Therefore, machine translation is actually the task of converting a sequence of words in the source language into a sequence of words in the target language. It is one of the most important research topics in the field of natural language processing (Neubig, 2017).

Statistical machine translation (SMT) techniques have been developed for early automatic MT systems (Brown et al., 1993). However, these statistical machine translation models have many shortcomings. The SMT models heavily rely on pre-processing techniques like word alignment, word segmentation and tokenization, rule-extraction and syntactic parsing. The error introduced in any of these steps could accumulate and impact the translation quality. Moreover, human designed features cannot cover all possible linguistic variations and cannot use all global features[9]. The recent development of deep learning provides new solutions to these aforementioned problems of SMT. Neural Machine Translation (NMT) (Sutskever et al., 2014) does not rely on pre-designed features. Rather, the goal of NMT is to design a fully trainable model of which every component is tuned based on large parallel training corpora to maximize its translation performance.

A fully trainable NMT model $\mathcal{M}$ starts from a raw representation of a source sentence and finishes by generating a raw representation of a target sentence, considering a sequence of words as the most raw representation of a sentence. Recently, there are some methods which consider characters as the basic representation of a sentence (Costa-jussà and Fonollosa, 2016; Lee et al., 2017). In our case, each word in a sequence is represented by its integer index in a fixed numbered vocabulary. For instance, in the vocabulary V of English words sorted according to their frequency of appearance in a training corpus, the first most frequent word is represented as an integer 1. Let, $X = (x_1, x_2, \ldots, x_N)$ be a source sentence, and $Y = (y_1, y_2, \ldots, y_M)$ be a target sentence (Note that, N and M are not necessarily the same number of words) (Sutskever et al., 2014). Given a source sequence $X = (x_1, x_2, \ldots, x_N)$ of word indices, the NMT model $\mathcal{M}$ tries to find an output sequence Y that maximizes the conditional probability of Y given an input sequence X:

$$\underset{\mathbf{Y} \in V}{arg\ max}\ P(\mathbf{Y}|\mathbf{X})$$

The sequence-to-sequence networks (or **seq2seq** for short) has received great attention from the NLP community to solve the problem of NMT (Sutskever et al., 2014; Bahdanau et al., 2015). In seq2seq, we have input and output sequences of different lengths (N and M in the case of NMT).

As illustrated in the Figure 2.10, we have "ABC" as the input sequence, and "WXYZ" as the output sequence. The lengths of the two sequences are different. So, how does the

[9]http://book.paddlepaddle.org/08.machine_translation/

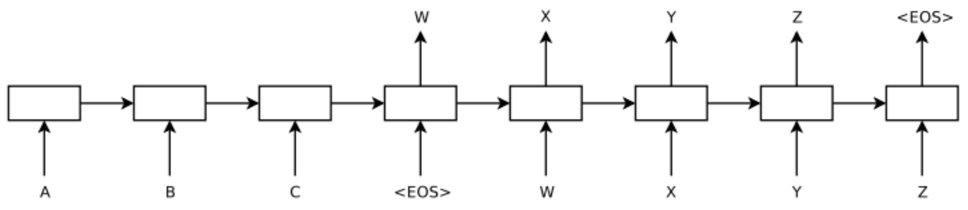

Figure 2.10: Sequence to Sequence Learning with Neural Networks (Sutskever et al., 2014)

seq2seq approach solve that problem of different sequence lengths? The answer is: they create a model which consists of two separate recurrent neural networks called **Encoder** and **Decoder** respectively.

2.5.1 Encoder-Decoder Framework

The Encoder-Decoder framework (Cho et al., 2014b) solves the mapping of a sequence to another sequence, for sequences with different lengths. The encoder turns a source sequence of words into a fixed size feature vector, which is then decoded by a decoder as a target sequence by maximizing the predictive probability. Both the encoder and the decoder are typically implemented via a simple RNN, LSTM or GRU.

Encoder

There are three steps for encoding a sentence as a sequence through an encoder:

1. In the case of the one-hot vector representation of a word where each word x_i in the source sentence $x = \{x_1, x_2, \ldots, x_N\}$ is represented as a vector $w_i \varepsilon \{0,1\}^{|V|}, i = 1, 2, \ldots, N$ where w_i has the same dimension as the size of the full word dictionary $|V|$, and has an element of one at the location corresponding to the location of the word in the dictionary and zero elsewhere.

2. As described earlier, there are two problems with the one-hot vector representation.

   - The dimension of each individual word vector is very large.

- It is hard to capture semantic relationship between words in a source sentence. Therefore, it is useful to project the one-hot vector into a low-dimensional semantic space as a dense vector with fixed dimensions. For instance, $s_i = Cw_i$ for the i-th word, with $C \varepsilon R^{K \times |V|}$ as the projection matrix and K is the dimensionality of the word embedding vector and $|V|$ is the size of the fixed vocabulary.

3. The source sequence of words is encoded via RNN: This can be described mathematically as:

$$h_i = \varnothing_\theta \left(h_{i-1}, s_i \right)$$

where, h_0 is a zero vector, $\varnothing_\theta$ is a non-linear activation function (e.g. sigmoid, ReLU, tanh), and $\mathbf{h} = \{h_1, \ldots, h_N\}$ is the sequential encoding of the first N words from the input source sequence. After the last words continuous vector s_N is read, the RNNs internal state h_n represents a summary of the whole source sentence.

Decoder

The task of the decoder is to maximize the probability of the next correct word in the target language sequence. The main idea of a decoder is as follows:

1. At each time step i, given the encoding vector or summary vector (or context vector) c of the source sentence sequence, the current i-th word u_i and the hidden state z_i, the next hidden state z_{i+1} is computed as:

$$z_{i+1} = \phi_\theta \left(c, u_i, z_i \right)$$

where ϕ_θ is a non-linear activation function and $c = q\mathbf{h}$ is the context vector of the source sentence sequence, c can be defined as $c = h_T.u_i$ denotes the i^{th} word from the target language sequence and u_0 denotes the beginning of the target language sequence (i.e., usually denoted by $< s >$), which indicates the beginning of the decoding. z_0 is an all zero vector and z_i is the RNN hidden state at time step i.

2. By calculating the probability p_{i+1} for the $i+1$-th word in the target language sequence as follows:

$$p\left(u_{i+1}|u_{<i+1},\mathbf{x}\right) = softmax(W_s z_{i+1} + b_z)$$

   where, $W_s z_{i+1} + b_z$ scores each possible words in the vocabulary $|V|$ and then the scores are normalized via **softmax** to convert the scores into probability p_{i+1} for the $i+1$-th word in the whole target sequence.

3. Compute the cost according to p_{i+1} and u_{i+1}.

4. Repeat the steps 1-3, until all the words in the target sentence have been processed which usually terminated by a $< eos >$ token.

2.5.2 Training: Maximum Likelihood Estimation (MLE)

After establishing the neural translation model, one needs to train the model with a sufficient amount of parallel data. Maximum log-likelihood estimation (MLE)[10] is a common statistical technique to train a previously described encoder-decoder model. Consider a parallel corpus D, where each sample in the corpus is a pair (X^n, Y^n) of source and target sentences. Each sentence is a sequence of integer indices corresponding to words in the full vocabulary set $|V|$, which is equivalent to a sequence of one-hot vectors. Multiplying an one-hot vector with a embedding matrix (from the left) is equivalent to taking the i^{th} column of the matrix, where the i^{th} element of the one-hot vector is 1. Given any pair from the corpus, the NMT model can compute the conditional log-probability of Y^n given X^n: $\log P(Y^n|X^n, \theta)$, where, θ is the training parameter and we write the log-likelihood of the whole training corpus as,

[10]https://en.wikipedia.org/wiki/Maximum_likelihood_estimation

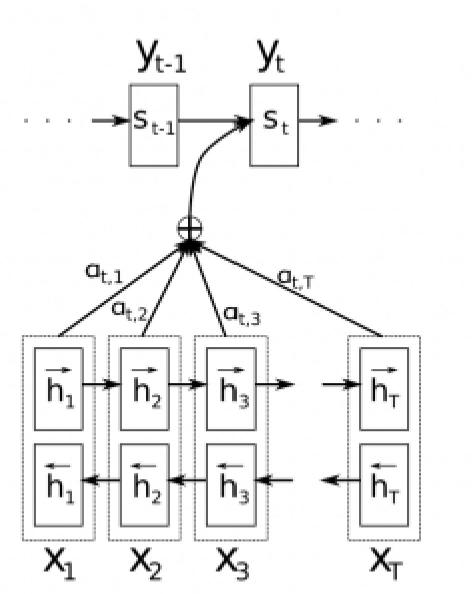
y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
X_1 X_2 X_3 X_T

Figure 2.11: Attention Model (Bahdanau et al., 2015)

$$L_t(\theta) = \sum_{(x,y)\in D} log\ P(\mathbf{Y}|\mathbf{X};\theta)$$

$$P(\mathbf{Y}|\mathbf{X};\theta) = \prod_{t=1} P(y_t|y_{1:t-1},\mathbf{X})$$

The generation process of machine translation is to translate the source sentence into a sentence in the target language according to a pre-trained model. In the decoding step, there are different strategies for generating next word in the output sequence such as greedy search and beam search (see section 2.5.4 and section 2.5.5 for details).

2.5.3 Attention Mechanism

There are a few problems associated with the fixed dimensional word vector representation from the encoding stage:

1. It seems unreasonable to encode all the information (e.g. syntactic and semantic) for a sentence with a fixed dimensional vector representation regardless of the length of

the sentence to be encoded. In theory, architectures like LSTMs should be able to deal with this, but in practice long-range dependencies are still problematic due to the vanishing gradient problem[11].

2. While translating a source input sentence, we generally pay more attention or concentration to the parts (e.g. in our case words) in the source sentence more relevant to the output translation, which is currently in the decoding stage. Moreover, the focus in the source sentence changes along the process of the translation. With a fixed dimensional vector, all the words or tokens from the source sentence are treated equally. This is not reasonable in any circumstances. Therefore, (Bahdanau et al., 2015) introduced attention mechanism for the first time in NMT (see Figure 2.11), which can decode based on different fragments of the context sequence in order to address the difficulty of feature learning for long sentences[12]. With an attention mechanism we no longer try to encode the full source input sentence into a fixed-length vector. Rather, we allow the decoder to attend the different parts of the source sentence at each time step of the output generation. In the case of a decoder with attention, the z_{i+1} is computed as:

$$z_{i+1} = \phi_\theta (c_i, u_i, z_i)$$

During each time step in the decoder, instead of using a fixed context (last hidden state of encoder), a distinct context vector c_i is used for generating word y_i. This context vector c_i is basically the weighted sum of the RNN hidden states (h_j) of the encoder. The weight a_{ij} denotes the strength of attention of the i^{th} word in the target language sentence to the j^{th} word in the source sentence.

[11]https://www.quora.com/What-is-the-vanishing-gradient-problem
[12]http://book.paddlepaddle.org/08.machine_translation/

$$c_i = \sum_{j=1}^{N} a_{ij} h_j$$

$$a_i = [a_{i1}, a_{i2}, \ldots, a_{iN}]$$

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{N} exp(e_{ik})}$$

$$e_{ij} = align(z_i, h_j)$$

where, $align$ is an alignment model that measures the fitness between the i-th word in the target language sentence and the j-th word in the source sentence. **Hard alignment** is used in the conventional alignment model, which means each word in the target language explicitly corresponds to one or more words from the target language sentence. On the other hand, **soft alignment** is used, where any word in source input sentence is related to any word in the target language output sentence, where the strength of the relation or attention is a real number computed via the alignment model. It solely depends on the task we are currently solving whether to use a hard or soft alignment model.

2.5.4 Greedy 1-Best Search

Greedy 1-Best output generation is useful in machine translation, and many other applications where we simply want to output the best according to the model. The simplest way of doing so is greedy 1-best search, in which we simply calculate p_t at every time step, select the word that gives us the highest probability (1-best), and use it as the next word in our sequence (Neubig, 2017). Greedy search is not guaranteed to find the output translation with the highest probability due to local optimum. One possible solution to this problem is to consider n-best words at each time step of the decoder.

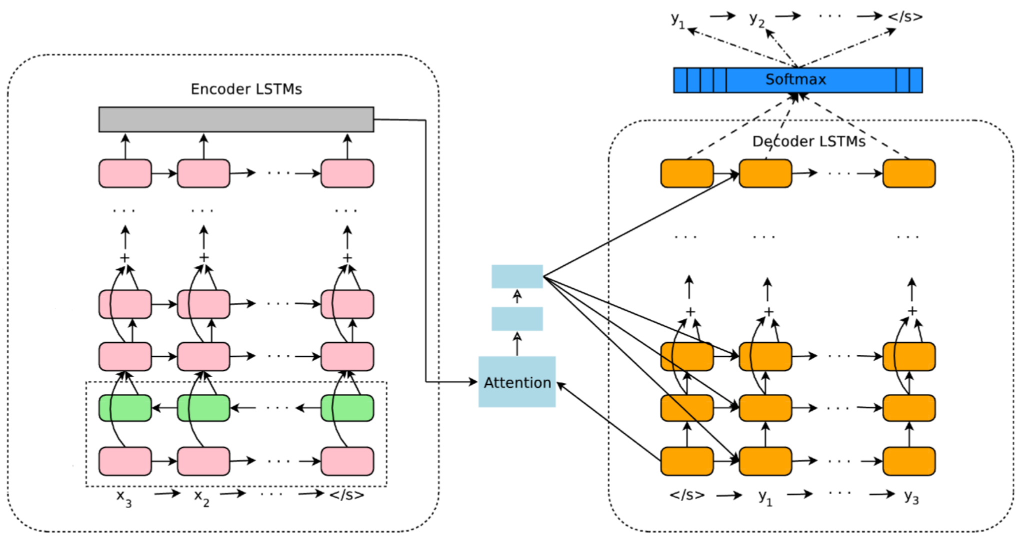
Encoder LSTMs
Attention
Softmax
Decoder LSTMs
x_3 → x_2 → ··· → </s>
</s> → y_1 → ··· → y_3
y_1 → y_2 → ··· → </s>

Figure 2.12: Google's Recent Neural Machine Translation (NMT) Model (Wu et al., 2016)

2.5.5 Beam Search Algorithm

Beam Search[13] is a heuristic search algorithm that explores a graph by expanding the most probable node in a limited set (usually called beam search size). It is often used when the solution space is significantly very large for the applications such as machine translation, speech recognition and natural language generation. It is extremely useful where there is not enough memory available for all the possible solutions to consider.

Beam search builds a search tree using breadth first search algorithm[14] and sorts the nodes according to a heuristic cost (sum of the log probability of the generated words) at each level of the tree. Beam search is similar to greedy search, but instead of considering only the 1-best word, we consider b best words at each time step of the decoder, where b is the width of the beam or sometimes called beam search size. Thus, b best nodes with highest scores are expanded in the next level. This reduces the space and time requirements significantly. However, there is no guarantee of a global optimum solution in case of beam search. In the decoding step, the search process usually stops when the end-of-sentence token $< eos >$ is generated or the maximum length of the sentence is reached.

[13]https://en.wikipedia.org/wiki/Beam_search
[14]https://en.wikipedia.org/wiki/Breadth-first_search

The Figure 2.12 is an example of the Google's recent machine translation framework which uses almost all the techniques described in this chapter.

2.6 Summary

In this chapter, we presented the necessary background information and discussed recent related work in summarization research. As a background information, we believe that a solid understanding of the terms such as summary evaluation, word embedding, Recurrent Neural Network (RNN), Neural Machine Translation (NMT), **seq2seq** encoder decoder framework and beam search decoder is necessary, as our proposed models heavily depend on these concepts. This chapter explains these terms from a computational linguists perspective. From the next chapters, we will start introducing our proposed text summarization models.

Chapter 3

Extractive Coherent Multi-Document Summarization

3.1 Introduction

In this chapter, we aim at developing an extractive summarizer in the multi-document setting. We implement a rank based sentence selection using continuous vector representations along with key-phrases. Furthermore, we propose a model to tackle summary coherence using semantic relations between entities and sentences to increase the readability. We conduct experiments on the Document Understanding Conference (DUC) 2004 datasets using the ROUGE toolkit. Our experiments demonstrate that the methods bring significant improvements over the state of the art methods in terms of information coverage and coherence.

3.2 Sentence Extraction

We here successively describe each of the steps involved in the sentence extraction process such as sentence ranking, sentence clustering, and sentence selection.

