MULTI-DOCUMENT SUMMARIZATION BASED ON DOCUMENT CLUSTERING AND NEURAL SENTENCE FUSION

TANVIR AHMED FUAD
Bachelor of Science, Military Institute of Science and Technology (MIST), 2015

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

MULTI-DOCUMENT SUMMARIZATION BASED ON DOCUMENT CLUSTERING AND NEURAL SENTENCE FUSION

TANVIR AHMED FUAD

Date of Defence: September 11, 2018

Dr. Y. Chali Supervisor	Professor	Ph.D.
Dr. W. Osborn Committee Member	Associate Professor	Ph.D.
Dr. J. Zhang Committee Member	Associate Professor	Ph.D.
Dr. H. Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

I dedicate this thesis to the almighty and my parents who inspired me at each and every step of my life.

Abstract

In this thesis, we have approached a technique for tackling abstractive text summarization tasks with state-of-the-art results. We have proposed a novel method to improve multi-document summarization. The lack of large multi-document human-authored summaries needed to train seq2seq encoder-decoder models and the inaccuracy in representing multiple long documents into a fixed size vector inspired us to design complementary models for two different tasks such as sentence clustering and neural sentence fusion. In this thesis, we minimize the risk of producing incorrect fact by encoding a related set of sentences as an input to the encoder. We applied our complementary models to implement a full abstractive multi-document summarization system which simultaneously considers importance, coverage, and diversity under a desired length limit. We conduct extensive experiments for all the proposed models which bring significant improvements over the state-of-the-art methods across different evaluation metrics.

Acknowledgments

"Bismillah ir-Rahman ir-Rahim"

In the name of Allah the almighty, most Gracious, most Compassionate...

I would like to thank the almighty Allah for giving me energy and ability to complete this thesis.

I would like to express my heartfelt thanks and sincere gratitude to my supervisor Professor Dr. 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 very much grateful to him.

I would also like to thank my M.Sc. supervisory committee members Dr. Wendy Osborn, and Dr. John Zhang for their time and effort.

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 Xp GPU machine to conduct my experiments. I am also deeply grateful to my supervisor for the financial assistance.

A special thanks to a special sister who wished me just before my flight, who came to the airport just to say goodbye to me. Probably, she would never read this. But still I like her very much for being an awesome personality. Being only child of my parents I never knew what it is like to have a sister. She gave me a heavenly feeling of having a sister. I never had any sad feeling of not having a sister before I met her. She gifted me a Ferrari. Just to remember her, I always carry that Ferrari with me. Wherever she is, I wish her all

the success in her life.

I am forever thankful to my friends for their support, friendship and encouragement which is worth more than I can express on paper. I am also thankful to those people who made my life in Lethbridge very challenging, comfortable and enjoyable.

Acknowledging to this person will never be enough but still, I am very much grateful to Mir Tafseer Nayeem, without whom this thesis would probably remain in imagination. He has been and will always be more than an inspiration in my life. This page is too short to express my gratitude towards him. In short, probably I would have to leave Lethbridge empty handed but because of him at least I got a chance to finish this work.

Last but not the least, I am grateful to my father Md Shahidul Islam who still goes to office by riding a bicycle just to support me and my mother Shahnaz Perveen who still does not sleep any night just to pray to the almighty for their ungrateful son. I still can not believe how they managed to teach me all those good things in my life.

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))

"Information is what you need"- this is the motto of the Internet world. In most cases someone needs some information at a minimum cost. The Internet consists of a huge collection of textual documents with an exponential growth rate. For a single query the Google, Bing and Yahoo-search engines generally return thousands of links. It becomes difficult to choose from this large scale result. Moreover, on a variety of topics, the Internet contains millions of texts. This is the reason for data redundancy and the difficulty to extract concise information. Sometimes users become so overwhelmed while reading large documents that they may miss interesting or important information. These concerns lead to the development of automatic summarization systems which focus on creating short summaries from a single document or a related set of documents. A good summary consists of most of the information from the original documents, while being non-redundant and grammatically readable. There are several methods for document summarization. Extractive summarization contains sentences without any modification from the original documents. Also, with extractive summaries, it is difficult to explain the whole thing since any modification is restricted. On the other hand, abstractive summarization generates different words and sentences from the original document to represent the summary. In this thesis, we have

proposed some novel approaches with outperforming state-of-the-art results in the area of abstractive multi-document text summarization.

1.2 Contributions

- We have designed an unsupervised sentence clustering model which is simple, effective, and outperforms several popular clustering methods when tested on two public datasets.
- We proposed a neural sentence fusion model. To the best of our knowledge, our work is the first to investigate adapting neural models to the sentence fusion task.
- We developed an ILP (Integer Linear Programming) based sentence selection process containing diverse information for given length for abstractive multi-document summarization.
- We applied our sentence clustering, sentence fusion model and sentence selection process to design a full abstractive multi-document summarization system and achieved state-of-the-art results on two different datasets.

1.3 Thesis Overview

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 text clustering. Chapter 4 is devoted to our neural sentence fusion model. Chapter 5 describes our novel approaches for the abstractive multi-document sentence generation using a tensor2tensor[1] based Transformer model at sentence level. Then both clustering and fusion models are applied to the multi-document summary generation.

[1] https://github.com/tensorflow/tensor2tensor

Chapter 2

Background

To complete this work, a hierarchical model consisting of several models has been proposed. To achieve this some previously implemented open source tools and models are used. In this chapter, we will be mostly discussing about these open source tools and models and their benefits which led us to choose them for our work.

2.1 Sentence Similarity

Sentences can be similar in many aspects. Sentences can have similar structures, similar topics or similar ideas. Many Natural Language Processing (NLP) tasks are required to be dealt with on similar sentences. For example, question-answer related sites like Quora [2] or StackOverflow [3] may need to determine whether a similar or same question has been asked before. There are many ways to compute similarity between two sentences based on requirement. Here, two sentence similarity approaches are discussed.

2.1.1 Cosine Similarity

Cosine similarity[4] is measured between two sentence vectors, which should be non-zero vectors. It is basically the cosine distance between two vectors. To compute cosine similarity, the sentences need to be converted first into a vector representation. This process can be performed in various ways. This is discussed in detail later in the thesis. After converting the sentences into vectors, the similarity between two vectors s_i and s_j is computed using:

[2]https://www.quora.com/

[3]https://stackoverflow.com/

[4]https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/cosdist.htm

$$cosine_{similarity}(s_i, s_j) = \frac{s_i \cdot s_j}{\|s_i\| \|s_j\|}$$

2.1.2 Jaccard Similarity

Jaccard similarity is computed using the Jaccard Index [5] between two sentences. Jaccard Index is also sometimes referred as "Intersection over Union". To compute the Jaccard Similarity between two sentences, both sentences are represented as sets. The jaccard Similarity of two sets A and B can be found using:

$$jaccard_{similarity}(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

2.2 Automatic Text Summarization: Overview of Recent Work

The task of automatic document summarization aims at finding the most relevant information in a text and presenting it in a condensed form. A good summary should retain the most important contents of the original document or a cluster of related documents, while being coherent, non-redundant and grammatically readable. There are two types of summarization: abstractive summarization and extractive summarization. Abstractive methods need extensive natural language generation to rewrite the sentences (Chali et al., 2017). 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. The abstractive techniques which are traditionally used are sentence compression, syntactic reorganization and lexical paraphrasing. Summarization is classified as single-document or multi-document based on the number of source documents. The

[5] https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/jaccard.htm

information overlap between the documents from the same topic makes the multi-document summarization more challenging than the task of summarizing single documents. However, in case of multi-document summarization where source documents usually contain similar information, the extractive methods would produce redundant summary or biased towards specific source document (Nayeem and Chali, 2017a).

2.2.1 Extractive Summarization

Over the past few decades, several extractive approaches have been proposed for automatic summary generation that combine a number of machine learning, graph-based and optimization techniques. Computing sentence importance for text summarization, LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004) are graph-based methods. The RegSum system (Hong and Nenkova, 2014) employs a supervised model for predicting word importance. Instead of greedily adding sentences to form a summary, treating multi-document summarization as a submodular maximization problem has proven successful by Lin and Bilmes (2011). The most widely used practice is to formulate the problem as integer linear programming (ILP). Therefore, concept-based ILP (Gillick and Favre, 2009; Nayeem and Chali, 2017a) 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 consider 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) and Parveen et al. (2015) proposed graph-based techniques to tackle coherence, which is also limited to single document summarization. A multi-document summarization system was recently

proposed by Wang et al. (2016), that combines both coherence and informativeness but this system is limited to syntactic linkages between named entities.

2.2.2 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 significant interest on compressive document summarization that attempts to compress original sentences to form a summary (Clarke and Lapata, 2006, 2008; Filippova, 2010) as a first intermediate step towards abstractive summarization. Compressive 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 the syntactic parsing to build the 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 POS (Parts of Speech) tagger and a list of stopwords have been proposed by Filippova (2010). A directed word graph is constructed in which nodes represent words and edges represent the adjacency between words in a sentence. Hence, compressed sentences are generated by finding the 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 in order to improve informativity in these works (Nayeem and Chali, 2017b; Nayeem et al., 2017).

Banerjee et al. (2015) proposed an abstractive multi-document summarization system using the sentence fusion approach of Filippova (2010) combined with Integer Linear Programming (ILP) sentence selection. Following Banerjee et al. (2015)'s work, several recent approaches have been proposed with slight modifications. Multiword Expressions (MWE)

was exploited by ShafieiBavani et al. (2016) to produce more informative compressions. Recently, Tuan et al. (2017) included syntax factors along with Banerjee et al. (2015) to improve performance. However, all of the above mentioned systems try to produce compressions by copying the source sentence words, without any paraphrasing in the process.

Recently end-to-end training with encoder-decoder neural networks have achieved huge success for abstractive summarization. These systems have adopted techniques such as encoder-decoder with attention (Bahdanau et al., 2015; Luong et al., 2015) neural network models 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. Unfortunately, this line of research under the term sentence summarization (Rush et al., 2015), which can generate only a single sentence, somewhat misleadingly called text summarization in some follow-up research works (Nallapati et al., 2016; Chopra et al., 2016; Suzuki and Nagata, 2017; Zhou et al., 2017; Ma et al., 2017; Nayeem et al., 2018). There are some limitations to the above mentioned models, one of which is the produced output is also very short (about 75 characters). Similar to headline generation, their model produces ungrammatical sentences during generation. However, there are some recent attempts which use the CNN/DailyMail corpus (Hermann et al., 2015) as supervised training data to generate a multi-sentence summary from a single document (See et al., 2017; Li et al., 2017b; Paulus et al., 2017; Narayan et al., 2018a,b; Chali et al., 2017). The recent abstractive summarization models actually produce compressed summaries by deleting the words from a single source document, with no direct paraphrasing being involved in the process. Hence, no new words were generated which are different form the source document words (other than morphological variation), which is pointed out by their own experimental results. Very recently, some researchers employ a neural network based framework to tackle the summarization problem in a multi-document setting (Yasunaga et al., 2017; Li et al., 2017a). However, Yasunaga et al. (2017)'s work is limited to extractive summmarization while Li et al. (2017a)'s work is limited to compressive sum-

mary generation using an ILP based model, and there is no explicit redundancy control in the summary side. Unfortunately, full abstractive summarization in a multi-document setting still lacks satisfactory solutions due to the lack of large multi-document summarization datasets needed to train the computationally expensive sequence-to-sequence models. In this paper, we tackle this issue in an unsupervised way using deep representation learning.

2.2.3 Automatic Summary Evaluation

To determine the quality of a machine generated summary by comparing it against a reference or a set of reference summaries (generally human-annotated) ROUGE (Lin, 2004) (Recall-Oriented Understudy for Gisting Evaluation)[6] is an automatic tool used widely for this purpose. There are 4 different ROUGE metrics - namely ROUGE-N (1,2,3,4), ROUGE-L, ROUGE-W, and ROUGE-S.

- **ROUGE-N** A summary evaluation 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 the LCS (Longest Common Sub-sequence).
- **ROUGE-W** For evaluating ROUGE-W, different weights are assigned to consecutive in-sequence matches in the LCS.
- **ROUGE-S** If there is any pair of words in the sentence order which allows for arbitrary gaps, this is used to evaluate ROUGE-S. Sometimes, it is 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[7]. 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".

[6]ROUGE package link: http://www.berouge.com
[7]http://www.rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/

The most commonly used among the above mentioned measures for multi-document summarization research is **ROUGE-N**. The number of overlapping n-grams is counted to evaluate between the system summary and human written reference summaries. ROUGE-N can be defined 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,

$\mathbf{g_n}$ is the maximum number of n-grams co-occurring in a candidate summary and

$\mathbf{Count_{match}(g_n)}$ is the set of reference summaries (Lin, 2004).

While evaluating, if multiple reference summaries are used, a pairwise summary-level ROUGE-N score is computed between a candidate system generated summary, s and every human annotated reference, r_i from the reference set, $R = \{r_1, r_2, \ldots, r_n\}$. The maximum among the summary-level ROUGE-N scores is the final ROUGE-N score. It can be described as follows:

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

For example:

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

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

If uni-grams are considered only, the number of overlapping words between the system summary and reference summary is 10. However, we can actually compute the precision and recall using the overlap of words to get a good quantitative value. Recall in terms of ROUGE simply means how much of the reference summary the system summary is

acquiring. If the individual words are considered only, the recall can be computed as:

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

It can be easily identified, that all the words in the human annotated reference summary have been captured by the machine generated system summary. However, a machine generated summary (system summary) can be extremely long, if it captures all of the words from the human annotated reference summary. In the system summary most of the words may be useless, which results in a summary with redundancy and repetitive information. Here, precision[8] is required. Precision is defined as how much of the system summary was actually 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 means, that 10 out of the 12 words in the system summary were relevant. Let us assume, 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 is:

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

The precision score has now decreased. The reason behind this is, a few redundant words appeared in the system summary. When we try to generate summaries that are concise in nature, the precision is really crucial . Therefore, the best way is to compute both the **Precision** and **Recall**. Sometimes, the system summaries are forced to be concise given some constraints (such as length limit constraint). Then using just the recall should be sufficient since precision is of less concern in this case. In this thesis, the limited length re-

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

call measure is only reported in our experiment. Moreover, the performance has also been reported in terms of **ROUGE-SU4**, where **S** means skip-bigram (match 2 non contiguous words with other words in between) allowing rephrasing and sentence reorganization. As the ROUGE score is supposed to evaluate abstractive summaries, its a good measure. For other in between words, **U4** has been used which means maximum of 4 unigram words are allowed within a skip-bigram.