3.2.1 Preprocessing

Our system first takes a set of related texts as input and preprocesses them, which includes tokenization, Part-Of-Speech (POS) tagging, Named Entity (NE) extraction, removal of stopwords, filtering punctuation marks and Lemmatization. We use the **NLTK** toolkit[15]

[15]http://www.nltk.org/

to preprocess each sentence to obtain a more accurate representation of the information. Here we describe each of the pre-processing steps in detail.

Tokenization

Tokenization is the process of splitting a stream of text into a list of pieces or tokens. A token can be a word or a sentence. For examples, each word is a token when a sentence is tokenized into words. Each sentence can also be a token, if the sentences are tokenized out of a paragraph.

- **Sentence tokenization** is the process of splitting a paragraph into a list of sentences. For a given input text as following, the output will be:

  Input Text: "The CPP fell short of the two-thirds majority needed to form a government alone. Ranariddh and Sam Rainsy have remained outside the country since the Sept. 24 ceremonial convening of parliament. The agreement will make Hun Sen prime minister and Ranariddh president of the National Assembly."

  Sentence Tokenization Output: ['The CPP fell short of the two-thirds majority needed to form a government alone.' , 'Ranariddh and Sam Rainsy have remained outside the country since the Sept. 24 ceremonial convening of parliament.' , 'The agreement will make Hun Sen prime minister and Ranariddh president of the National Assembly.']

- **Word tokenization** is the process of splitting a sentence into a list of individual words. For a given sentence, the output will be:

  Input Sentence: "Ranariddh and Sam Rainsy have remained outside the country since the Sept. 24 ceremonial convening of parliament."

  Word Tokenization Output: ['Ranariddh' , 'and' , 'Sam' , 'Rainsy' , 'have' , 'remained' , 'outside' , 'the' , 'country' , 'since' , 'the' , 'Sept.' , '24' , 'ceremonial' , 'convening' , 'of' , 'parliament' , '.']

Removing punctuation

We also remove punctuation from the sentence as they do not contribute anything to the meaning of the sentence.

Removal of stopwords

Stopwords are common words that generally do not contribute to the meaning of a sentence. Most search engines will filter stopwords out of search queries and documents in order to save space. NLTK comes with a stopwords corpus that contains 128 word list for English language. In this thesis, we remove stopwords and focus more on the important and topic-related words. We used a stop word list of 571 words to obtain only topic-related important words from the documents which is shown in Appendix A, we name it as smart stop-word list[16].

Input Sentence: "Israel radio quoted officials saying the Cabinet would also decide to renew construction of the Har Homa neighborhood in the traditionally Arab sector of Jerusalem."

Sentence after filtering stopword: ['Israel', 'radio', 'quoted', 'officials', 'Cabinet', 'decide', 'renew', 'construction', 'Har', 'Homa', 'neighborhood', 'traditionally', 'Arab', 'sector', 'Jerusalem', '.']

Lemmatization

Lemmatization is the process of grouping together the different inflected forms of a word to its common base form, which is useful for many text-processing applications. Lemmatization process involves first determining the part of speech of a word, and applying different normalization rules for each part of speech. Words can appear in several inflected forms depending on the context. For example, the verb "talk" may appear as, "talked", "talks", "talking". The base form, "talk", that one might look up in a dictionary,

[16]http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

is called the lemma for the word[17] .

Input Sentence: “Gingrich’s vision seems to have blurred this time around, costing Republicans a net of five seats in Tuesday’s election and leaving the party’s narrow governing majority even narrower.”

Sentence after Lemmatization: [’Gingrich’, “’s”, ’vision’, u’seem’, ’to’, ’have’, u’blur’, ’this’, ’time’, ’around’, ’,’, u’cost’, ’Republicans’, ’a’, ’net’, ’of’, ’five’, u’seat’, ’in’, ’Tuesday’, “’s”, ’election’, ’and’, u’leave’, ’the’, ’party’, “’s”, ’narrow’, u’govern’, ’majority’, ’even’, ’narrower’, ’.’]

Part-Of-Speech (POS) tagging

Part-of-speech Tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech. Each word in a sentence is classified as a Part Of Speech (POS) that depends on the way the word is being used. We used NLTK (Natural Language ToolKit) POS tagger (Bird et al., 2009) which uses the most popular Penn Treebank POS tag set[18]. NLTK converts a list of words into a list of tuples, where each tuple is of the form (word, tag). The tag is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, and so on. An example of part of speech tagging is illustrated below.

Input Text: “The Israeli Cabinet had convened for a second day in an attempt to approve the Wye agreement. Israel radio quoted officials saying the Cabinet would also decide to renew construction of the Har Homa neighborhood in the traditionally Arab sector of Jerusalem. Groundbreaking there in March 1997 led to a break-off in negotiations with the Palestinians.”

Input Text after POS Tagging: [[(’The’, ’DT’), (’Israeli’, ’NNP’), (’Cabinet’, ’NNP’), (’had’, ’VBD’), (’convened’, ’VBN’), (’for’, ’IN’), (’a’, ’DT’), (’second’, ’JJ’), (’day’, ’NN’), (’in’, ’IN’), (’an’, ’DT’), (’attempt’, ’NN’), (’to’, ’TO’), (’approve’, ’VB’), (’the’, ’DT’), (’Wye’, ’NNP’), (’agreement’, ’NN’), (’.’, ’.’)] , [(’Israel’, ’NNP’), (’radio’, ’NN’),

[17] https://en.wikipedia.org/wiki/Lemmatisation

[18] http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html

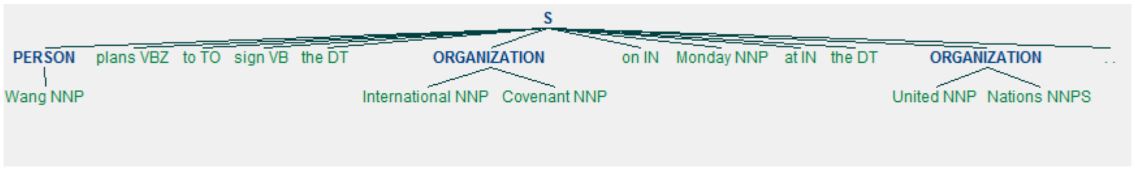

Figure 3.1: Extracted Named Entity (NE) for the example sentence

('quoted', 'VBD'), ('officials', 'NNS'), ('saying', 'VBG'), ('the', 'DT'), ('Cabinet', 'NNP'), ('would', 'MD'), ('also', 'RB'), ('decide', 'VB'), ('to', 'TO'), ('renew', 'VB'), ('construction', 'NN'), ('of', 'IN'), ('the', 'DT'), ('Har', 'NNP'), ('Homa', 'NNP'), ('neighborhood', 'NN'), ('in', 'IN'), ('the', 'DT'), ('traditionally', 'RB'), ('Arab', 'NNP'), ('sector', 'NN'), ('of', 'IN'), ('Jerusalem', 'NNP'), ('.', '.')] , [('Groundbreaking', 'VBG'), ('there', 'RB'), ('in', 'IN'), ('March', 'NNP'), ('1997', 'CD'), ('led', 'VBD'), ('to', 'TO'), ('a', 'DT'), ('break-off', 'NN'), ('in', 'IN'), ('negotiations', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('Palestinians', 'NNPS'), ('.', '.')]]

Named Entity (NE) extraction

Named-entity extraction is a subtask of information extraction from natural language documents, that seeks to identify elements or entities in the text into pre-defined categories such as the names of persons, organizations, locations and expressions of times.[19] (see Figure 3.1 for details)

Input Sentence: Wang plans to sign the International Covenant on Monday at the United Nations.

Extracted NE (Named Entity): (S (PERSON Wang/NNP) plans/VBZ to/TO sign/VB the/DT (ORGANIZATION International/NNP Covenant/NNP) on/IN Monday/NNP at/IN the/DT (ORGANIZATION United/NNP Nations/NNPS) ./.)

[19]https://en.wikipedia.org/wiki/Named-entity_recognition

3.2.2 Sentence Similarity

We take the pre-trained word embeddings[20] (Mikolov et al., 2013b) of all the non stop-words in a sentence and take the weighted vector sum according to the term-frequency (TF) of a word (w) in a sentence (S). We define E as the word embedding model and $idx(w)$ as the index of the word w. More formally, for a given sentence S in the document D, the weighted sum becomes,

$$S = \sum_{w \in S} TF(w,S) \cdot E[idx(w)]$$

Then we calculate the cosine similarity[21] between the sentence vectors obtained from the above equation to find the relative distance between S_i and S_j. We also calculate $NESim(S_i,S_j)$ by finding the Named Entities present in S_i and S_j using the NLTK Toolkit, then calculating their overlap.

$$CosSim(S_i,S_j) = \frac{S_i \cdot S_j}{||S_i||\ ||S_j||}$$

$$NESim(S_i,S_j) = \frac{|NE(S_i) \cap NE(S_j)|}{min(|NE(S_i)|,|NE(S_j)|)}$$

$$Sim(S_i,S_j) = \lambda \cdot NESim(S_i,S_j) + (1-\lambda) \cdot CosSim(S_i,S_j) \tag{3.1}$$

The overall similarity calculation involves both $CosSim(S_i,S_j)$ and $NESim(S_i,S_j)$ where, $0 \leq \lambda \leq 1$ decides the relative contributions of them to the overall similarity computation. This standalone similarity function can be used in this work with different λ values to accomplish different tasks. The main challenge is finding an optimal λ threshold, we here use a hold out dataset to measure the value.

[20]https://code.google.com/archive/p/word2vec/
[21]https://en.wikipedia.org/wiki/Cosine_similarity

Table 3.1: Results in terms precision (P), recall (R), f-measure (F) on SICK dataset for finding optimal threshold λ.

λ	**P**	**R**	**F**
0.1	0.88	0.94	0.91
0.2	0.88	0.94	0.91
0.3	**0.89**	**0.95**	**0.92**
0.4	0.86	0.92	0.89
0.5	0.82	0.88	0.85

To find the optimal threshold λ for the similarity function $Sim(S_i, S_j)$ described earlier, we use the SICK dataset[22] of SemEval-2014. The SICK (Marelli et al., 2014) data set consists of about 10,000 English sentence pairs which is annotated with a relatedness score [1, 5] by means of crowd sourcing techniques. A higher relatedness score indicates the close relatedness between two sentences, 1 (completely unrelated) and 5 (very related). As we are interested in finding related sentences, we filter out the partially related sentences. The pair of sentences in which the relatedness scores is lower than 2 are dissimilar; on the other hand the scores higher than 4 are considered similar. The other sentences in between are filtered out as they are partially related sentences. The remaining dataset consists of 923 dissimilar sentence pairs and 3305 similar sentence pairs. The results in terms precision (P), recall (R) and f-measure (F) on the SICK dataset for finding an optimal threshold λ are presented in Table 3.1. We observe that the performance of the $Sim(S_i, S_j)$ measure is slightly better when $\lambda = 0.3$, that is why we use $\lambda = 0.3$ in our similarity measure.

3.2.3 Sentence Ranking

Next, we rank the sentences by applying the TextRank algorithm (Mihalcea and Tarau, 2004) which involves constructing an undirected graph where sentences are vertices, and weighted edges are formed by connecting sentences by a similarity metric. TextRank determines the similarity based on the lexical overlap between two sentences. However, this algorithm has a serious drawback: If two sentences are talking about the same topic

[22]http://clic.cimec.unitn.it/composes/sick.html

without using any overlapped words, there will be no edge between them. Instead, we use the continuous skip-gram model introduced by (Mikolov et al., 2013b) to measure the semantic similarity along with the entity overlap. We use the similarity function described in Equation (3.1) by setting $\lambda = 0.3$.

After we have our graph, we can run the main TextRank algorithm (Mihalcea and Tarau, 2004) on it. This involves initializing a score of 1 for each vertex, and repeatedly applying the TextRank update rule until convergence. The update rule is:

$$Rank(S_i) = (1-d) + d * \sum_{S_j \in N(S_i)} \frac{Sim(S_i, S_j)}{\sum_{S_k \in N(S_j)} Sim(S_j, S_k)} Rank(S_j) \tag{3.2}$$

Where, $Rank(S_i)$ indicates the importance score assigned to sentence S_i. $N(S_i)$ is the set of neighboring sentences of S_i, and $0 \leq d \leq 1$ is a dampening factor, which the literature suggests its setting to 0.85. After reaching convergence, we extract the sentences along with TextRank scores.

3.2.4 Sentence Clustering

The sentence clustering step allows us to group similar sentences. We use a hierarchical agglomerative clustering (Murtagh and Legendre, 2014) with a complete linkage criteria. This method proceeds incrementally, starting with each sentence considered as a cluster, and merging the pair of similar clusters after each step using bottom up approach. The complete linkage criteria determines the metric used for the merge strategy, which means largest distance between a sentence in one cluster and a sentence in the other candidate cluster are selected for merging. In building the clusters, we use the similarity function described in Equation (3.1) with $\lambda = 0.3$. We set a similarity threshold ($\tau = 0.5$) to stop the clustering process. If we cannot find any cluster pair with a similarity above the threshold ($\tau = 0.5$), the process stops, and the clusters are released. The clusters may be small, but are highly coherent as each sentence they contain must be similar to every other sentence in the same cluster. The whole process is presented in Figure 3.2.

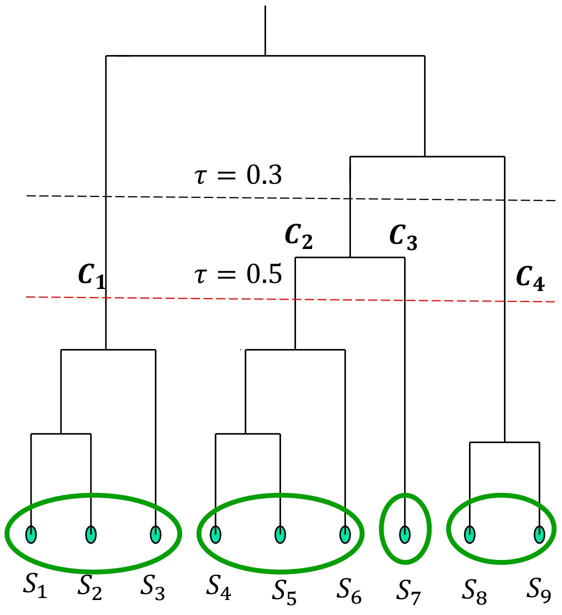

Figure 3.2: Hierarchical Agglomerative Sentence Clustering

This sentence clustering step is very important due to two main reasons: (1) Selecting at most one sentence from each cluster of related sentences will decrease redundancy from the summary side (2) Selecting sentences from the diverse set of clusters will increase the information coverage from the document side as well.

3.2.5 Sentence Selection

In our work, we use the concept-based ILP (Integer Linear Programming)[23] framework introduced in (Gillick and Favre, 2009) with some suitable changes to select the best subset of sentences. This approach aims to extract sentences that cover as many important concepts as possible, while ensuring the summary length is within a given budgeted constraint. Unlike (Gillick and Favre, 2009) which uses bigrams as concepts, we use keyphrases as concepts. Keyphrases are the words or phrases that represent the main topics of a document. Sentences containing the most relevant keyphrases are important for the summary generation. We extracted the keyphrases from the document cluster using RAKE[24] (Rose

[23]https://en.wikipedia.org/wiki/Integer_programming
[24]https://github.com/aneesha/RAKE

et al., 2010). We assign a weight to each keyphrase using the score returned by RAKE.

Let $\overline{w}_i$ be the weight of keyphrase i and k_i a binary variable that indicates the presence of keyphrase i in the extracted sentences. Let l_j be the number of words in sentence j, s_j a binary variable that indicates the presence of sentence j in the extracted sentence set and L the length limit for the set. Let Occ_{ij} indicate the occurrence of keyphrase i in sentence j, the ILP formulation is,

$$Maximize : (\sum_i \overline{w}_i k_i + \sum_j Rank(S_j) \cdot s_j) \tag{3.3}$$

$$Subject\ to : \sum_j l_j s_j \leq L \tag{3.4}$$

$$s_j Occ_{ij} \leq k_i, \quad \forall i, j \tag{3.5}$$

$$\sum_j s_j Occ_{ij} \geq k_i, \quad \forall i \tag{3.6}$$

$$\sum_{j \in g_c} s_j \leq 1, \quad \forall g_c \tag{3.7}$$

$$k_i \in \{0, 1\} \quad \forall i \tag{3.8}$$

$$s_j \in \{0, 1\} \quad \forall j \tag{3.9}$$

We try to maximize the weight of the keyphrases (3.3) in the extracted sentences, while avoiding repetition of those keyphrases (3.5, 3.6) and staying under the maximum number of words allowed for the sentence extraction (3.4).

In addition to the ILP formulation of (Gillick and Favre, 2009), we put some extra features such as maximizing the sentence rank scores returned from the sentence ranking section in our ILP objective function. In order to ensure only one sentence per cluster in the extracted sentences we add an extra constraint (3.7), this will ensure non-redundancy from the summary side. In this process, we extract the optimal combination of sentences that maximize information coverage while minimizing redundancy (Figure 3.3 illustrates our sentence extraction process in brief).

3.3 Sentence Ordering

One crucial step in generating a coherent summary is to order the sentences in a logical manner to increase the readability. A wrong order of sentences convey an entirely different idea to the reader of the summary and also make it difficult to understand. In a single document, summary information can be presented by preserving the sentence position in the original document. In multi-document summarization, the sentence position in the original document does not provide any clues to the sentence arrangement. Hence it is a very challenging task to perform the arrangement of sentences in the summary. Classic reordering approaches include inferring order from a weighted sentence graph (Barzilay et al., 2002), or perform a chronological ordering algorithm (Cohen et al., 1999) that sorts sentences based on timestamp and position.

We here propose a simple greedy approach to sentence ordering in a multi-document setting. Our assumption is that a good sentence order implies the similarity between all adjacent sentences since word repetition (more specifically, named entity repetition) is one of the formal signs of text coherence (Barzilay et al., 2002). We define coherence of document D which consists of sentences from S_1 to S_n in the following equation. For calculating $Sim(S_i\ ,S_{i+1})$, we use the similarity function described in equation (3.1) with $\lambda = 0.5$, giving the named entities a little more preference.

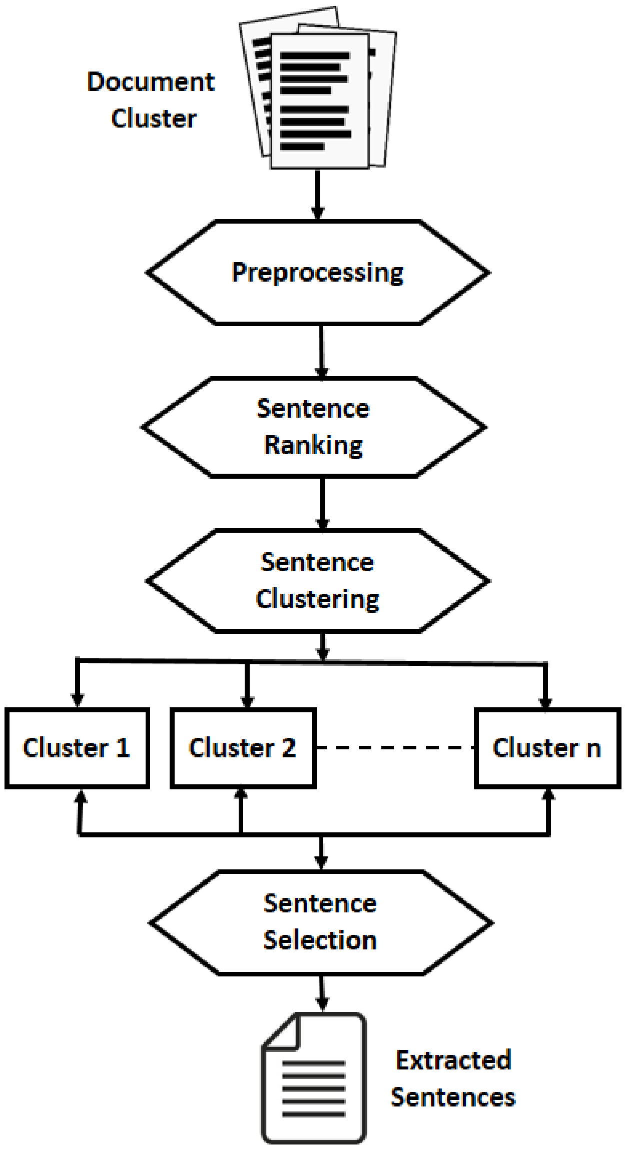

Figure 3.3: Sentence Extraction Process

$$Coherence(D) = \frac{\sum_{i=1}^{n-1} Sim(S_i, S_{i+1})}{n-1}$$

We propose a greedy algorithm for placing a sentence in a document based on the coherence score we discussed above[25]. At the beginning, we randomly select a sentence from the extracted sentences without any position information and place the sentence in the ordered set D. We then incrementally add each extracted sentences to the document set D using Algorithm below to get the final order of summary sentences.

Algorithm 1: Place a sentence in a document

```
Procedure SentencePositioning($D, S_n$)
    Data: Input document $D$ which is assumed sorted. New sentence $S_n$ which we will place
          in the document $D$.
    Result: Return new document $D_n$ after placing the sentence $S_n$.
    $t \leftarrow 1$;
    $Coh_{max} \leftarrow 0$ ;
    $D_{tmp} \leftarrow D$ ;
    $l \leftarrow DocLength(D)$ ;
    while $t \leq l+1$ do
        $\Rightarrow$Place the $S_n$ in $t^{th}$ position of $D_{tmp}$ ;
        $Coh_{tmp} \leftarrow Coherence(D_{tmp})$;
        if $Coh_{tmp} > Coh_{max}$ then
            $D_n \leftarrow D_{tmp}$;
            $Coh_{max} \leftarrow Coh_{tmp}$;
            $\Rightarrow$ Remove $S_n$ from the $t^{th}$ position of the document $D_{tmp}$ ;
        end
        $t \leftarrow t+1$;
    end
    return $D_n$;
```

3.4 Evaluation

We evaluate our system **ILPRankSumm** (**ILP** based sentence selection with Text**Rank** for Extractive **Summ**arization) using **ROUGE**[26] (Lin, 2004) on the DUC 2004 document set (Task-2, Length limit (L) = 100 words). However, ROUGE scores are biased towards

[25]Note that, we didn't take any position information of the original sentences to be extracted from the document.

[26]ROUGE-1.5.5 with options: -n 2 -m -u -c 95 -x -r 1000 -f A -p 0.5 -t 0

lexical overlap at the surface level and insensitive to summary coherence. Moreover, sophisticated coherence evaluation metrics are seldom adopted for summarization, and therefore many of the previous systems used human evaluation for measuring readability. For this reason, we evaluate our summary coherence using (Lapata and Barzilay, 2005; Barzilay and Lapata, 2008) which defines coherence probabilities for an ordered set of sentences.

3.4.1 Baseline Systems

We compare our system with the following baseline (LexRank, GreedyKL) and state-of-the-art systems (Submodular, ICSISumm). **LexRank** (Erkan and Radev, 2004) represents input texts as a graph where nodes are the sentences and the edges are formed between two sentences if the cosine similarity is above a certain threshold. Sentence importance is calculated by running the PageRank algorithm on the graph. **GreedyKL** (Haghighi and Vanderwende, 2009) iteratively selects the next sentence for the summary that will minimize the KL divergence between the estimated word distributions. Lin and Bilmes (2011) treat the document summarization problem as maximizing a **Submodular** function under a budget constraint. They achieved a near-optimal information coverage and non-redundancy using a modified greedy algorithm (Carbonell and Goldstein, 1998). On the other hand, **ICSISumm** (Gillick and Favre, 2009) employs a global linear optimization framework, finding the globally optimal summary rather than choosing sentences according to their importance in a greedy fashion.

The summaries generated by the baselines and the state-of-the-art extractive summarizers on the DUC 2004 dataset were collected from (Hong et al., 2014).

3.4.2 Results

Our results include R-1, R-2, and R-SU4, which counts matches in unigrams, bigrams, and skip-bigrams respectively. The skip-bigrams allow four words in between. According to Table 3.2, R-1, R-2 scores obtained by our system outperform all the baseline and state-of-the-art systems on the DUC 2004 datasets. One of the main reasons for getting the

Table 3.2: Results on DUC 2004 (Task-2) for the baseline, state-of-the-art and our proposed system **ILPRankSumm**.

System	Models	R-1	R-2	R-SU4	Coherence
Baseline	LexRank	35.95	7.47	12.48	0.39
	GreedyKL	37.98	8.53	13.25	0.46
State-of-the-art	Submodular	39.18	9.35	**14.22**	0.51
	ICSISumm	38.41	9.78	13.31	0.44
Proposed System	ILPRankSumm	**39.45**	**10.12**	14.09	**0.68**

Table 3.3: System's output (100 words) for the document set **d30015t** from DUC 2004.

Summary Generated (After Sentence Extraction)
But U.S. special envoy Richard Holbrooke said the situation in the southern Serbian province was as bad now as two weeks ago. A Western diplomat said up to 120 Yugoslav army armored vehicles, including tanks, have been pulled out. On Sunday, Milosevic met with Russian Foreign Minister Igor Ivanov and Defense Minister Igor Sergeyev, Serbian President Milan Milutinovic and Yugoslavia's top defense officials. To avoid such an attack, Yugoslavia must end the hostilities, withdraw army and security forces, take urgent measures to overcome the humanitarian crisis, ensure that refugees can return home and take part in peace talks, he said.
Summary Generated (After Sentence Ordering)
On Sunday, Milosevic met with Russian Foreign Minister Igor Ivanov and Defense Minister Igor Sergeyev, Serbian President Milan Milutinovic and Yugoslavia's top defense officials. But U.S. special envoy Richard Holbrooke said the situation in the southern Serbian province was as bad now as two weeks ago. A Western diplomat said up to 120 Yugoslav army armored vehicles, including tanks, have been pulled out. To avoid such an attack, Yugoslavia must end the hostilities, withdraw army and security forces, take urgent measures to overcome the humanitarian crisis, ensure that refugees can return home and take part in peace talks, he said.

improved R-1 and R-2 scores is the use of keyphrases. Moreover, there is no significant difference between our proposed system and submodular in the case of R-SU4. We also obtain better coherence probability because of our sentence ordering technique. Our system's output for a randomly selected document set (e.g. d30015t) from DUC 2004 is shown in Table 3.3 (for more examples see Appendix B)

3.4.3 Limitations

One of the essential properties of the text summarization systems is the ability to generate a coherent summary with a fixed length (DUC 2004, Task-2: Length limit = 100 words). According to (Hong et al., 2014) all the summarizers from the previous research either truncated the summary at the 100^{th} word, or removed the last sentence from the summary set. In our work, we follow the second option to produce a grammatically correct summary. However, the former produces a certain ungrammatical sentence, while the later can lose a lot of information in the worst case, if the sentences are long. We focus more on the grammaticality of the final summary.

3.5 Summary

In this chapter, we implemented an ILP based sentence selection model along with TextRank scores and key phrases for extractive multi-document summarization. Our extractive summarizer can jointly maintain information coverage from the document side and non-redundancy from the summary side. We further model the coherence to increase the readability of the generated summary using a greedy algorithm which can reorder the extracted sentences. Evaluation results strongly indicate the benefits of using continuous word vector representations in all the steps involved in the overall system. In the next chapter, we will focus more on the abstractive text summarization where the extracted sentences are transformed using sentence fusion and lexical substitution.

Chapter 4

Abstractive Coherent Multi-Document Summarization

4.1 Introduction

Extractive summarization systems select the salient (important) sentences from the source document without any modification by copy and paste method. On the other hand, abstractive summarization methods rewrite the sentences to create a summary. The abstractive techniques which are traditionally used are sentence compression, syntactic reorganization and lexical paraphrasing. However, in the case of multi-document summarization where source documents usually contain similar information, the extractive methods would produce a redundant summary or are biased towards a specific source document(s).

Multi-sentence compression (MSC) can be a useful solution for the above problem. It usually takes a group of related sentences about the same topic and produces an output sentence through merging the sentences while retaining the most important information and still maintain the grammaticality of the generated sentence. MSC originally called sentence fusion (Barzilay and McKeown, 2005) is a text-to-text generation process in which a novel sentence is produced as a result of summarizing a set of similar sentences. On the other hand, Lexical paraphrasing aims at replacing some selected words with other similar words while preserving the meaning of the original text. A good lexical substitution for a target word needs to be semantically similar to the target word and compatible with the given context (Melamud et al., 2015). For example, the sentence "Jack composed these verses in 1995" could be lexically paraphrased into "Jack wrote these lines in 1995" without altering the sense of the initial sentence.

This chapter presents a first attempt towards finding an abstractive compression generation system for a set of related sentences which jointly models sentence fusion and paraphrasing using continuous vector representations. Our paraphrastic fusion system attempts to improve information coverage and grammaticality of generated sentences. Our system can be applied to various real world applications such as text simplification, microblog, opinion and newswire summarization. We conduct experiments on a human-generated multi-sentence compression dataset and evaluate our system using several newly proposed Machine Translation (MT) evaluation metrics. Our experiments demonstrate that our method brings significant improvements over the state-of-the-art systems across different metrics.

4.2 Sentence Abstraction : An Overview

Most of the previous works rely only on one of the following techniques for abstracting sentences (Clarke and Lapata, 2006, 2008; Filippova, 2010; Boudin and Morin, 2013; Filippova and Strube, 2008). Instead, in this thesis we take the first step towards finding a joint representation for sentence abstraction using sentence fusion and lexical paraphrase rather than treating these two independently.

1. **Sentence Compression**

    - [*ACD*~~**EFA**~~*GED*] $\Rightarrow$ [*ACDGED*]
    - Deletion of unimportant words from the input sentence.
    - Mainly used for summarizing a sentence or headline generation.
    - Sentence compression example:

      Input Sentence: “Reporter Jennifer Griffin has been on the road today , heading south from Beirut , and she joins us by phone from Tyre .”

Compressed Sentence: "Reporter Jennifer Griffin , heading south from Beirut , joins us by phone from Tyre ."

2. **Sentence Fusion**

   - [*ACD*~~**EFA**~~*GED*] + [~~**CDEF**~~*BADE*] ⇒ [*ACDGEDBADE*]
   - Involves the merging of two or more sentences into one.
   - Reduces redundancy in the final generated summary.
   - Sentence fusion example:

     Input Sentences: Obama told NBC "I'm frustrated with myself" for unintentionally sending a message that there are "two sets of rules" for paying taxes, "one for prominent people and one for ordinary folks."
     "We can't send a message to the American people that we have got two sets of rules – one for prominent people and one for ordinary people," Obama said, defending his administration's standards.

     Fused Sentence: Obama told NBC "I'm frustrated with myself" for unintentionally sending a message to the American people that we have got two sets of rules for paying taxes, one for prominent people and one for ordinary folks.

3. **Syntactic Reorganization**

   - [*ACDEF***AGED**] ⇒ [**AGED***ACDEF*]
   - Helps to make sentence coherent and paraphrase.
   - Example of syntactic reorganization:
     Input Sentence: The cleaning crew vacuums and dusts the office every night.
     Reorganized Sentence: Every night the office is vacuumed and dusted by the cleaning crew.

4. **Lexical Paraphrase**

   - $[A\cancel{CD}EFA\cancel{GE}DHB] \Rightarrow [A\mathbf{BG}EFA\mathbf{BC}DHB]$
   - Replaces complex words with simple words to make the sentence easier to understand.
   - Example of lexical paraphrasing:

     Input Sentence: In fact, not many people do think female troops should be confined to desk jobs .

     Paraphrased Sentence: In fact , not many people do think female troops should be restricted to desk jobs .

4.3 Paraphrastic Sentence Fusion Model

4.3.1 Word Graph Construction for Sentence Fusion

In order to generate a one sentence representation from a cluster of related sentences we use the word-graph approach of (Filippova, 2010). Let $S = \{s_1, s_2, ..., s_n\}$ be a set of related sentences, we construct a graph $G = (V, E)$ by iteratively adding sentences to it. The vertices are the words along with the parts-of-speech (POS) tags, and directed edges are formed by simply connecting the adjacent words in the sentences. Once the first sentence is added, words from the other related sentences are mapped onto a node in the graph provided that they have exactly the same lower-case word form and the same POS tag. Each sentence is connected to dummy start and end nodes to mark the beginning and ending of the sentences. Words are added to the graph in the following order:

- non-stopwords are added for which no candidate exists in the existing graph or for which an unambiguous mapping is possible;
- non-stopwords are added for which there are either several possible candidates, multiple mappings are possible in the graph or which occur more than once in the sentence;

- stopwords.

For the last two cases where mapping is ambiguous (i.e. there are two or more nodes in the graph that refer to the same word and same POS tag), the immediate context (the preceding and following words in the sentence and the neighboring nodes in the graph) are used to select the candidate node for forming the edge (Boudin and Morin, 2013). In (Filippova, 2010), punctuation marks are not considered. To generate well-punctuated compressions which in turn represents complete sentences, following (Boudin and Morin, 2013) we considered a fourth step for adding punctuation marks in the graph. Moreover, similarly as (Boudin and Morin, 2013) we also use the stopword list included in NLTK[27] extended with temporal nouns such as “yesterday”, “Friday”, and etc. Figure 4.1 illustrates an example word-graph for the following two sentences,

S1: The video was made on Feb. 19-20, 2003.