2.3 Word Embedding

Word embedding is a process of 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.

2.3.1 One-Hot Vectors

Before building the above applications, the similarity between two words, sentences or even paragraphs has to be measured. Through a vector space model, the one-hot vector is a representation of all the words. The vector representation has the corresponding entry in the vector for each word as 1 (presence), and all other entries as 0 (absence). The size of the dictionary or vocabulary will be the length of one-hot vectors. Cosine similarity[9] on one-hot vectors is not capable of capturing semantic information when documents say exactly same thing in entirely different words. Let us consider these two following news examples:

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

These two sentences do not have any word in common (except for the stopwords such as *the* and *in*, which is not so important for measuring semantic similarity). According to

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

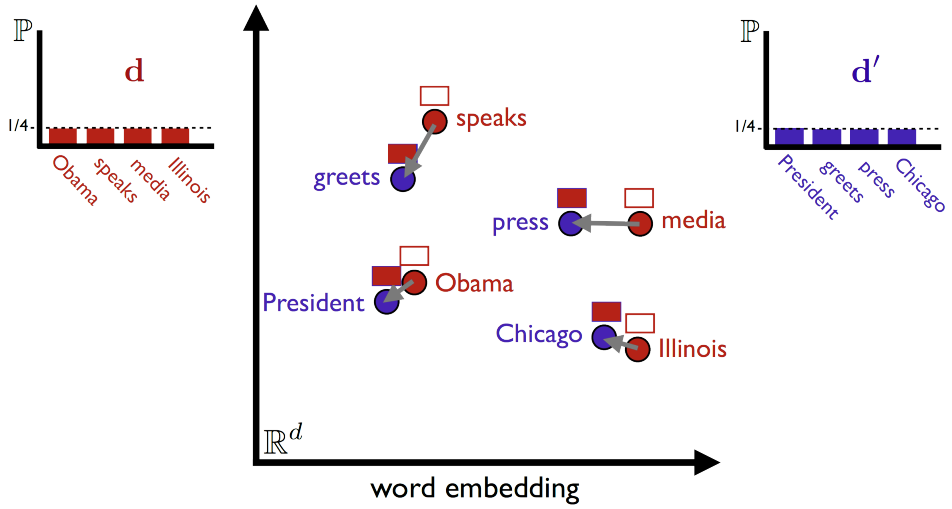

Figure 2.1: Visualization of word to word similarity of all non-stop words from both headlines is embedded into a word2vec space (Nayeem et al., 2017).

the one-hot vectors representation, their cosine distance would be maximal. To measure their semantic similarity properly, further information is needed, which can be learned using large amounts of data through machine learning models (Kusner et al., 2015). Figure 2.1 taken from Kusner et al. (2015) visualizes the word to word similarity of the example headlines.

2.3.2 Word2Vec

In distributional semantics, vector space models have been used 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 first introduced in Bengio et al. (2003) where a word embedding model was proposed 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 represents the i^{th} word in the sentence. In the language model, higher probabilities are assigned to grammatical and meaningful sentences, and lower probabilities are assigned to meaningless sentences. For example, let us assume that we are searching for something on the Internet using Figure

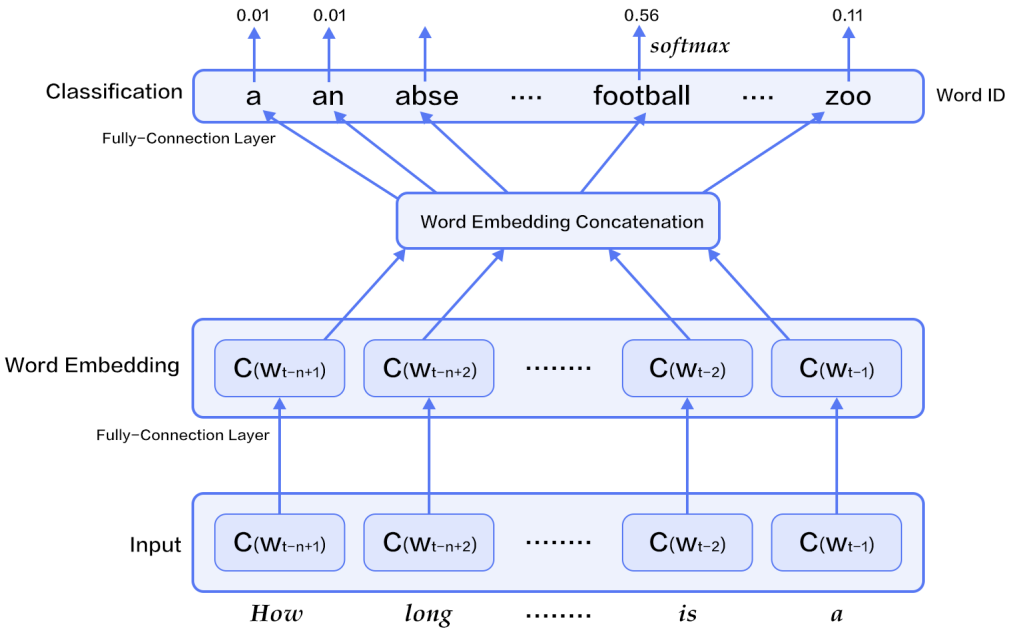

Figure 2.2: N-gram neural language model (Nayeem et al., 2017).

2.2[10], if we write "How long is a", the search engines would suggest the next word "football". The reason behind this is the probability according to the language model among all words in the target vocabulary of "How long is a football" is very high.