S2: The morning after the video was made, she said, three social workers came and interviewed them.

As we can see, the two input sentences contain similar information, but differ in sentence length, syntax, and the detail of information. The solid directed arrows connect the words in the first sentence S1, while the dotted arrows join the words in the second sentence S2. After constructing the word-graph using (Filippova, 2010; Boudin and Morin, 2013) described above, we can generate the K-shortest paths from the dummy start node to the end node in the word graph (see figure 4.1). For example, we can generate these paths:

Ex1: The morning after the video was made on feb. 19-20 , 2003.

Ex2: The video was made , she said , three social workers came and interviewed them.

Ex3: The morning after the video was made , she said , three social workers came and interviewed them.

Consider another example for better understanding of the constructed word graph from the following sentences,

[27] http://www.nltk.org/

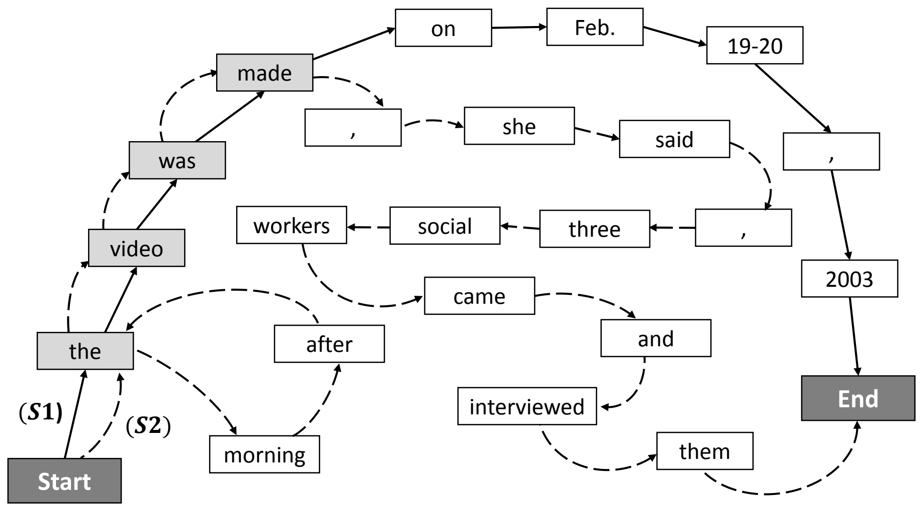

Figure 4.1: Constructed Word graph and a possible compression path (light gray nodes)

S1: In Asia Japan Nikkei lost 9.6% while Hong Kongs Hang Seng index fell 8.3%.

S2: Elsewhere in Asia Hong Kongs Hang Seng index fell 8.3% to 12,618.

From the word graph which is shown in figure 4.2, we can generate these following paths:

Ex1: In Asia Hong Kongs Hang Seng index fell 8.3%.

Ex2: Elsewhere in Asia Hong Kongs Hang Seng index fell 8.3%.

Ex3: Elsewhere in Asia Japan Nikkei lost 9.6% while Hong Kongs Hang Seng index fell 8.3%.

The above examples are sampled from the *K*-shortest paths generated from the word graph *G* (*K* is usually ranges from 50 to 200 according to the literature (Filippova, 2010; Boudin and Morin, 2013). The main challenge of the sentence fusion is to generate sentences that are grammatically correct and contain the most important information. Hence, we design a candidate ranking strategy to sort the generated *K*-shortest paths based on grammaticality and informativeness.

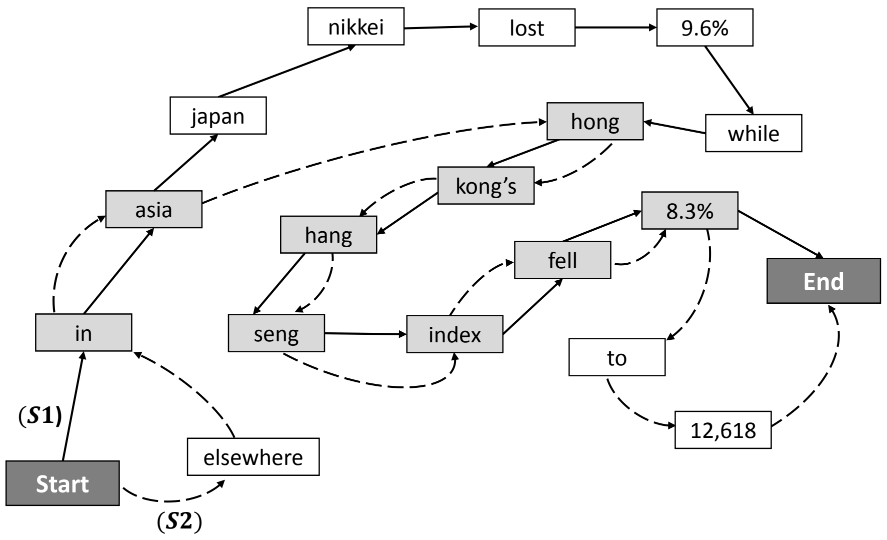

Figure 4.2: Constructed Word graph (2) and a possible compression path (light gray nodes)

4.3.2 Candidate Ranking

We rank the fused candidates by applying the sentence ranking algorithm described in Chapter 3 (section 3.2.3). We extract the fused candidate sentences along with CandidateRank scores $Rank(C_i)$.

4.3.3 Grammatical Quality

We compute grammatical quality of a fused sentence candidate using a 3-gram (trigram) language model similarly to (Banerjee et al., 2015), which assigns probabilities to sequence of words in a generated candidate. Suppose that a candidate contains a sequence of m words $\{w_1, w_2, w_3, \ldots, w_m\}$. The score GQ (Gramatical Quality) assigned to each candidate is defined as follows:

$$GQ(w_1, ..., w_m) = \frac{1}{1 - Score_{LM}(w_1, ..., w_m)} \tag{4.1}$$

$$Score_{LM}(w_1, ..., w_m) = \frac{log_2 \prod_{t=3}^{m} P(w|w_{t-1}w_{t-2})}{N} \tag{4.2}$$

The scores are normalized by N, the word length of the candidates. The Language model scores are negative. Therefore, in Equation (4.1), we take the reciprocal of the logarithmic value with smoothing to compute $GQ(w_1, ..., w_m)$. In our experiments, we used a 3-gram model that is trained on the English Gigaword corpus[28]. We use the implementation[29] provided by (Buck et al., 2014) for scoring the n-grams.

Finally, we rank the K candidate fusions and find the N-best sentence fusion which balances the grammaticality and the informativeness. The score of a candidate sentence fusion c is given by the following linear combination between the candidate rank score and the grammaticality score (where, we set $\alpha = 0.6$)

$$score(c) = \alpha \cdot Rank(c) + (1-\alpha) \cdot GQ(c) \tag{4.3}$$

4.3.4 Context Sensitive Lexical Substitution

Target Word Identification for Substitution: We first label the words in all N-best candidates using Part-Of-Speech (POS) tagging. We then filter out the named entities where $NE \in \{PER; LOC; ORG; MISC\}$. We take only the nouns and verbs for possible substitution candidates.

Substitution Selection: The PPDB 2.0[30](Pavlick et al., 2015) provides millions of lexical, phrasal and syntactic paraphrases which come in packages of different sizes (going from S to XXXL)[31]. For instance, we can gather lexical substitution set S = {*gliding, sailing, diving, travelling*} for the target word ($t = flying$) from PPDB 2.0. We hardcoded the model to select substitutes with the same POS tag and that are not a morphological variant (*such as fly, flew, flown*).

[28]Available : http://www.keithv.com/software/giga/ (We used the 64K NVP vocabulary version)
[29]https://github.com/kpu/kenlm
[30]http://paraphrase.org/
[31]For our experiment, we use the XXL lexical one

Substitution Ranking: Word embeddings are low-dimensional vector representations of words such as word2vec (Mikolov et al., 2013b) that recently gained much attention in various semantic tasks. Word2vecf (Levy and Goldberg, 2014) is an extension of word2vec to produce syntax-based word embeddings. They show that these embeddings tend to capture functional word similarity (as in *manage* $\rightarrow$ *supervise*) rather than topical similarity (as in *manage* $\rightarrow$ *manager*). We use the word and context vectors released by (Melamud et al., 2015) which were shown to perform strongly on the lexical substitution task. These embeddings contain 600d (600 dimension) vectors for 173k words and about 1M syntactic contexts processed using the dependency based word2vecf model (Levy and Goldberg, 2014). Their measure *addCos* for estimating the appropriateness of a substitute s from the substitution set S, for the target word t in the set of the target word's context elements $C = \{c_1, c_2, ..., c_n\}$, is defined as follows,

$$addCos(s|t,C) = \frac{cos(s,t) + \sum_{c \in C} cos(s,c)}{|C| + 1}$$

Finally, we select the best substitution **s** according to maximum **addCos** scores over $\mathbf{0.7}$ and replace it with the target word **t**.

4.4 Experimental Setup

In this section, we present our experimental setup for assessing the performance of the paraphrastic fusion model described above. We give details on the datasets we used, evaluation metrics, and the baseline systems used for comparison with our approach.

Our system first takes a set of related texts as input and preprocesses them using the same way as described in chapter 3, section 3.2.1. We generate 50 shortest paths from start to end nodes from each cluster in the graph using the K-shortest path algorithm (Boudin and Morin, 2013). Paths shorter than eight words or that do not contain a verb are filtered. To ensure pure abstractive compression generation, we remove the paths that have *cosineSimilarity* ≥ 0.9 to any of the original sentence in the cluster. We then select the

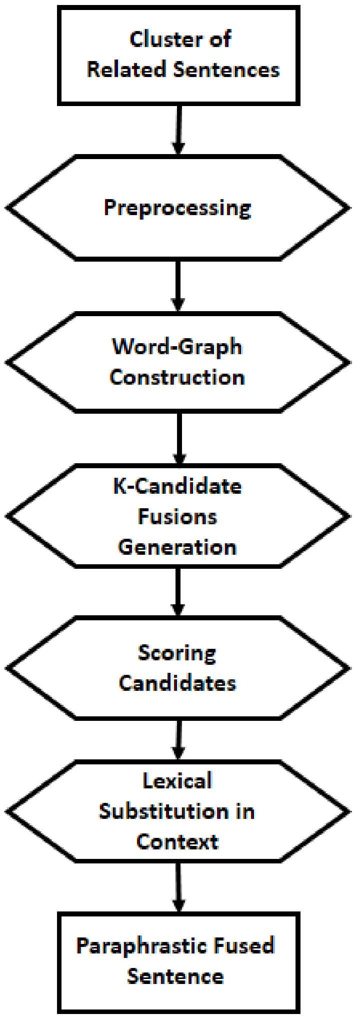
Cluster of Related Sentences
Preprocessing
Word-Graph Construction
K-Candidate Fusions Generation
Scoring Candidates
Lexical Substitution in Context
Paraphrastic Fused Sentence

Figure 4.3: Paraphrastic Sentence Fusion Model

3-best candidates from the K paths for lexical substitution. The whole process is presented in the figure 4.3. For fair evaluation, we also select the 3-best candidates for the baseline systems that we compare with our model.

4.4.1 Dataset

We conducted experiments on the human generated sentence fusion dataset released by (McKeown et al., 2010). This dataset consists of 300 English sentence pairs taken from newswire clusters accompanied by human-produced sentence fusions rewrites collected via the Amazon Mechanical Turk service[32]. We filtered the sentences which have no main verbs. The resulting set contains 296 pairs of sentences.

4.4.2 Evaluation Metric

We evaluate our system automatically using various automatic metrics. We also introduce some new automatic evaluation metrics.

BLEU (Papineni et al., 2002) is the most commonly used metric for Machine Translation evaluation. BLEU relies on the exact matching of *n*-grams and has no concept of synonymy or paraphrasing. We used the implementation provided in NLTK[33] considering up to 4-gram matching.

SARI (Xu et al., 2016) is a recently proposed metric which compares **S**ystem output **A**gainst **R**eferences and against the **I**nput sentence. SARI computes the arithmetic average of n-gram precision and recall of three rewrite operations: addition, copying, and deletion which correlates well with human references. One caveat with using SARI as a reward is the fact that it relies on the availability of multiple references.

METEOR-E (Denkowski and Lavie, 2014) uses a combination of both precision and recall in the METEOR metric. Furthermore, the alignment is based on exact token matching, followed by WordNet synonyms, stemmed tokens and then look-up table paraphrases. Recently, an augmented version of METEOR using distributed representations named **METEOR-E** (Servan et al., 2016) has been released[34].

Compression Ratio is a measure of how terse a compression is and is given in the following equation. A compression ratio of zero implies that the source sentence is fully uncompressed.

$$Compression\ Ratio\ (CR) = \frac{\#tok_{del}}{\#tok_{orig}}$$

[32] http://www.mturk.com

[33] https://github.com/nltk/nltk/tree/develop/nltk/translate

[34] https://github.com/cservan/METEOR-E

Table 4.1: Comparison with baselines and our **Paraphrastic Fusion** model across different automatic evaluation metrics (the scores are averaged)

Model	**BLEU**	**SARI**	**METEOR-E**	**Compression Ratio**	**Copy Rate**	**Gramaticality(%)**
Filippova (2010)	40.6	34.6	0.31	**0.57**	99.8	58.2%
Boudin and Morin (2013)	**44.0**	37.2	0.36	0.42	99.9	65.8%
Banerjee (2015)	42.3	36.5	0.34	0.45	99.8	71.4%
Paraphrastic Fusion	42.5	**37.4**	**0.43**	0.41	**76.2**	**73.5%**

Table 4.2: The output generated by the baseline and our Paraphrastic Fusion system (the paraphrased words are marked bold)

Input Sentences	Bush, who initially nominated Roberts to replace retiring Justice Sandra Day O'Connor, tapped him to lead the court the day after Rehnquist's death. President Bush initially nominated Roberts in July to succeed retiring Justice Sandra Day O'Connor.
Filippova (2010)	president bush initially nominated roberts to replace retiring justice sandra day o'connor .
Boudin and Morin (2013)	bush , who initially nominated roberts in july to succeed retiring justice sandra day o'connor , tapped him to lead the court the day after rehnquist 's death .
Banerjee et al. (2015)	bush , who initially nominated roberts to replace retiring justice sandra day o'connor , tapped him to lead the court the day after rehnquist 's death .
Paraphrastic Fusion	president bush initially **recommended** roberts in july to **accomplish** retiring justice sandra day o'connor , tapped him to **run** the court the day after rehnquist 's death .

Copy Rate: We define the copy rate as how many tokens are copied to the abstract sentence from the source sentence without paraphrasing in the following equation. A lower copy rate score means more paraphrasing is involved in the abstract sentence. A copy rate of 100% means no paraphrasing is involved in the process.

$$Copy\ Rate = \frac{|S_{orig} \cap S_{abs}|}{|S_{abs}|}$$

Grammaticality: We define grammaticality as the parsing problem, if the sentence is successfully parsed, then it has valid grammar; if not, then it does not. We did not use any statistical parser because the parser will still return a parse for a sentence with bad grammar as it uses the statistics to make the best guess possible. Instead, we use a chart parser to parse a sentence, given a CFG (Context-Free Grammar) which is implemented in the NLTK Toolkit.

4.4.3 Baseline Systems and Results

We compare our system with (Filippova, 2010), (Boudin and Morin, 2013)[35] and (Banerjee et al., 2015)[36]. Table 4.2 shows the output generated by the baseline and our system. We report our system's performance compared with the baselines in terms of different evaluation metrics in Table 4.1. We get a slightly higher score in SARI because of the multiple human abstractive rewrites along with the input sentence. The copy Rate score of other baseline systems clearly indicates the fact that they are performing complete compression, no paraphrasing is involved. Moreover, we also get higher score in METEOR-E metric because of the lexical substitution operation. In our experiments, we used a LM (Language Model) which tends to choose longer sentences. Therefore, we get a lower compression ratio than Filippova (2010). However, we achieve a higher grammaticality percentage. As expected, we get a slightly lower BLEU score compared to (Boudin and Morin, 2013) for two main reasons: (1) We tried to balance between gammaticality and information coverage (2) BLEU works well on surface level lexical overlap.

4.5 Document Level Abstractive Summarization

In this section, we will apply our sentence level paraphrastic fusion model to generate document level abstractive summarization.

- Our system first takes a set of related documents as input according to the same topic and preprocesses them (see section 3.2.1 for details).
- We cluster all the sentences in the document using the sentence clustering technique proposed in Chapter 3 (section 3.2.4).
- For each cluster of related sentences, we generate the 5-best abstractive fused sentences using our paraphrastic fusion model described in section 4.3. However, for the

[35]https://github.com/boudinfl/takahe
[36]https://github.com/StevenLOL/AbTextSumm

clusters containing only one sentence, we use our context sensitive lexical substitution model to generate just the abstractive version of the source sentence.

- We use the concept-based ILP framework described in chapter 3 (section 3.2.5) to select the best subset of sentences under a certain limit (L = 100 Words). In comparison with the ILP formulation described in section 3.2.5, we excluded the sentence ranking section from the objective function as our paraphrastic fusion model generates ranked sentences. For a clear understating of the document level abstractive summarization model, the whole process is presented in figure 4.4.
- Finally, we order the extracted sentences using our greedy sentence ordering technique described in chapter 3 (section 3.3)

4.6 Experimental Setup

4.6.1 Dataset

We consider the generic multi-document summarization dataset provided at the Document Understanding Conference (DUC 2004)[37] which is one of the main benchmark dataset in the multi-document summarization field. It contains 50 document clusters and each is composed of 10 news wire articles about a given topic from the Associated Press and The New York Times that are published between 1998 to 2000. The dataset also contains multiple human-written summaries which are used for the evaluation of system-generated summaries.

4.6.2 Evaluation Metric

We evaluate our summarization system using **ROUGE**[38] (Lin, 2004) on DUC 2004 (Task-2, Length limit (L) = 100 words). However, ROUGE scores are unfairly biased towards lexical overlap at surface level. Taking this into account, we also evaluate our sys-

[37]http://duc.nist.gov/duc2004/

[38]ROUGE-1.5.5 with options: -n 2 -m -u -c 95 -x -r 1000 -f A -p 0.5 -t 0

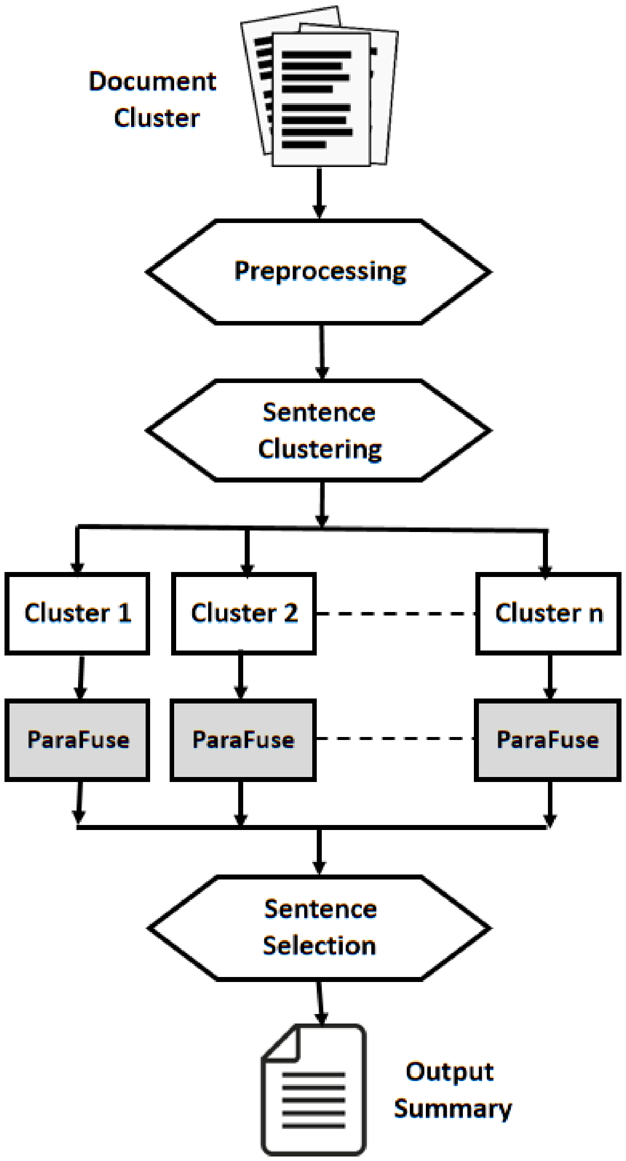

Figure 4.4: Our document level Paraphrastic Fusion model

Table 4.3: Results on DUC 2004 (Task-2) for the baseline, state-of-the-art and our proposed abstractive summarization system (**ParaFuse_doc**).

System	Models	R-1	R-2	R-WE-1	R-WE-2	Coherence
Baseline	LexRank	35.95	7.47	-	-	0.39
	GreedyKL	37.98	8.53	-	-	0.46
State-of-the-art	Submodular	39.18	9.35	-	-	0.51
	ILPSumm	39.24	11.99	40.31	12.40	0.59
Proposed System	ParaFuse_doc	**40.13**	**12.08**	**42.73**	**13.02**	**0.70**

tem with recently proposed metric **ROUGE-WE** (Ng and Abrecht, 2015), which considers word embeddings to compute the semantic similarity of the words. Moreover, both metrics are insensitive to summary coherence. For this reason, we evaluate our summary coherence using (Lapata and Barzilay, 2005; Barzilay and Lapata, 2008).

4.6.3 Baseline Systems

We compare our system with baseline (**LexRank** (Erkan and Radev, 2004), **GreedyKL** (Haghighi and Vanderwende, 2009)) and state-of-the-art systems (**Submodular** (Lin and Bilmes, 2011), **ILPSumm** (Banerjee et al., 2015)). ILPSumm is a pure abstractive summarization technique for a multi-document setting. For fair comparison, we use the author provided implementation[39] to generate a summary from their model. The summaries generated by the other baselines and the state-of-the-art extractive summarizers on the DUC 2004 dataset were collected from (Hong et al., 2014).

4.7 Results & Discussion

According to Table 4.3, the R-1, R-2, R-WE-1, RWE-2 scores obtained by our system outperform all the baselines and state-of-the-art systems on DUC 2004 dataset. Other than ILPSumm all the other systems are purely extractive, so we did not report the performance on R-WE of those systems. We also get better performance on the coherence probability score because of our sentence ordering technique. Some system outputs generated by our

[39]https://github.com/StevenLOL/AbTextSumm

abstractive summarization system are shown in Appendix C.

4.8 Summary

In this chapter, we designed a new abstractive compression generation model which jointly performs sentence fusion and paraphrasing using a skip-gram word embedding model. Furthermore, we also applied our model to a abstractive multi-document summarization system where documents usually contain related set of sentences and achieved state-of-the art results. In the next chapter, we will design another abstractive compression generation model. The difference with the model presented in the next chapter is that, we will jointly model sentence compression (other than sentence fusion) and paraphrasing using a standard **seq2seq** encoder decoder model.

Chapter 5

Neural Paraphrastic Sentence Compression Generation

5.1 Introduction

In this chapter, we will design a paraphrastic compression model which jointly performs compression and paraphrase in a single sentence in order to produce an abstractive compression. We will use a standard **seq2seq** encoder decoder model in order to accomplish this task. According to the state of the art, this is the first neural model to tackle compression and paraphrase in a single sentence.

5.2 Overview of the Model

In this section we present ParaComp, our neural **Para**phrastic **Comp**ression model based on Neural Machine Translation (NMT). **ParaComp** uses neural machine translation to translate from a source sentence to an abstractive compression. In the following, we briefly overview the basic encoder-decoder NMT framework and then discuss how it can be extended to our task.

We formalize the task of paraphrastic compression generation as follows. Given a source sentence $\mathbf{X} = (x_1, x_2, ..., x_N)$, our model learns to predict its paraphrastic compression target $\mathbf{Y} = (y_1, y_2, ..., y_M)$, where $M < N$. Inferring the target Y given the source X is a typical sequence to sequence learning problem, which can be modeled with attention-based encoder-decoder models (Bahdanau et al., 2015; Luong et al., 2015). Our model is mainly inspired from (Luong et al., 2015) with some specific changes to accomplish the task.

As the name suggests, the basic form of an encoder-decoder model consists of two components: (a) an encoder which computes a representation for a source sentence X and (b) a decoder which generates one target word at a time which is conditioned on all previously generated words $y_{1:t-1}$ like the following,

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{M} P(y_t|y_{1:t-1}, \mathbf{X})$$

5.3 Encoder

The encoder in our case is a bi-directional GRU (Bi-GRU) unlike (Luong et al., 2015) which uses uni-directional LSTM. The GRU (Cho et al., 2014a) achieves similar performance as LSTM but it is fast to train and can improve performance on long sequences. In the simplest uni-directional case, while reading input symbols from left to right, a GRU learns the hidden annotations h_t at time t with

$$h_t = \mathbf{GRU}(h_{t-1}, e(x_t)) \tag{5.1}$$

where, the $h_t \in \mathrm{I\!R}^n$ encodes all content seen so far at time t which is computed from h_{t-1} and $e(x_t)$, where $e(x_t) \in \mathrm{I\!R}^m$ is the m-dimensional embedding of the current word x_t.

The t^{th} unit is fed with previous output h_{t-1} and current input x_t, and produces its output h_t. When calculating h_t, it uses update gate u_t and reset gate r_t to improve the performance on long sequences. The forward propagation of GRU is computed as follows.

$$\mathbf{h_t} = (\mathbf{1} - \mathbf{u_t}) \odot \mathbf{h_{t-1}} + \mathbf{u_t} \odot \hat{\mathbf{h_t}}$$

$$\hat{\mathbf{h_t}} = tanh\ (\mathbf{W}e(x_t) + \mathbf{U}(\mathbf{r_t} \odot \mathbf{h_{i-t}}))$$

$$\mathbf{r_t} = \sigma\ (\mathbf{W_r}e(x_t) + \mathbf{U_r}\mathbf{h_{t-1}}))$$

$$\mathbf{u_t} = \sigma\left(\mathbf{W_u}e(x_t) + \mathbf{U_u}\mathbf{h_{t-1}}\right))$$

where, W_u , W_r, $W \in \mathbb{R}^{n\times m}$ and U_u , U_r, $U \in \mathbb{R}^{n\times n}$ are weight matrices, n is the number of hidden units, $\sigma()$ is the sigmoid function, and $\odot$ is the element-wise multiplication.

Conventional RNNs typically deal with a text sequence from start to end, and build the hidden state of each word only by considering its preceding words. The hidden state should also consider its following words as well. Hence, we apply a bidirectional RNN (Bi-RNN) (Graves et al., 2013) to learn hidden states using both the preceding and the following words. In our work, we actually applied bi-directional GRUs (bi-GRUs), which we found achieves better results than single directional GRUs consistently.

As shown in Figure 5.1, Bi-GRU processes the input document in both the forward direction and backward direction with two separate hidden layers calculated with GRUs, to obtain the forward hidden states ($\overrightarrow{h_1}$, ... , $\overrightarrow{h_N}$) and the backward hidden states ($\overleftarrow{h_1}$, ... , $\overleftarrow{h_N}$). For each position t, we simply concatenate its both forward and backward states into the final hidden state:

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$$

in which operator $\oplus$ indicates concatenation. $\overrightarrow{h_t}$ is calculated following Eq. (5.1) and $\overleftarrow{h_t}$ is calculated using following equation.

$$\overleftarrow{h_t} = \mathbf{GRU}(\overleftarrow{h_{t+1}}, e(x_t))$$

One other important modification we can do to the GRUs following (Luong et al., 2015) is stacking multiple layers on top of each other (stacked GRUs Figure 5.1). For example, in a 3-layer stacked GRU, the calculation at time step t would look as follows:

$$h_{1,t} = \mathbf{BiGRU_1}(e(x_t), h_{1,t-1})$$

$$h_{2,t} = \mathbf{BiGRU_2}(h_{1,t}, h_{2,t-1})$$

$$h_{3,t} = \mathbf{BiGRU_3}(h_{2,t}, h_{3,t-1})$$

They can extract more abstract features of the current words or sentences. For instance, (Shi et al., 2016) proved that in a two-layer stacked LSTM, the first layer tends to learn granular features of words such as part of speech tags, while the second layer learns more abstract features of the sentence such as voice or tense. However, stacking RNNs suffer from the vanishing gradient problem in the vertical direction from the output layer ($\mathbf{GRU_3}$) to the layer close to the input ($\mathbf{GRU_1}$), just as the standard RNN did in the horizontal direction. This causes the earlier layers of the network to be under-trained. A simple solution to this problem is to add residual connections, which has been shown to be extremely useful for the image recognition task recently (He et al., 2016). The idea behind these networks is simply to add the output of the previous layer directly to the result of the next layer. For example, in a 3-layer stacked GRU with residual connections, the calculation at time step t would look as follows:

$$h_{1,t} = \mathbf{BiGRU_1}(e(x_t), h_{1,t-1}) + e(x_t)$$

$$h_{2,t} = \mathbf{BiGRU_2}(h_{1,t}, h_{2,t-1}) + h_{1,t}$$

$$h_{3,t} = \mathbf{BiGRU_3}(h_{2,t}, h_{3,t-1}) + h_{2,t}$$

Therefore, we use the idea of residual networks in building our encoder decoder model **ParaComp** for the abstractive compression task which is illustrated in Figure 5.2. The initial hidden states of the encoder are set to zero vectors, i.e., $\overrightarrow{h^S_{1,1}} = 0$, $\overleftarrow{h^S_{1,N}} = 0$. In our **ParaComp** model, the encoder transforms the source sentence $\mathbf{X}$ into a sequence of hidden states $(\mathbf{h}^S_{3,1}, \mathbf{h}^S_{3,2}, \ldots, \mathbf{h}^S_{3,N})$ with a stacked residual network.

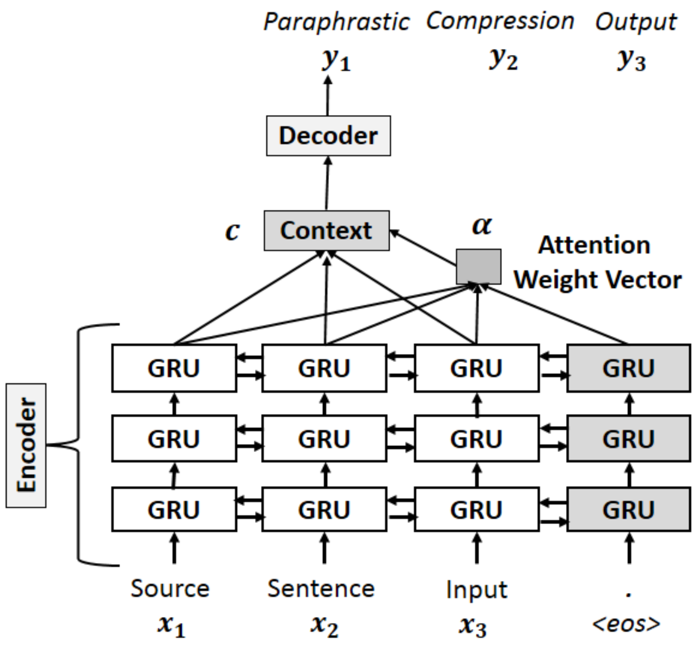

Figure 5.1: Neural Paraphrastic Compression Generation Model (Stacked GRUs)

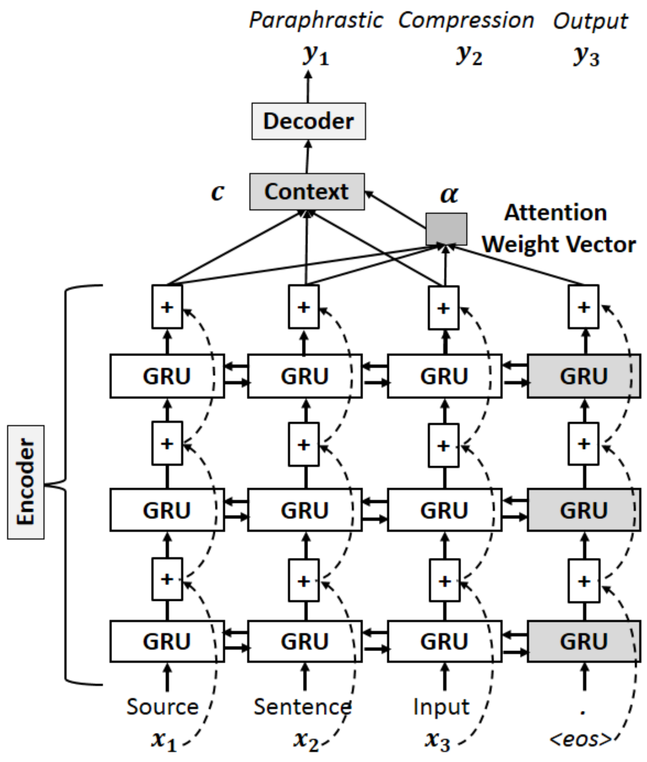

Figure 5.2: Neural Paraphrastic Compression Generation Model (Stacked GRUs with Residual Connections)