To create high-dimensional (50 to 300 dimensional) representations of words in an unsupervised manner from a large amount of text, Word2vec (Mikolov et al., 2013b,a) is the most popular of the word embedding models for learning word embeddings . As illustrated in Figure 2.3, Word2Vec embeds words in a continuous vector space where semantically similar words are placed as nearby points to each other. Recently, it was shown that the word vectors are able to capture many linguistic regularities. For example, vector arithmetic operations [vector("Paris") - vector("France") + vector("Italy")] implement a vector that is very close to vector("Rome"), and [vector("king") - vector("man") + vector("woman")] is close to vector("queen")[11]. Mikolov et al. (2013b) defined two architectures for learning word

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

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

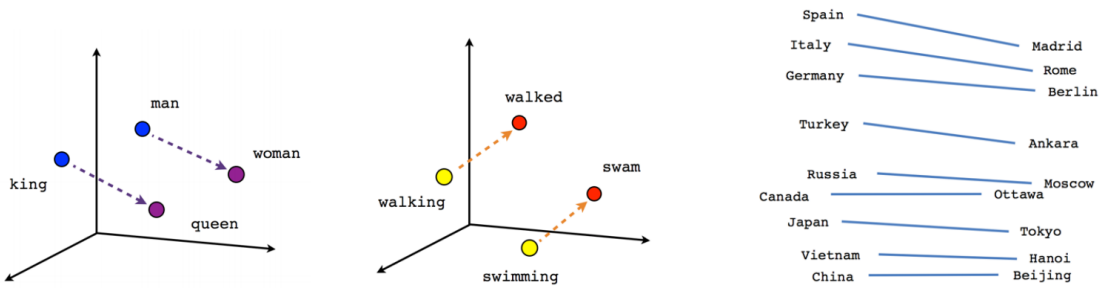

Figure 2.3: Visualization of semantic relationships, e.g. male-female, verb tense and even country-capital relationships between words (Mikolov et al., 2013b).

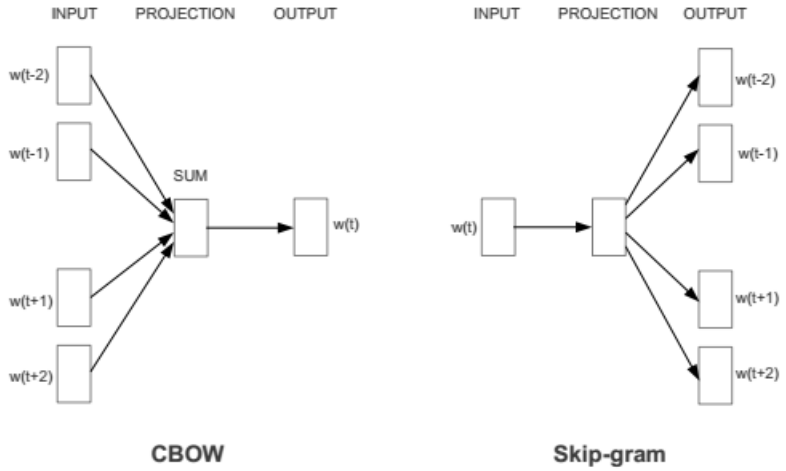

Figure 2.4: CBOW model (left) and Skip-gram model (right) from (Mikolov et al., 2013b).

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).

2.3.3 GloVe

Unlike word2vec, GloVe[12] (Pennington et al., 2014) takes advantage of two primary families of word vectors- 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 prediction based 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.

2.3.4 FastText

Compared to other word embeddings FastText (Mikolov et al., 2018) is very new with very good results. FastText uses a hierarchical classifier instead of a flat structure, which is organized in a tree in the different categories. The depth in the tree of very frequent categories is therefore smaller than for infrequent ones, leading to further computational efficiency. A text is represented using FastText by a low dimensional vector, obtained by summing vectors corresponding to the words that appear in the text. A low dimensional vector is associated with each word of the vocabulary in FastText. This hidden representation is shared among all classifiers for different categories, which allows information about words learned for one category to be used by other categories. These kind of representations is called a bag of words which ignores word order.

2.4 Multi-Sentence Compression (Sentence Fusion)

Multi-sentence compression (**MSC**) can be a useful solution for the summarization problem. It usually takes a group of related sentences and produces an output sentence through merging the sentences about the same topic, retaining the most important information and still maintaining the grammaticality of the generated sentence. MSC (originally called sentence fusion by Barzilay and McKeown (2005)) is a text-to-text generation pro-

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

cess 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.

2.5 Sentence Fusion Evaluation

This section describes the set of automatic metrics which can be useful for sentence fusion evaluation. At first word-overlap metrics are considered and then embedding-based metrics are presented. In every case, given multiple references, the similarity between the prediction and all the references are computed one-by-one, and then the maximum value is selected. After that, the average score is computed for the entire corpus.

2.5.1 Word-overlap based metrics

BLEU

The BLEU metric (Papineni et al., 2002) compares *n*-grams between the candidate sentence and the references. At the corpus-level, the BLEU score is computed using modified precision, which can be described as follows:

$$p_n = \frac{\sum_{C \in \{References\}} \sum_{n-gram \in C} Ct_{clip}(n-gram)}{\sum_{C' \in \{References'\}} \sum_{n-gram' \in C'} Ct_{clip}(n-gram')}$$

where, {References} are the candidate output produced by the system and Ct_{clip} is the clipped count for *n*-gram which is the number of times the *n*-gram, is common to the candidate answer and the reference answer clipped by the maximum number of occurrences of

the *n*-gram in the reference answer. The BLEU-N score is defined as:

$$\text{BLEU-N} = \text{BP } \exp(\sum_{n}^{N} \omega_n \log(p_n))$$

where N represents the maximum length of the n-grams, ω is a weighting that is sometimes uniform and BP is a brevity penalty.

METEOR

The METEOR metric (Banerjee and Lavie, 2005) was first proposed as an evaluation metric which evaluates more effectively at the sentence level. Before computing METEOR score, at first, an alignment between the system generated sentence and the reference sentence is mapped using each uni-gram in the system generated sentence to 0 or 1 uni-gram in the reference sentence. The alignment is based on not only exact matches but aslo stem, synonym, and paraphrase matches. Based on the mapping, uni-gram precision and recall are computed. Then the METEOR score is computed which can be described as follows:

$$\text{METEOR} = F_{mean}(1 - p)$$

where, F_{mean} is the harmonic mean between precision and recall with weight for recall 9 times a high as weight for precision, and p is the penalty.

2.5.2 Embedding based metrics

There are another set of metrics where the cosine similarity is computed between the embeddings of the system generated sentence and the reference sentence instead of relying on word overlaps.

Skip-Thought

The Skip-Thought model (Kiros et al., 2015) uses a recurrent network to encode a given sentence into an embedding, to train in an unsupervised way and then decoded to pro-

duce the preceding and following sentences. The model was trained using the BookCorpus dataset (Zhu et al., 2015). The pre-trained model[13] shared by the author was used in this thesis.

Embedding average

This metric is computed using a sentence-level embedding by averaging the embeddings of the words composing the sentence. It can be described as follows:

$$\bar{e}_C = \frac{\sum_{w \in C} e_w}{|\sum_{w' \in C} e_{w'}|}.$$

where, vectors e_w, represents the embeddings for words w in system generated sentence C.