5.4 Decoder and Attender

The decoder uses a simple GRU with attention to generate one word y_{t+1} at a time in the paraphrastic compression target sentence $\mathbf{Y}$. Abstractive sentence generation is conditioned on all previously generated words $y_{1:t}$ and a context vector $\mathbf{c_t}$, which encodes the source sentence:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{M} P(y_t|y_{1:t-1}, \mathbf{X}) \tag{5.2}$$

$$P(y_{t+1}|y_{1:t}, \mathbf{X}) = softmax\ (g(\mathbf{h}_t^T, \mathbf{c}_t)) \tag{5.3}$$

where $g(\cdot)$ is a one-hidden-layer neural network with the following parameters:

$$g(\mathbf{h}_t^T, c_t) = \mathbf{W}_o\ tanh(\mathbf{U}_h \mathbf{h}_t^T + \mathbf{W}_h \mathbf{c}_t) \tag{5.4}$$

where $\mathbf{W}_o \in \mathbb{R}^{|V|\times n}$, $\mathbf{U}_h \in \mathbb{R}^{n\times n}$, and $\mathbf{W}_h \in \mathbb{R}^{n\times n}$; $|V|$ is the vocabulary size and n is the hidden unit size. $\mathbf{h}_t^T$ is the hidden state of the decoder **GRU** which summarizes what has been generated so far $y_{1:t}$. At the very beginning of the decoding, we initialize the GRU hidden state $\mathbf{h}_0^T$ with the last backward encoder hidden state $\overleftarrow{h_{3,1}^S}$ as input from our proposed stack residual network.

$$\mathbf{h}_t^T = \mathbf{GRU}(e(y_t), \mathbf{h}_{t-1}^T) \tag{5.5}$$

During each time step in the decoder, instead of using a fixed context (i.e. last hidden state of encoder), a distinct context vector $\mathbf{c_t}$ is used for generating word y_{t+1}. This context vector $\mathbf{c_t}$ is basically the weighted sum of the stacked residual GRU hidden states ($\mathbf{h}_{3,i}^S$) of the encoder. The weight $\alpha_{\mathbf{ti}}$ denotes the strength of attention of the t^{th} word in the target language sentence to the i^{th} word in the source sentence. The basic idea behind the attention vector is that it is telling us how much we are focusing on a particular source word at a particular time step of the decoder. The larger the value in $\alpha_{\mathbf{ti}}$, the more impact a word will have when predicting the next word in the output sentence.

$$\mathbf{c}_t = \sum_{i=1}^{N} \alpha_{ti} \mathbf{h}^S_{3,i} \tag{5.6}$$

$$\alpha_{\mathbf{ti}} = \frac{\exp(\mathbf{h}_t^T \cdot \mathbf{h}^S_{3,i})}{\sum_i \exp(\mathbf{h}_t^T \cdot \mathbf{h}^S_{3,i})} \tag{5.7}$$

We use the $(\cdot)$ *dot* attention mechanism of (Luong et al., 2015) due to its efficiency and which is simple to implement. The dot attention mechanism is actually the dot product between two hidden vectors.

5.5 Training and Decoding

The training objective of our **ParaComp** model is to maximize the log likelihood of the sentence-abstract pairs in a given training set D (where, θ is the training parameters):

$$L_t(\theta) = \sum_{(x,y)\in D} log\ p(\mathbf{Y}|\mathbf{X};\theta)$$

$$p(\mathbf{Y}|\mathbf{X};\theta) = \prod_{t=1} P(y_t|y_{1:t-1}, \mathbf{X})$$

Once the models are trained, we use the following modified beam search algorithm to find the output that maximizes the conditional probability.

Algorithm 2: Restricted Beam Search Decoding Algorithm

```
Data: Vocabulary size $|V|$, beam size $B$, max output length $N$.
Result: Return $K$ paraphrastic compression variations of a source sentence.
$\Rightarrow$ Computed probabilities of all the words in vocabulary
$\Rightarrow$ Choose the $B$ most likely words and initialize the $B$ hypotheses
while $t \leq N$ do
    $\Rightarrow$ For each hypothesis, compute the next conditional probabilities, then have
      $B \times |V|$ candidates with the corresponding probabilities
    $\Rightarrow$ Use the $[AttentionScore]$ to choose B most likely candidates those are not in
      the $V_{history}[(t-3):(t-1)]$
end
```

5.6 Repetition control using restricted beam search decoding

Repetition is a common problem in the **seq2seq** encoder decoder model (Tu et al., 2016; Sankaran et al., 2016) and our model is no exception. The following examples are picked by manually investigating our model's generated output.

Output#1: It is not appropriate appropriate to insist upon a Syrian withdrawal.

Output#2: Lebanese parliamentary sessions parliamentary session have to be open to the public.

Output#3: This has been ruled has been out .

The reduction of redundant repeating generation of tokens for neural abstractive summarization first tackled by (Suzuki and Nagata, 2017). They jointly estimate the upper-bound frequency of each target vocabulary in the encoder and control the output words based on the estimation in the decoder. Recently, (See et al., 2017) adapted the coverage model of (Tu et al., 2016) to solve this problem in their work for generating a multi-sentence summary from a single document. See et al. (2017) maintain a coverage vector, which is the sum of attention distributions over all previous decoder time steps.

In our work, we follow a simple approach in order to solve this problem. Our main goal is to reduce the complexity in the decoder. We keep track of all the previously generated tokens at the t^{th} time step of the decoder in a separate variable called $V_{history}(t)$ for each beam. While generating the t^{th} word our model looks into the $V_{history}(t-1)$ for immediate uni-gram repetition, $V_{history}(t-2)$ for bi-gram repetition and $V_{history}(t-3)$ for possible tri-gram repetition. We hard-code the decoder not to choose these words (or any morphological variation of these words) which can or may cause repetition.

5.7 Dealing with out of vocabulary problem

At each generation step of the decoder, the output word is selected according to the probability distribution over the whole target vocabulary in the softmax layer, which is the most time and capacity-consuming part of the system. Therefore, most summary generation systems keep a fix-sized target vocabulary according to the word frequency. The infrequent words were removed from the vocabulary and were replaced with the symbol $< UNK >$, meaning unknown word. However, it has been observed that the infrequent words are usually proper nouns or named-entities that have an impact on the meaning of the overall sentence. Therefore, (Gu et al., 2016) introduced COPYNET which is an encoder-decoder architecture equipped with a copying mechanism. COPYNET can integrate the regular way of word generation in the decoder with the new copying mechanism which can choose words or subsequences in the input sequence and put them at appropriate places in the output sequence. Gulcehre et al. (2016) also propose to solve the unknown word related problem in an end-to-end neural network. When predicting an output word, the model first makes a decision whether to pick a word from target vocabulary or copy from source input, which was later adopted by (Nallapati et al., 2016) for abstractive sentence summarization.

In our work, we follow an easier approach to tackle the “UNK replace” problem. From our training samples, we acquired almost 300K unique words in the vocabulary. We limit our vocabulary to include the top 50K most frequent words, while the other words are replaced with the token $< UNK >$. By doing this, we are explicitly predicting in which contexts and places the model will be expecting an unknown word (they are usually proper nouns or named-entities). We use 100 $< UNK >$ placeholders to represent out of vocabulary $< OOV >$ words. The embeddings for these unknown words are less interchangeable in the encoder. The placeholders are working as a queue and taken from the model vocabulary’s last 100 places. The main advantage is the probability of seeing $< UNK >$ during training is evenly split out into 100 placeholders. During generation we copy the unknown words from the input sentence to the placeholders according to their position of appearance in the source sentence (e.g. from left to right in the source sentence, the first unknown word is copied to the first placeholder and so on). We let the model decide according to the probability distribution over the whole target vocabulary including the placeholders.

5.8 Paraphrasing in Context

We trained our model on paraphrase, abstractive compression and text simplification sentence pairs. Our model implicitly learned how to paraphrase and can eventually generate paraphrases from the data itself. Moreover, to ensure complete paraphrasing we also impose some explicit edit operations.

5.8.1 Pre-Edit Paraphrasing

We use the 50K most frequent words, as our model vocabulary out of almost 300K unique words from the training set. We create an alignment table for the words outside the vocabulary to the words inside the vocabulary using **GloVe** embedding (Pennington et al., 2014). The word-to-word alignment has been done by calculating the cosine distance between glove average word vectors. We found an alignment table of almost 8K words outside of the vocabulary to words inside with $CosDistance \geq 0.7$ (e.g. pricey $\Rightarrow$ expensive, detested $\Rightarrow$ hated). Our model tries to replace the out of vocabulary words with the words inside the vocabulary using the alignment table during training as well as generation.

5.8.2 Post-Edit Paraphrasing

We apply post-edit paraphrasing operations, after the K paraphrastic compression variations generated by the model for a single sentence. We use the context sensitive lexical substitution described in Chapter 4 (section 4.3.4) to accomplish post-edit paraphrasing. We collect a substitution set S for a particular target word t from the model described in section 4.3.4. To generate paraphrastic compression variations, we select different words for the different output sentence from the substitution set S.

5.9 Compression Variation Selection

The model generates K different variations from the decoding step due to the beam search size ($B = 10$). We filter out the candidates that are too close to each other using string edit distance[40]. Finally, we rank the candidates according to the following scoring function,

$$Rank_c(C_k) = \delta \cdot \frac{\#tok_{del}(C_k)}{\#tok_{orig}(S)} + (1-\delta) \cdot BeamSearchScore(C_k)$$

where, S is the original sentence, C_k is a candidate paraphrastic compression, $tok_{del}(C_k)$ is the number of tokens deleted in the candidate K. We set $\delta = 0.6$ (optimally tuned based on the validation data) to rank the shorter candidates higher. One typical output of our model is shown in Table 5.1.

[40]https://pypi.python.org/pypi/python-Levenshtein

Table 5.1: Our model's output for the source sentence. (**CR** means Compression Ratio, *italic* means paraphrasing in context.)

Source Sentence	It is the right message, sent while it is still early enough to do something constructive about the disappointing quality of the work so far.
Reference(*Best*)	It is the right message to send to correct the disappointing quality of work so far. **(CR: 0.36)**
Output#1	*this* message is the right message. **(CR: 0.76)**
Output#2	it is the right message, sent while it is still early enough to do something *suitable*. **(CR: 0.44)**
Output#3	it is the right message, sent while it is still early enough to do something *faster* about the work. **(CR: 0.24)**
Output#4	*this* message is the right message, sent while it is still early enough to do something *useful* about the work so far. **(CR: 0.12)**
Output#5	it is the right message, sent while it is still early enough to do something *faster* about the work so far. **(CR: 0.16)**

5.10 Experimental Setup

In this section, we present our experimental setup for assessing the performance of the paraphrastic compression model described above. We give details on the datasets we used, model training, evaluation metrics, and the baseline and state-of-the-art systems used for comparison with our approach.

5.10.1 Dataset

Deep Neural Network (DNN) architectures are completely data driven. More training data will produce a good quality output sequence. The DUC[41] (Document Understanding Conference) and the TAC[42] (Text Analysis Conference) datasets are not sufficient for training a full end-to-end neural network. Therefore, almost all the past works on abstractive summarization using neural networks (Rush et al., 2015; Filippova et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Suzuki and Nagata, 2017; Zhou et al., 2017) made use of the English Gigaword dataset (Napoles et al., 2012). They take the first sentence of a news

[41] http://www-nlpir.nist.gov/projects/duc/data.html
[42] https://tac.nist.gov/data/

document, align it with the headline of that document and generated 3.8M source-summary training pairs. It was argued that the first sentence is enough to capture the gist of a document, but its not perfect in all cases. They assume that the first sentence of a document as a source sentence and headline as a summary sentence. However, headlines are not expected to be grammatical and complete, therefore in the generation step their model produces ungrammatical sentences. In our work, instead of generating training pairs using headlines we use the existing human annotated datasets which is either used for compression, paraphrase or abstractive compression. All the sentence-summary pairs used for our model training are purely grammatical, complete and human-generated.

Deletion-based compression corpora: Clarke and Lapata (2006, 2008) provided two manually annotated corpora, which are named written news (WN) and broadcast news (BN) respectively[43]. The Written news corpus contains 1,629 gold compressions pairs. Moreover, the broadcast news corpus contains 1,370 pairs with three different trained human annotations, therefore we have 4,110 unique compression sentence pairs from the BN corpus. Filippova et al. (2015) created 2M sentence pairs using (Filippova and Altun, 2013) by aligning news headlines to first sentences. Only 10K parallel sentences has been publicly released[44], as they are not human annotated we will not use this dataset for training.

Paraphrase corpora: The Microsoft Research paraphrase corpus (MSRP)[45] (Dolan et al., 2004) contains 3,900 human labeled true paraphrase pairs from both the training and test set. The Plagiarism Detection Corpus (PAN)[46] is constructed by deriving aligned corresponding sentences from 41,233 plagiarised documents made available by (Madnani et al., 2012). PAN consists of 13,000 sentence pairs in total, among them 6,500 are true paraphrase pairs. The SICK (Marelli et al., 2014) data set consists of about 10,000 English sentence pairs[47]. Each sentence pair is annotated with a relatedness score [1, 5] by means

[43]http://jamesclarke.net/research/resources
[44]http://storage.googleapis.com/sentencecomp/compression-data.json
[45]https://www.microsoft.com/en-us/download/details.aspx?id=52398
[46]http://bit.ly/mt-para
[47]http://clic.cimec.unitn.it/composes/sick.html

of crowd sourcing techniques, with higher scores indicating the two sentences are more closely-related. We extracted 3,672 sentence pairs with a relatedness score [4,5].

Abstractive compression corpora: Recently, Kajiwara and Komachi (2016) built a monolingual parallel corpus for text simplification by aligning sentences from English Wikipedia (normal) and Simple English Wikipedia (simple) using various sentence similarity measures which use word embeddings. The released version[48] consists of 492,493 sentence pairs, which we can consider as abstractive compression. Toutanova et al. (2016) also recently introduced a manually-created, multi-reference dataset for abstractive sentence and short paragraph compression. It contains approximately 6,000 source texts with multiple compressions (about 26,000 pairs of source and compressed texts) which is accompanied by up to five crowd-sourced rewrites. They divided the whole dataset into training,valid and test sets. We filtered out the compressions which involves multiple source sentences. From their traing set we obtained almost 12,050 unique pairs for our training. We are not aware of any published results on this two recently released datasets.

Text Simplification corpora[49]: (Xu et al., 2015) introduced NewSela[50] corpus consisting of 1,130 news articles, each re-written four times for children at different reading or grade levels by professional editors. It has a total of 10,787 documents, each with a unique article identifier and a version indicator between 0 and 4 (which defines the level of simplicity), where 0 refers to the original form, and 4 to its simplest version. We use the sentence alignment algorithm by (Xu et al., 2015) for producing 141,582 complex-simple sentence pairs.

In total, we collected 665,936 human-generated training pairs for our model. As we are doing paraphrastic compression (In general, abstractive compression), we took the validation and test sets of (Toutanova et al., 2016) as ours. The validation and test set contains 271 and 459 pairs of single sentence abstractive compression with maximum of five human

[48]https://github.com/tmu-nlp/sscorpus
[49]https://newsela.com/data/
[50]Newsela (https://newsela.com), a company that produces reading materials for pre-college classroom use.

rewrite variations. The basic information of the datasets are presented in Table 5.2.

Table 5.2: Statistics of the datasets

Datasets	**# of Pairs**	**Source Length**	**Target Length**	**# of Vocab**
Compression	5,739	22.17	15.81	9.8K
Paraphrase	14,072	22.74	21.91	31.8K
Abstractive Compression	504,543	25.38	18.00	267K
Text Simplification	141,582	25.68	16.97	40K

5.10.2 Training Details

We trained our model on an **Nvidia TITAN X GPU** card with 12G RAM. We use 300-dimensional uncased pre-trained GloVe embeddings[51] (Pennington et al., 2014) to initialize word embedding matrices of our model. We use a reverse training sequence (Sutskever et al., 2014) which is a trick that avoids long-distance dependencies in RNN. The assumptions is that the input sequence and the output sequence usually has similar word orders. We use **Adam** (Kingma and Ba, 2014) to optimize parameters with a mini-batch of size 80 and randomly shuffled the training data at every epoch. We followed scheduled sampling (Bengio et al., 2015) that dynamically adjusts the balance between target feeding and self generation. We limit our vocabularies to be the top 50K most frequent words. Our stacked residual BiGRU model have 3 layers, with each GRU containing 512 cells. We did not use any dropout. We used early stopping based on the validation set and used the best model on the validation set. The beam size of the decoder was set to be 10.