Vector extrema

Vector extrema (Forgues et al., 2014) is computed in sentence-level embedding by collecting the most extreme value of the embeddings of the words. It can be described as follows:

$$e_{rd} = \begin{cases} \max_{w \in C} e_{wd} \text{if } e_{wd} > |\min_{w' \in C} e_{w'd}| \\ \min_{w \in C} e_{wd} \text{otherwise.} \end{cases}$$

where, d is an index over the dimensions of embedding and C is the system generated sentence.

Greedy matching

Greedy matching does not compute a sentence embedding but a similarity score directly between a candidate C and a reference r (Rus and Lintean, 2012). This similarity score is

[13] https://github.com/ryankiros/skip-thoughts

computed as follows:

$$G(C,r) = \frac{\sum w \in C \max_{\hat{w} \in r} cos\_sim(e_w, w_{\hat{w}})}{|C|}$$

$$GM(C,r) = \frac{G(C,r) + G(r,C)}{2}. \tag{2.1}$$

In other words, each word in the candidate sentence is greedily matched to a word in the reference sentence based on the cosine similarity of their embeddings. The score is an average of these similarities over the number of words in the candidate sentence. The same score is computed by reversing the roles of the candidate and reference sentences and the average of the two scores gives the final similarity score.

2.6 Neural Machine Translation (NMT)

The process of translating from the source language to the target language is called Machine Translation (MT). The input language to the machine translation system is known as the source language, and the output language is known as the target language. In short, machine translation is the task of conversion of a sequence of words in the source language into a sequence of words in the target language. It is one of the most important and well known research topics in the field of Natural Language Processing (Neubig, 2017).

For early automatic MT systems, Statistical Machine Translation (SMT) techniques have been used (Brown et al., 1993). But these statistical machine translation models posses many shortcomings. Pre-processing techniques of SMT heavily relies on processes like word alignment, word segmentation and tokenization, rule-extraction and syntactic parsing. However, the problem is that human designed features cannot cover all possible linguistic variations and cannot use all global features[14]. The recent development of deep learning provides new and better solutions compared to previous approaches to these afore-

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

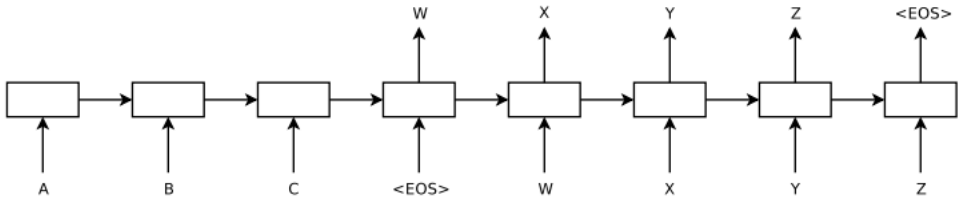

Figure 2.5: Sequence to Sequence Learning with Neural Networks (Sutskever et al., 2014)

mentioned problems of SMT. Neural Machine Translation (NMT) (Sutskever et al., 2014) does not require any pre-designed features. The goal of NMT is to design a fully trainable model where every component is tuned based on a large-scale training corpora in order to maximize its performance.

Considering a sequence of words as the most raw representation of a sentence, 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. In a fixed numbered of vocabulary, each word in a sequence is represented by its integer index. In the English words vocabulary V, which is sorted according to their frequency of appearance in a training corpus, the very first frequent word is represented as an integer 1. Also given, $X = (x_1, x_2, \ldots, x_N)$ a source sentence, and $Y = (y_1, y_2, \ldots, y_M)$ a target sentence where, $N \pm M$ (Sutskever et al., 2014). The NMT model $\mathcal{M}$ attempts 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 network (**seq2seq**) has become very popular in the NLP community to solve the problem of NMT (Sutskever et al., 2014; Bahdanau et al., 2015).

For example, according to the Figure 2.5, we have "ABC" as the input, and "WXYZ" as the output. The two sequences are different in lengths. So the question is, how does seq2seq solve that problem of different sequence lengths? The solution is: two different models are developed, which consists of two separate recurrent neural networks called **Encoder** and

Decoder respectively.

2.6.1 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 Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU).

Encoder

Encoding a sequence consists of three steps where a sentence as a sequence through an encoder:

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

2. There are two problems with the one-hot vector representation.

   - The dimension is very large of each individual word vector .
   - It is very difficult to capture semantic relationship between words in a source sentence. That is why, it is very useful to convert 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 then encoded using RNN:

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

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 word's continuous vector s_N is projected, the RNNs internal state h_n represents a summary of the whole source sentence.

Decoder

The goal of the decoder is to maximize the probability of the next possible correct word in the target language sequence. The main way of building the decoder is:

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

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

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 described as $c = h_T.u_i$ which denotes the i^{th} word from the target language sequence and u_0 denotes the beginning of the target language sequence, which indicates the beginning of the decoding. Lastly, z_0 is an all zero vector and z_i is the RNN hidden state at time step i.

2. Calculating the probability p_{i+1} for the $(i+1)$-th word in the target language sequence is described as:

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

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

3. The cost is computed according to p_{i+1} and u_{i+1}.

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

2.6.2 Training: Maximum Likelihood Estimation (MLE)

After developing the neural translation model, the model needs to be trained using parallel data. The previously described encoder-decoder model uses Maximum log-likelihood estimation (MLE)[15] which is a common statistical technique for training. Let us 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 based on the vocabulary set V, which is equivalent to a sequence of one-hot vectors. Multiplying an one-hot vector with an embedding matrix 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 can describe the log-likelihood of the whole training corpus as,

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

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

The generation process of machine translation is to process the source sentence into a sentence in the target language according to a pre-trained model. In the decoding step,

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

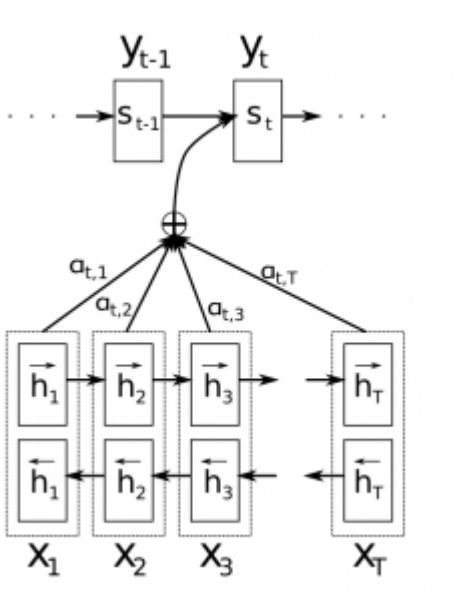

Figure 2.6: Attention Model (Bahdanau et al., 2015)

there are different strategies for generating next word in the output sequence such as greedy search and beam search.

2.6.3 Attention Mechanism

1. It does not seem reasonable to encode all the information for a sentence with a fixed dimensional vector representation regardless of the length of the sentence. In theory, algorithms like LSTMs should be able to deal with this. But in practice long-range dependency issues still occur problems due to the vanishing gradient problem[16].