5.10.3 Evaluation Metric

We evaluate our system automatically using various automatic metrics described in Chapter 4 (section 4.4.2) such as BLEU, SARI, METEOR-E, compression ratio and copy rate.

[51] http://nlp.stanford.edu/data/glove.6B.zip

5.10.4 Baseline Systems

We compare our model with the systems which includes both deletion-based and abstractive models. **ILP**, an integer linear programing approach for sentence compression which involves word deletion (Clarke and Lapata, 2008); **T3**, a tree-to-tree transduction model for abstractive sentence compression (Cohn and Lapata, 2009); **seq2seq**, a neural model for deletion-based compression (Filippova et al., 2015); and **NAMAS**, a neural model for abstractive compression and sentence summarization (Rush et al., 2015).

Table 5.3: Performance of different systems compare to our proposed **ParaComp_sent** (A sentence level **Para**phrastic **Comp**ression) model

Model	BLEU	SARI	METEOR-E	Compression Ratio	Copy Rate	Gramaticality(%)
T3	11.1	25.7	0.22	**0.75**	90.6	57.2%
ILP	**54.7**	38.1	0.35	0.29	99.5	59.8%
seq2seq	53.8	35.5	0.31	0.39	99.7	56.4%
NAMAS	38.7	36.6	0.32	0.24	99.8	49.3%
ParaComp_sent	49.2	**39.3**	**0.41**	0.47	**71.3**	**74.3%**

5.10.5 Performance Comparison & Discussion

The output generated by the above mentioned systems were collected from (Toutanova et al., 2016). As we used the identical test set, we use the output directly to compare our system with the metrics discussed earlier. For fair comparison, we added all the top ($N = 5$) candidates in the evaluation process. The results of different systems across the different metrics are presented in Table 5.3. As our model is generating paraphrastic compression, we obtain a lower BLEU score compare to ILP, as BLEU works well on surface level lexical overlap. We get a slightly higher score in SARI because of the multiple human references. ThecCopy Rate scores of the baseline systems other than T3 clearly indicates that they are doing complete compression, and no paraphrasing is involved. We also get a higher score in METEOR-E metric because of the lexical substitution operation. Moreover, compare to the baseline our model has the benefit of using human level sentence-summary pairs in terms of gramaticality score.

5.11 Summary

In this chapter, we have designed another abstractive compression generation model which jointly models sentence compression and paraphrasing using a standard **seq2seq** encoder decoder model. We propose simple yet effective solutions to several common problems in neural **seq2seq** models such as redundant repetition and unknown token replacement. In the next chapter, we will apply our sentence level **Para**phrastic **Comp**ression model to multi-document level abstractive summarization.

Chapter 6

Neural Abstractive Multi-Document Summarization

6.1 Introduction

In this chapter, we will propose a neural paraphrastic compression model at the document level. To the best of our knowledge, this is the first model to generate a multi-sentence abstractive summary using a **seq2seq** encoder decoder model in a multi-document setting. The difference between the multi-document abstractive summarization model described in Chapter 4 (section 4.5) and the model proposed here is the first one uses paraphrastic sentence fusion and this one uses the neural paraphrastic sentence compression model presented in chapter 5. Furthermore, we also introduce a new concept called "**Reader Aware Summary**" which can generate summaries for some critical readers (non-native readers). Finally, we designed an optimal solution for the classical summary length limit problem in a multi-document setting.

6.2 Document Level Neural Paraphrastic Compression Model

In this section, we apply our sentence level neural paraphrastic compression model to the abstractive multi-document summarization.

- We use our sentence extraction technique proposed in Chapter 3 (section 3.2) to extract the important and no-redundant sentences from each related document set.

- We put a longer length limit ($L = 200$ words) in our ILP formulation for sentence extraction, as our paraphrastic compression model will further compress the extracted sentences.

- We then order the sentences using our greedy sentence ordering algorithm proposed in Chapter 3 (section 3.3).

- We take the ordered set of sentences and incrementally produce abstraction variations using our neural paraphrastic compression model (presented in chapter 5) one by one for each sentence. For each extracted sentence, we generate 5-best paraphrastic compressions using our **ParaComp** model ($K = 5$) described in Chapter 5. We compute the grammatical quality of a generated paraphrastic compression sentence candidate using the Equation 4.1 (presented in chapter 4, section 4.3.3).

- Finally, we use the following ILP formulation to select the best subset of paraphrastic compressions for each extracted sentences.

$$Maximize : \sum_i (GQ(s_i) + Sim(s_{ext_i},\ s_i) + \frac{l_i}{\hat{L}}) \cdot s_i \tag{6.1}$$

$$Subject\ to : \sum_i l_i s_i \leq \hat{L} \tag{6.2}$$

$$\sum_{i \in g_K} s_i \leq 1, \quad \forall g_K \tag{6.3}$$

$$GQ(s_i) \geq \alpha \tag{6.4}$$

$$Sim(s_{ext_i},\ s_i) \geq \beta \tag{6.5}$$

$$Sim(s_{ext_i},\ s_i) \leq \gamma \tag{6.6}$$

$$s_i \in \{0,1\} \quad \forall j \tag{6.7}$$

where, $GQ(s_i)$ measures the grammatical quality of paraphrastic compressions, and $Sim(s_{ext_i},\ s_i)$ measures the semantic similarity between the sentence extracted (s_{ext_i}) and the abstraction of the sentence (we use the sentence similarity measure described in Chapter 3, section 3.2.2). Moreover, $Sim(s_{ext_i},\ s_i)$ defines the margin between near extractive to full abstractive summary. In order to ensure only one paraphrastic compression variation is included in the final summary, we add an extra constraint (6.3). $\hat{L}$ is the length budget, which is set to 100 words. We also introduce three summary quality parameters namely α , β and γ ; α ensures grammaticality of the abstracted sentence; β and γ presents a window between near extractive to complete abstractive sentence selection. By the term, near extractive means the generated sentences using our neural paraphrastic compression model are more closer to the extracted sentences. To ensure complete abstractive summarization, we put a upper bound γ in the sentence selection. Moreover, we do not want a sentence to be included in the summary which is not semantically similar to the original extracted sentence, that's why a lower bound β was introduced to restrict the dissimilar sentences.

6.3 Optimal Summary Budget

One of the essential properties of the text summarization systems is the ability to generate a summary with a fixed length (DUC 2004, Task-2 (Multi-Document): Length limit = 100 words). According to (Hong et al., 2014) all of the multi-document summarizer from the previous research either truncated the summary at the 100^{th} word, or removed the last sentence from the summary set. However, the first option produces a certain ungrammatical sentence, the later one can lose a lot of information in the worst case, if the sentence is long. Recently, (Kikuchi et al., 2016) propose four methods in order to tackle this issue. Two of them are based on different decoding procedures without model architecture modification. The other two are learning-based, (i.e., the models take the desired length information as input and encode it into the model architecture). However, their model is limited to the headline generation task (DUC 2004, Task-1) where models generate a single sentence headline of a document. In this thesis, we tackle this issue in a multi-document setting by generating multiple paraphrastic compression length variations of a sentence (see table 5.1 for the examples). In our ILP formulation for the document level summary generation, we are trying to maximize the total summary length to optimally solve the length limit problem. Under any circumstances, our model can choose a shorter variation of a sentence automatically to be included in the summary (Appendix D presents some system outputs of our model).

6.4 Experimental Setup

6.4.1 Dataset

We use the DUC 2004[52] dataset to evaluate our summarization system. The DUC 2004 dataset (Task-2) is made of 50 sets of documents, each consisting of 10 newswire articles related to a topic. The task (we are considering Task-2) consists of generating a summary of a maximum of 100 words for each document set. Four human-authored reference summaries are provided for each document set, and can be used for automatic evaluation. We randomly select 15 sets of documents as our validation data for tuning the parameters α , β and γ. We set the parameters ($\alpha = 0.12$, $\beta = 0.4$ and $\gamma = 0.8$) based on the validation data for optimal performance. The remaining 35 document sets are used for the final evaluation.

[52]http://duc.nist.gov/duc2004/

Table 6.1: Results on DUC 2004 (Task-2) for the baseline, state-of-the-art and our system.

System	**Models**	**R-1**	**R-2**	**R-WE-1**	**R-WE-2**	**Coherence**
Baseline	LexRank	35.95	7.47	-	-	0.39
	GreedyKL	37.98	8.53	-	-	0.46
State-of-the-art	Submodular	39.18	9.35	-	-	0.51
	ILPSumm	39.24	11.99	40.31	12.40	0.59
Proposed System	ParaComp_doc	**40.06**	**12.01**	**42.41**	**12.73**	**0.71**

6.4.2 Evaluation Metric

We evaluate our system using **ROUGE**[53] (Lin, 2004). As ROUGE scores are unfairly biased towards lexical overlap, we further evaluate our system with recently proposed metric **ROUGE-WE** (Ng and Abrecht, 2015), which considers word embeddings to compute the semantic similarity of the words. Moreover, both metrics are insensitive to summary coherence. For this reason, we evaluate our summary coherence using (Lapata and Barzilay, 2005; Barzilay and Lapata, 2008).

6.4.3 Baseline Systems

We compare our system with two baseline systems (**LexRank** (Erkan and Radev, 2004), **GreedyKL** (Haghighi and Vanderwende, 2009)) and two state-of-the-art systems (**Submodular** (Lin and Bilmes, 2011), **ILPSumm** (Banerjee et al., 2015)). **ILPSumm** (Banerjee et al., 2015) is a recently proposed abstractive summarizer which relies on multi-sentence compression and ILP to perform abstractive summarization in a multi-document setting. For fair comparison, we use the author provided implementation[54] to generate a summary from their model. The summaries generated by the other baseline and the state-of-the-art extractive summarizers on the DUC 2004 dataset were collected from (Hong et al., 2014).

[53]ROUGE-1.5.5 with options: -n 2 -m -u -c 95 -x -r 1000 -f A -p 0.5 -t 0

[54]https://github.com/StevenLOL/AbTextSumm

6.4.4 Performance Comparison & Discussion

According to Table 6.1, the **R-1**, **R-WE-1**, **RWE-2** scores obtained by our system outperform all of the baseline and state-of-the-art systems on the DUC 2004 dataset[55]. We also obtain better performance on coherence score because of our sentence ordering technique.

6.5 Reader Level Summary Generation

In this thesis for the first time, we introduced the concept of "**Reader Level Summary**", which means the output of the summarization system depends largely on the reader of the summary. The readers of a summary can be classified according to the **demographic information** (e.g. age, gender, educational background, income and cultural background), **cognitive aspects** (e.g. prior experience, technical skills, memorizing ability and fixation rate), **personality traits** (e.g. curiosity, patience, mood and confidence) and several other contextual factors. We only introduced this concept and are not claiming that our system can generate summaries for these variant readers. However, sophisticated systems can be build based on this concept which can extend the document summarization research in a new level.

In this thesis, we simply further tuned the summary quality parameters such as α, β, and γ based on the reader of the summary (e.g. Non-Native reader). Consider a non-native reader of an abstractive summarization system, they can expect a more grammatically readable summary. Native readers can still extract the meaning if the sentences are not properly grammatical. Moreover, the original source documents are expected to be grammatical as they are written by the professional editors. In our work, we introduced β and γ which represents a window between near extractive to complete abstractive sentence selection. For non-native readers, we prefer near extractive summarization. Furthermore, our α parameter measures grammatical quality. In case of non-native readers, we set the parameters ($\alpha = 0.15$, $\beta = 0.5$ and $\gamma = 0.9$).

[55]Other than ILPSumm all the other systems are purely extractive, so didn't report the performance on **R-WE** of those systems

6.6 Summary

In this work, we developed an end-to-end abstractive summarizer in a multi-document setting using a neural **seq2seq** model. We use our sentence level **Para**phrastic **Comp**ression model presented in Chapter 5 to design the document level summary generation system. According to state-of-the-art, this is the first model to generate multi-sentence abstractive summary using a **seq2seq** encoder-decoder model in a multi-document setting. We also propose an optimal solution to a summary length limit problem. Furthermore, we introduced a new concept called "Reader Aware Summary" which can generate summaries for some critical readers.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

In this thesis, we have developed several techniques for modeling both the extractive and the abstractive text summarization tasks. We implemented an ILP-based sentence selection along with TextRank scores and key phrases for extractive multi-document summarization. To increase the readability of the generated summary, we further model the summary coherence. Our novel abstractive fusion generation model jointly performs sentence fusion and paraphrasing using skipgram word embedding model. We designed a neural paraphrastic compression model, which jointly performs compression and paraphrasing in a single sentence in order to produce abstractive compression using a **seq2seq** encoder decoder model. According to the state of the art, this is the first neural model to tackle compression and paraphrasing in a single sentence. We propose simple yet effective solutions to several common problems in neural **seq2seq** models such as redundant repetition and unknown token replacement. For ensuring pure sentence abstraction, we propose several novel sentence abstraction techniques which jointly perform sentence compression, fusion and paraphrasing at the sentence level. Our sentence level models improve the information coverage as well as the grammaticality of the generated sentences. Furthermore, we applied our sentence abstraction techniques to the multi-document setting. We also introduce an optimal solution to the summary length limit problem which was overlooked by the previous models. At the end of this thesis, we also introduced a new concept called "**Reader Aware Summary**" which can generate summaries for some critical readers (e.g. non-native readers). For the sentence level tasks, we conduct our experiments on human generated abstractive compression datasets and evaluate our system on several newly proposed Machine Translation (MT) evaluation metrics. In the case of document level summary, we conduct experiments on the Document Understanding Conference (DUC) 2004 datasets using ROUGE toolkit. Our experiments demonstrate that the methods bring significant improvements over the state-of-the-art methods.

7.2 Future Work

Although the results we obtained have shown the effectiveness of the proposed sentence level and document level models, it could be further improved in a number of ways:

- We will focus on jointly extracting the sentences to maximize the information coverage and readability while minimizing redundancy using a single ILP model.
- We have plans to propose a neural paraphrastic fusion model using seq2seq encoder decoder framework. We can consider using the dataset proposed by (Hermann et al., 2015) in our task.
- We can modify our **seq2seq** encoder decoder model with recently proposed hierarchical attention networks (Yang et al., 2016) to encode a full document or to some extent a document set with our model.
- Syntactic reorganization of natural language sentences are extremely difficult. In future, we will try to propose a model for this using Bidirectional Beam Search (Sun et al., 2017) which has been shown to perform well on image caption generation.
- We will conduct some extensive experiments for our reader level summary generation using some readability metrics[56]. We will also try to propose a full end-to-end model focusing on different critical readers.

[56]https://en.wikipedia.org/wiki/Readability

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, IJCAI'15, pages 1208–1214. http://dl.acm.org/citation.cfm?id=2832415.2832417.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *J. Artif. Int. Res.* 17(1):35–55. http://dl.acm.org/citation.cfm?id=1622810.1622812.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Comput. Linguist.* 34(1):1–34.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Comput. Linguist.* 31(3):297–328. https://doi.org/10.1162/089120105774321091.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1171–1179.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.* 5(2):157–166. https://doi.org/10.1109/72.279181.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3:1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3:993–1022. http://dl.acm.org/citation.cfm?id=944919.944937.

Florian Boudin and Emmanuel Morin. 2013. Keyphrase extraction for n-best reranking in multi-sentence compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 298–305. http://www.aclweb.org/anthology/N13-1030.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.* 19(2):263–311. http://dl.acm.org/citation.cfm?id=972470.972474.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*. Reykjavík, Iceland. https://kheafield.com/papers/stanford/crawl_paper.pdf.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '98, pages 335–336. https://doi.org/10.1145/290941.291025.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 484–494.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches pages 103–111. http://www.aclweb.org/anthology/W14-4012.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 93–98. http://www.aclweb.org/anthology/N16-1012.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555. http://arxiv.org/abs/1412.3555.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 377–384.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:399–429.