2. While processing a source input sentence, the model generally pays more attention or concentration to the parts in the source sentence which is more relevant to the output translation, which is currently in the decoding stage. However, in the source sentence, the focus changes in process of the translation. With a fixed dimensional vector, all the words from the source sentence are treated equally. This is unreasonable in any circumstances. That is why, Bahdanau et al. (2015) proposed attention mechanism for the very first time in NMT (see Figure 2.6), which is able to decode based on

[16]https://www.quora.com/What-is-the-vanishing-gradient-problem

different parts of the context sequence to address the difficulty of feature learning for long sentences[17]. With an attention mechanism, it is not required anymore to encode the full source input sentence into a fixed-length vector. Instead, the model allows the decoder to attend (focus on) the different parts of the source sentence at each time step of the output generation process. 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, a distinct context vector c_i is used for processing word y_i. In short, This context vector c_i is the weighted sum of the RNN hidden states (h_j) of the encoder. The weight a_{ij} which denotes the strength of attention of the i^{th} word in the target language sentence to the j^{th} word in the source sentence.

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

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

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

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

where, *align* is an alignment model that computes the fitness between the i^{th} word in the target language sentence and the j^{th} word in the source sentence. In the conventional alignment model **hard alignment** is used, which means each word in the target language corresponds to one or more words from the target language sentence.

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

On the other hand, if any word in source input sentence is related to any word in the target language output **soft alignment** is used, where the strength of the relation or attention is a real number computed via the alignment model. It completely depends on the problem whether to use a hard or soft alignment model.

2.6.4 Greedy 1-Best Search

Greedy 1-Best output is very useful in machine translation if we simply require the best output according to the model. The greedy 1-best search, calculates probability p_t at every time step, then the word is selected which gives the highest probability (1-best), and use it to predict the next word in the sequence (Neubig, 2017). Due to local optimum, a greedy search is not guaranteed to be able to find the output with the highest probability. Considering the n-best words at each time step of the decoder can be a solution to this problem.

2.6.5 Beam Search Algorithm

Beam Search[18] is a heuristic search algorithm which explores a graph by expanding it to find the most probable node in a limited set. It is used when the probable solution is significantly large for the applications such as machine translation, speech recognition and natural language processing. It is very useful if there is not enough memory to use for considering all the possible solutions.

Using a breadth first search algorithm (BFS)[19], beam search builds a search tree and sorts the nodes according to a heuristic cost (sum of the log probability of the generated words) at each level of the tree. The beam search is almost similar to the greedy search, but instead of considering only the 1-best word, it considers b best words at each time step, where b is the width of the beam size(sometimes called beam search size). Thus, in the next level, b best nodes with highest scores are expanded. Through this process the space

[18] https://en.wikipedia.org/wiki/Beam_search
[19] https://en.wikipedia.org/wiki/Breadth-first_search

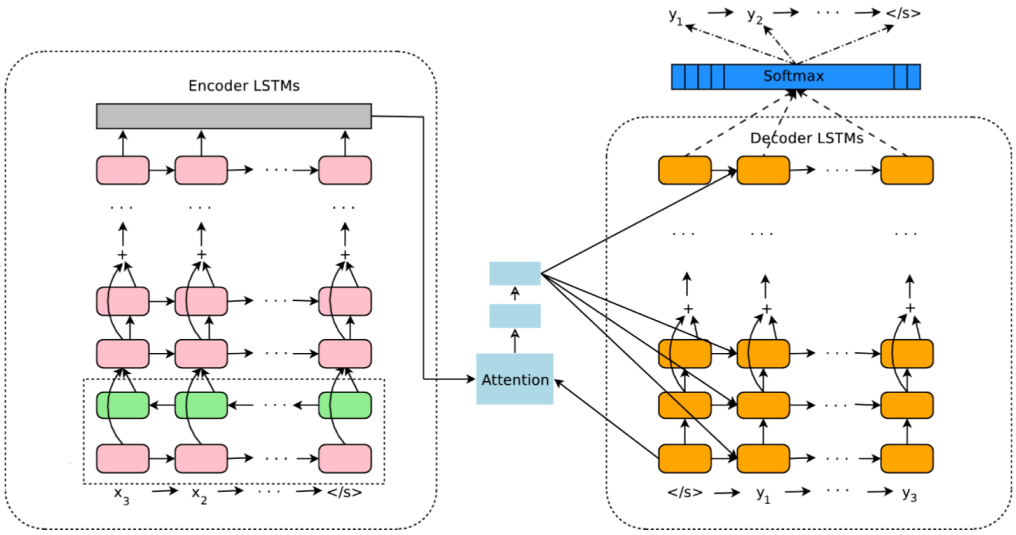

Figure 2.7: Google's Neural Machine Translation (NMT) Model (Wu et al., 2016)

and time requirements are reduced significantly. However, in case of beam search, there is also no guarantee of a global optimum solution. When decoding, if the end-of-sentence token $< eos >$ is generated the search process generally stops or the maximum length of the sentence is reached.

The Figure 2.7 is an example of the Google's recent machine translation framework which uses almost all the techniques described.

2.7 Recurrent Layers

Recurrent layers are powerful algorithms used in artificial intelligence and are especially useful for processing sequential data like written natural language. Recurrent networks based on recurrent layers are different from feed-forward networks because they include a feedback loop. For example, if a network is trained using some words letter by letter, and is asked to guess each subsequent letter, the very first letter of any word will help to determine what the recurrent network thinks the second letter and after letters will be. Although some data, like images, do not seem to be sequential, still they can be learned as sequences when fed into a recurrent network. There are several different layers such as RNN, LSTM and GRU. we will be summarizing the most useful layers next.

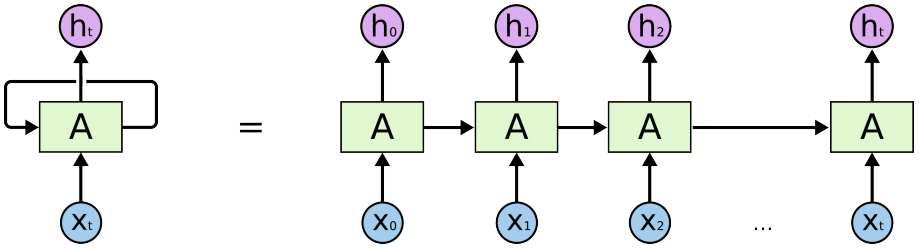

Figure 2.8: An unrolled recurrent neural network

2.7.1 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 2.8[20], 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 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:

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

2.7.2 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

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

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[21].