William W. Cohen, Robert E. Schapire, and Yoram Singer. 1999. Learning to order things. *J. Artif. Int. Res.* 10(1):243–270. http://dl.acm.org/citation.cfm?id=1622859.1622867.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *J. Artif. Int. Res.* 34(1):637–674.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 357–361. http://anthology.aclweb.org/P16-2058.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 376–380. http://www.aclweb.org/anthology/W14-3348.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '04.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.* 22(1):457–479.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 322–330. http://dl.acm.org/citation.cfm?id=1873781.1873818.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*. pages 1481–1491.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '08, pages 177–185. http://dl.acm.org/citation.cfm?id=1613715.1613741.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, ILP '09, pages 10–18. http://dl.acm.org/citation.cfm?id=1611638.1611640.

Alex Graves, Navdeep Jaitly, and Abdel rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *ASRU*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. 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, Berlin, Germany, pages 1631–1640. http://www.aclweb.org/anthology/P16-1154.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. 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, Berlin, Germany, pages 140–149. http://www.aclweb.org/anthology/P16-1014.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '09, pages 362–370.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. IEEE Computer Society, pages 770–778.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*. http://arxiv.org/abs/1506.03340.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kai Hong, John Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland, pages 1608–1616. ACL Anthology Identifier: L14-1070.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 712–721.

Tomoyuki Kajiwara and Mamoru Komachi. 2016. Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1147–1158.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the*

2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Austin, Texas, pages 1328–1338.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, ICML'15, pages 957–966. http://dl.acm.org/citation.cfm?id=3045118.3045221.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25(2-3):259–284.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'05, pages 1085–1090.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. In *Transactions of the Association for Computational Linguistics (TACL), 2017*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 302–308. http://www.aclweb.org/anthology/P14-2050.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 510–520.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL HLT '12, pages 182–190.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 317–320. http://www.aclweb.org/anthology/N10-1044.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Association for Computational Linguistics, Denver, Colorado, pages 1–7. http://www.aclweb.org/anthology/W15-1501.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*. Association for Computational Linguistics, Barcelona, Spain, pages 404–411. http://www.aclweb.org/anthology/W04-3252.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://dblp.uni-trier.de/db/journals/corr/corr1301.htmlabs-1301-3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

Fionn Murtagh and Pierre Legendre. 2014. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *J. Classif.* 31(3):274–295. https://doi.org/10.1007/s00357-014-9161-z.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016* page 280.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge*

Base Construction and Web-scale Knowledge Extraction. Association for Computational Linguistics, Stroudsburg, PA, USA, AKBC-WEKEX ’12, pages 95–100. http://dl.acm.org/citation.cfm?id=2391200.2391218.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR* abs/1703.01619. http://arxiv.org/abs/1703.01619.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1925–1930.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. 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, Stroudsburg, PA, USA, ACL ’02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1949–1954.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, IJCAI’15, pages 1298–1304.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. 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, Beijing, China, pages 425–430. http://www.aclweb.org/anthology/P15-2070.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, John Wiley and Sons, Ltd, pages 1–20. https://doi.org/10.1002/9780470689646.ch1.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *CoRR* abs/1608.02927.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.* 45(11):2673–2681. https://doi.org/10.1109/78.650093.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1073–1083. http://aclweb.org/anthology/P17-1099.

Christophe Servan, Alexandre Berard, zied elloumi, Hervé Blanchon, and Laurent Besacier. 2016. Word2vec vs dbnary: Augmenting meteor using vector representations or lexical resources? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1159–1168. http://aclweb.org/anthology/C16-1110.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond K. Wong, and Fang Chen. 2016. An efficient approach for multi-sentence compression. In *Proceedings of The 8th Asian Conference on Machine Learning*. PMLR, The University of Waikato, Hamilton, New Zealand, volume 63 of *Proceedings of Machine Learning Research*, pages 414–429. http://proceedings.mlr.press/v63/ShafieiBavani24.html.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1526–1534. https://aclweb.org/anthology/D16-1159.

Qing Sun, Stefan Lee, and Dhruv Batra. 2017. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. *CoRR* abs/1705.08759.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 291–297. http://www.aclweb.org/anthology/E17-2047.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 340–350.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 76–85. http://www.aclweb.org/anthology/P16-1008.

Dung Tran Tuan, Nam Van Chi, and Minh-Quoc Nghiem. 2017. *Multi-sentence Compression Using Word Graph and Integer Linear Programming*, Springer International Publishing, Cham, pages 367–377. https://doi.org/10.1007/978-3-319-56660-3$_3$2.

Xun Wang, Masaaki Nishino, Tsutomu Hirao, Katsuhito Sudoh, and Masaaki Nagata. 2016. Exploring text links for coherent multi-document summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 213–223.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics* 3:283–297. https://transacl.org/ojs/index.php/tacl/article/view/549.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics* 4:401–415. https://www.transacl.org/ojs/index.php/tacl/article/view/741.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1095–1104. http://aclweb.org/anthology/P17-1101.

Appendix A

Smart Stopwords List

Table A.1: Smart Stopwords List

a	contain	hers	nine	some	very
a’s	containing	herself	no	somebody	via
able	contains	hi	nobody	somehow	viz
about	corresponding	him	non	someone	vs
above	could	himself	none	something	w
according	couldn’t	his	noone	sometime	want
accordingly	course	hither	nor	sometimes	wants
across	currently	hopefully	normally	somewhat	was
actually	d	how	not	somewhere	wasn’t
after	definitely	howbeit	nothing	soon	way
afterwards	described	however	novel	sorry	we
again	despite	i	now	specified	we’d
against	did	i’d	nowhere	specify	we’ll
ain’t	didn’t	i’ll	o	specifying	we’re
all	different	i’m	obviously	still	we’ve
allow	do	i’ve	of	sub	welcome
allows	does	ie	off	such	well
almost	doesn’t	if	often	sup	went
alone	doing	ignored	oh	sure	were
along	don’t	immediate	ok	t	weren’t
already	done	in	okay	t’s	what
also	down	inasmuch	old	take	what’s
although	downwards	inc	on	taken	whatever
always	during	indeed	once	tell	when
am	e	indicate	one	tends	whence
among	each	indicated	ones	th	whenever
amongst	edu	indicates	only	than	where
an	eg	inner	onto	thank	where’s
and	eight	insofar	or	thanks	whereafter
another	either	instead	other	thanx	whereas
any	else	into	others	that	whereby
anybody	elsewhere	inward	otherwise	that’s	wherein
anyhow	enough	is	ought	thats	whereupon
anyone	entirely	isn’t	our	the	wherever

anything	especially	it	ours	their	whether
anyway	et	it’d	ourselves	theirs	which
anyways	etc	it’ll	out	them	while
anywhere	even	it’s	outside	themselves	whither
apart	ever	its	over	then	who
appear	every	itself	overall	thence	who’s
appreciate	everybody	j	own	there	whoever
appropriate	everyone	just	p	there’s	whole
are	everything	k	particular	thereafter	whom
aren’t	everywhere	keep	particularly	thereby	whose
around	ex	keeps	per	therefore	why
as	exactly	kept	perhaps	therein	will
aside	example	know	placed	theres	willing
ask	except	knows	please	thereupon	wish
asking	f	known	plus	these	with
associated	far	l	possible	they	within
at	few	last	presumably	they’d	without
available	fifth	lately	probably	they’ll	won’t
away	first	later	provides	they’re	wonder
awfully	five	latter	q	they’ve	would
b	followed	latterly	que	think	would
be	following	least	quite	third	wouldn’t
became	follows	less	qv	this	x
because	for	lest	r	thorough	y
become	former	let	rather	thoroughly	yes
becomes	formerly	let’s	rd	those	yet
becoming	forth	like	re	though	you
been	four	liked	really	three	you’d
before	from	likely	reasonably	through	you’ll
beforehand	further	little	regarding	throughout	you’re
behind	furthermore	look	regardless	thru	you’ve

being	g	looking	regards	thus	your
believe	get	looks	relatively	to	yours
below	gets	ltd	respectively	together	yourself
beside	getting	m	right	too	yourselves
besides	given	mainly	s	took	z
best	gives	many	said	toward	zero
better	go	may	same	towards	
between	goes	maybe	saw	tried	
beyond	going	me	say	tries	
both	gone	mean	saying	truly	
brief	got	meanwhile	says	try	
but	gotten	merely	second	trying	
by	greetings	might	secondly	twice	
c	h	more	see	two	
c’mon	had	moreover	seeing	u	
c’s	hadn’t	most	seem	un	
came	happens	mostly	seemed	under	
can	hardly	much	seeming	unfortunately	
can’t	has	must	seems	unless	
cannot	hasn’t	my	seen	unlikely	
cant	have	myself	self	until	
cause	haven’t	n	selves	unto	
causes	having	name	sensible	up	
certain	he	namely	sent	upon	
certainly	he’s	nd	serious	us	
changes	hello	near	seriously	use	
clearly	help	nearly	seven	used	
co	hence	necessary	several	useful	
com	her	need	shall	uses	
come	here	needs	she	using	
comes	here’s	neither	should	usually	
concerning	hereafter	never	shouldn’t	uucp	
consequently	hereby	nevertheless	since	v	
consider	herein	new	six	value	
considering	hereupon	next	so	various	

Appendix B

Sample system generated extractive summaries

Sample summaries for extractive multi-document summarization

In the following, we show some examples of our system-generated summary using our extractive coherent summary generation model described in Chapter 3 and human-written reference summary from DUC 2004 (Task-2) dataset .

Human-written summary for the document set D31032t

President Boris Yeltsin's health has become a matter of great concern to the Russian leadership. The concern began in 1996 when he had a heart attack followed by bypass surgery. Illness has often sidelined him during his seven years in power. He recently cut short a trip to Central Asia because of a respiratory infection and he later canceled two out-of-country summits. This revived questions about his ability to lead Russia through any crisis. Yeltsin refuses to admit he is seriously ill and his condition is kept secret, even the cause for burns on his hands. Russia's leaders are calling for his resignation and question his legal right to seek reelection.

System-generated coherent summary for the document set D31032t

President Boris Yeltsin has suffered minor burns on his right hand, his press office said Thursday. Yeltsin has a history of health problems, and underwent heart bypass surgery in 1996. The president insists he has no major health problems and will serve out the remaining two years of his term. A respiratory infection forced him to cut short a trip to Central Asia earlier this week. The trip is Yeltsin's first high-profile foray since an economic crisis swamped his country in August. In an interview with the British Broadcasting Corp., he stopped short of calling on Yeltsin to step down.

Human-written summary for the document set D30026t

As the U.S. government pressed its antitrust case against Microsoft Corp. in November 1998, America Online (AOL) proposed an alliance with Netscape Communications and Sun Microsystems. The three-pronged deal promised to provide on-line services, Internet software and electronic commerce. AOL was to buy Netscape and forge a partnership with Sun, benefiting all three and giving technological independence from Microsoft. Microsoft lawyers argued unconvincingly that AOL's purchase of Netscape would undermine the government's antitrust case, based in large part on Netscape's complaint. It remained to be seen whether AOL could achieve a vast virtual mall.

System-generated coherent summary for the document set D30026t

Sun's version of Unix, Solaris, is among the most popular operating systems for the large, powerful computers that run Netscape's server software. Even before the Netscape deal, America Online was moving to provide some electronic commerce software and services. Microsoft Corp. argued in federal court Monday that the proposed acquisition of Netscape Communications Corp. by America Online seriously undermined the government's antitrust suit against the software giant. It would strengthen two of Microsoft's leading rivals, AOL and Sun Microsystems. Its Navigator was the runaway leader in the market for the browser software used to navigate the World Wide Web.

Appendix C

Sample system generated abstractive summaries

Sample summaries for abstractive multi-document summarization

In the following, we show some examples of our system-generated summary using our abstractive coherent summary generation model which is based on paraphrastic sentence fusion described in Chapter 4 and human-written reference summary from DUC 2004 (Task-2) dataset .

Human-written summary for the document set D30002t

Hurricane Mitch, category 5 hurricane, brought widespread death and destruction to Central American. Especially hard hit was Honduras where an estimated 6,076 people lost their lives. The hurricane, which lingered off the coast of Honduras for 3 days before moving off, flooded large areas, destroying crops and property. The U.S. and European Union were joined by Pope John Paul II in a call for money and workers to help the stricken area. President Clinton sent Tipper Gore, wife of Vice President Gore to the area to deliver much needed supplies to the area, demonstrating U.S. commitment to the recovery of the region.

System-generated coherent summary for the document set D30002t

Hurricane Mitch killed an estimated 9,000 people throughout Central America in a disaster of such proportions that relief agencies have been overwhelmed. Jerry Jarrell, the weather center director, said Mitch was the strongest hurricane to strike the Caribbean since 1988, when Gilbert killed more than 300 people. “Mitch is closing in,” said Monterrey Cardenas, mayor of Utila, an island 20 miles (32 kilometers) off the Honduran coast. In honduras, at least 231 deaths have been blamed on mitch, bringing the storm’s death toll in the region to 357, the national emergency commission said saturday.

Human-written summary for the document set D30003t

Pinochet arrested in London on Oct. 16 at a Spanish judge's request for atrocities against Spaniards in Chile during his rule. Castro, Chilean legislators and Pinochet's lawyers protested and claimed he had diplomatic immunity. His wife asked for his release because he was recovering from recent back surgery. Pinochet visited Thatcher before his surgery. The British and Spanish governments defended the arrest, saying it was strictly a legal matter. The EC president hoped Pinochet would stand trial. None of his Swiss accounts have been frozen yet. The Swiss government also asked for his arrest for the 1977 disappearance of a Swiss-Chilean student.

System-generated coherent summary for the document set D30003t

Pinochet was arrested in London on Oct. 16 at the instigation of Spanish magistrate Baltasar Garzon who is seeking to extradite the former dictator on charges of genocide, terrorism and torture. Pinochet's detention in the London clinic where he was recovering from back surgery. The Spanish and British governments appeared Wednesday to be seeking shelter from the political storm brewing over the possible extradition of former Chilean dictator Augusto Pinochet to Spain. The court's national court have appealed the judicial investigations into human rights abuses in chile and argentina that underpin garzon's arrest warrant.

Appendix D

Sample system generated neural abstractive summaries

Sample summaries for neural abstractive multi-document summarization

In the following, we show some examples of our system-generated summary using our neural abstractive coherent summary generation model (chapter 6) which is based on the neural **seq2seq** paraphrastic sentence compression generation model described in Chapter 5 and human-written reference summary from DUC 2004 (Task-2) dataset .

Human-written summary for the document set D30011t

Malaysian Prime Minister Mahathir Mohamad ruled adroitly for 17 years until September 1998 when he suddenly reversed his economic policy and fired his popular deputy and heir apparent, Anwar Ibrahim. Anwar organized a political opposition leading Mahathir to arrest him. Mahathir met street demonsrations with tear gas and water cannon, but his censorship did not reach Anwar's internet support. News that police had beaten Anwar brutally brought protests from around the world, but the Malaysian trade minister discounted any unrest. Anwar remained in custody as lawyers appealed. Malaysia hardly provided a salubrious setting for the forthcoming economic summit.

System-generated coherent summary for the document set D30011t

on sept. 2 , malaysia 's prime minister mahathir mohamad fired anwar , calling him morally unfit for office. the two had differed over economic policy and anwar says mahathir feared him as an alternative leader. anwar since has been charged with illegal homosexual acts and corruption in connection with those allegations. anwar was arrested sept. 20 under the internal security act , which allows jail without trial. the arrest of former malaysian deputy prime minister anwar 's trade minister said thursday. last week , philippine president joseph estrada said he was considering not going to the asia-pacific economic cooperation forum because of anwar ibrahim 's arrest.

Human-written summary for the document set D30022t

On the eve of China's signing the International Covenant of Civil and Political Rights (ICCPR) in October 1998, police detained Chinese human rights advocate Qin Yongmin for questioning. Eight weeks after signing the ICCPR, Chinese police arrested Qin and an associate in the China Democracy Party (CDP), Xu Wenli, without stating charges. Another CDP leader already in custody, Wang Youcai, was accused of "inciting the overthrow of the government". Qin and Wang went to trial in December for inciting subversion. Police pressure on potential defense attorneys forced the accused to mount their own defenses. Xu Wenli had not yet been charged.

System-generated coherent summary for the document set D30022t

china plans to sign the international covenant on civil and political rights on monday at the united nations. almost 200 dissidents signed a letter to the chinese prisons , labor camps or detention centers , said the information center of human rights and democratic movement in china. wang was a student leader of the 1989 tiananmen square democracy movement. china 's government said thursday that two prominent dissidents , both promoters of the new party. as the founder of the suppressed new political party , the china democracy party , he publicly announced its charter in june during president clinton 's visit to china.