Long Short Term Memory networks (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 have been 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:

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

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

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

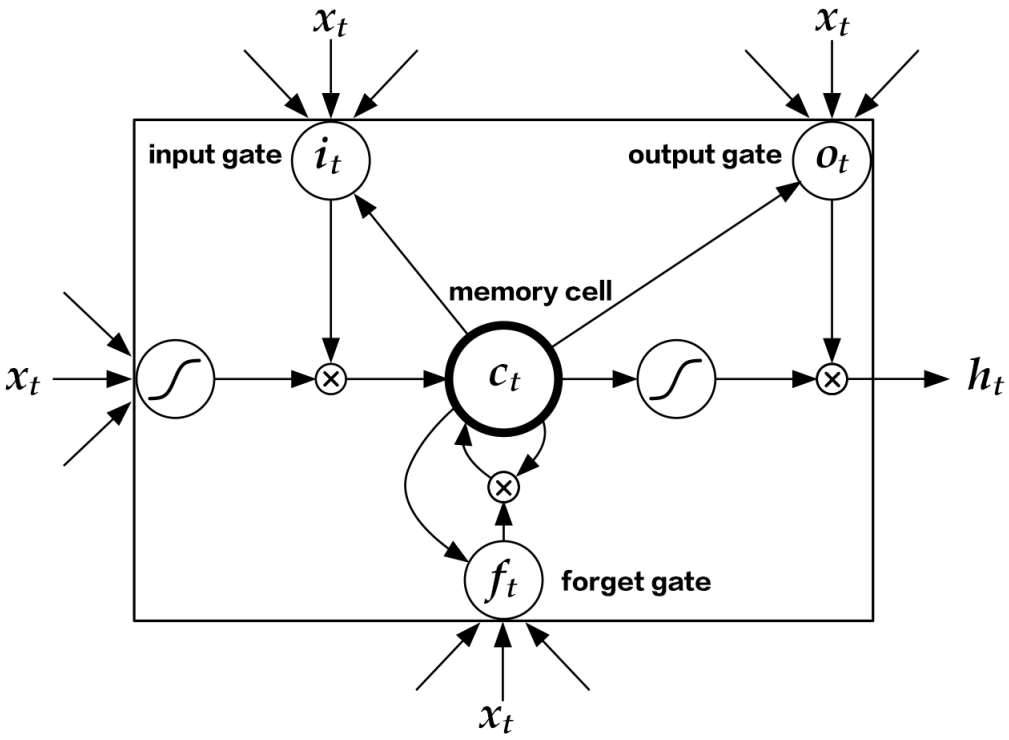

Figure 2.9: LSTM at time step t (Hochreiter and Schmidhuber, 1997)

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{h-1} + W_{ci}c_{t-1} + b_i) \tag{2.2}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{h-1} + W_{cf}c_{t-1} + b_f) \tag{2.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{h-1} + b_c) \tag{2.4}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{h-1} + W_{co}c_t + b_o) \tag{2.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.6}$$

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 2.9.

2.7.3 Gated Recurrent Unit (GRU)

GRU (Cho et al., 2014b) is related to a LSTM, but both uses a different gating mechanism to prevent the long-distance dependencies problem. GRUs are relatively new, have a less complex structure, train faster, are computationally more efficient and perform better than a LSTM on less training data (Chung et al., 2014). The 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):

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

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

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

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

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 2.10.

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.

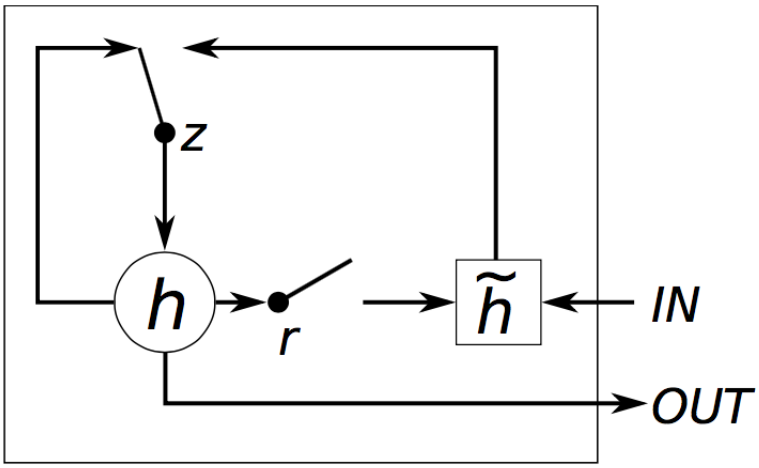

Figure 2.10: GRU Gating Mechanism (Chung et al., 2014)

2.8 Transformer

Like most NMT models, the Transformer (Vaswani et al., 2017) is also based on the popular encoder-decoder structure. The difference between the transformer and any other NMT model is that, it is entirely based on attention mechanisms and dot-products, containing fully connected layers for both the encoder and the decoder sides. The model follows the actual architecture for a standard encoder-decoder model but the most commonly used recurrent layers in encoder-decoder architectures are replaced by the mult-head self-attention. In short, this model is cheaper computationally than any other NMT models.

2.8.1 Transformer Encoder

A Transformer's encoder is composed of a stack of N = 6 identical layers. Each layer consists of two sub-layers. The first one is a multi-head self-attention mechanism, and the second layer is a simple layer which is a position-wise fully connected feed-forward network. Followed by layer normalization, a residual connection is employed around each of the two sub-layers. The output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ represents the function implemented by the sub-layer itself. To operate these residual connections properly, all sub-layers in the architecture, along with the embedding

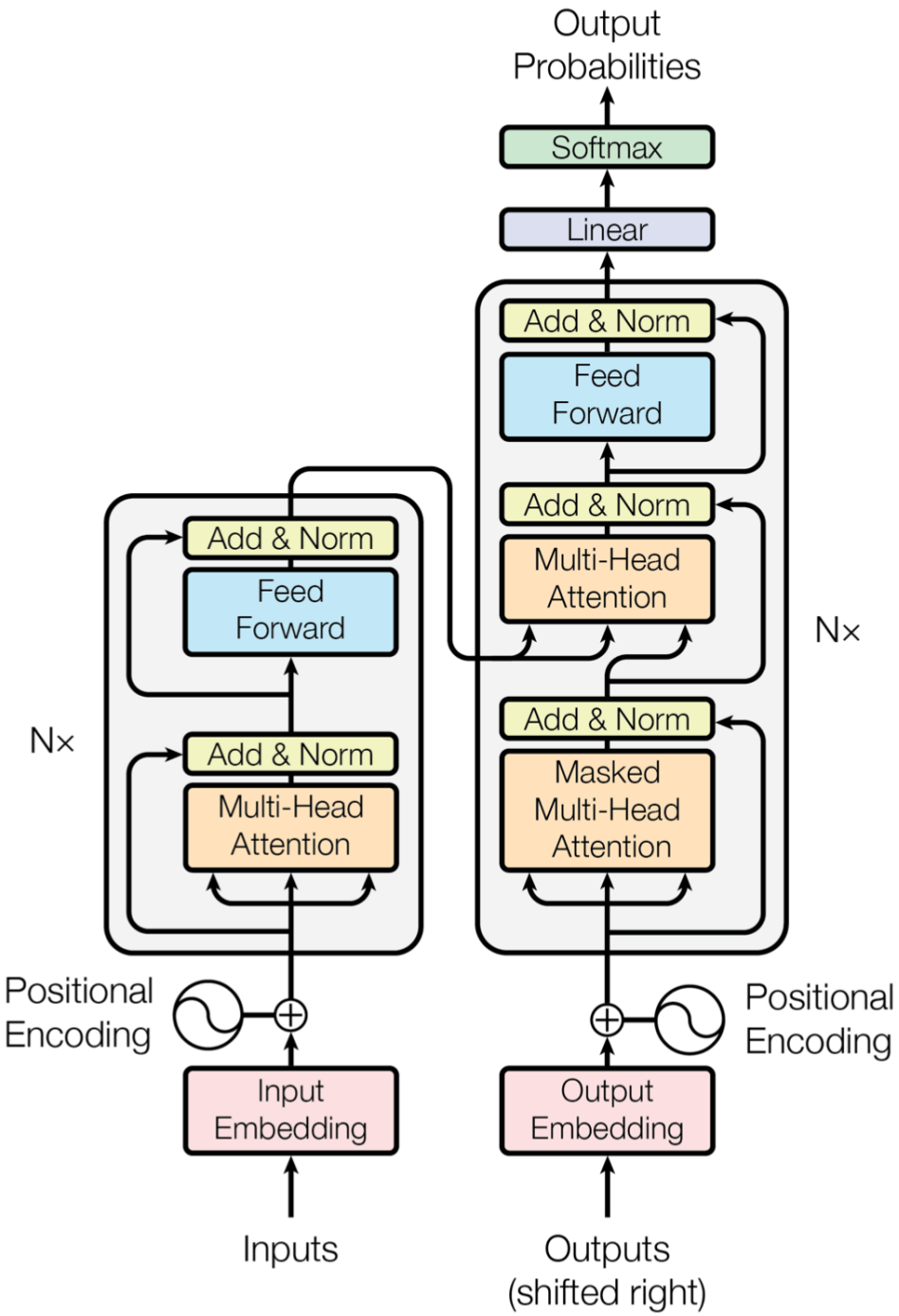

Figure 2.11: Transformer model architecture (Vaswani et al., 2017)

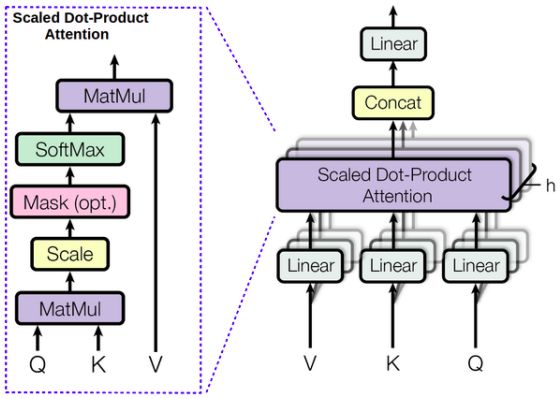

Figure 2.12: Multi-Head Attention Mechanism

layers, produce outputs of dimension $d_{model} = 512$.

2.8.2 Transformer Decoder

Like the encoder, the decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder adds a third sub-layer, which executes a multi-head attention mechanism over the output of the encoder side. Like the encoder, followed by layer normalization, residual connections are employed around each of the sub-layers. The self-attention sub-layer in the decoder stack is also modified to prevent positions from attending to subsequent positions.

2.8.3 Positional Encoding

Since the transformer model contains no recurrence or no convolution, to ensure the use of the order of the sequence by the model, we require some additional information about the relative or absolute position of the tokens in the sequence. To perform this, positional encodings are added to the input embeddings at the end of the encoder and decoder stacks. The positional encodings holds the same dimension d_{model} as the embeddings, as a result the two can be summed.

2.8.4 Multi-Head Attention

Instead of performing a single attention function with d_{model} dimensional keys, values and queries, it has been found beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_q, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values the attention function is then performed in parallel, yielding dv-dimensional output values. These are concatenated, resulting in the final values. Recently, multi-head attention[22] based models are gaining popularity among the researchers of Natural Language Processing (NLP).

2.9 Summary

Through the chapter, the necessary background information is presented and recent related works in research are discussed. As a background, solid understanding of the terms i.e. summary evaluation, word embedding, Recurrent Neural Network (RNN), Neural Machine Translation (NMT), tensor2tensor, encoder decoder framework, beam search decoder and transformer is necessary, as the proposed method is heavily depended on these concepts. From the perspective of a computational linguists, this chapter explains this terms along with necessary details.

[22]Figure 2.12 taken from https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/

Chapter 3

Sentence Clustering

Multi document summarization has been a popular problem in Natural Language Processing (NLP) field. To solve this problem, our proposed method is given in Figure 3.1. At the very beginning, all the similar documents are merged in order to solve multi-doc summarization. Then the merged document is clustered. In this chapter, the clustering process is explained. Other processes are explained in the following chapters.

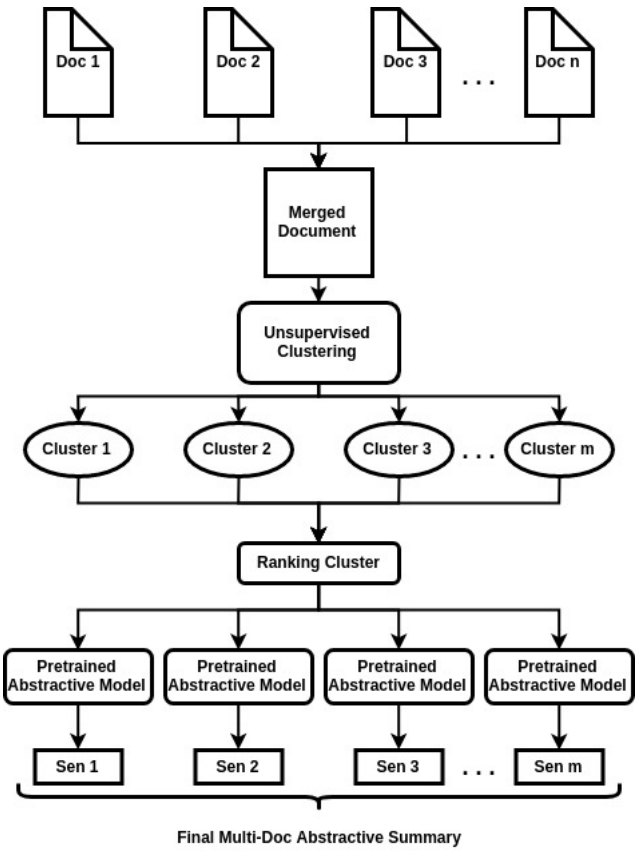

Figure 3.1: Proposed method to solve Multi-Doc Abstractive Summarization

3.1 Text Clustering

Text clustering is a challenging problem due to its sparseness of text representation as most words only occur once in a text (Aggarwal and Zhai, 2012). As a result, the Term Frequency-Inverse Document Frequency (TF-IDF) measure will not work well. In order to address this problem, we use word embedding and deep neural network architectures for better representation of text and hence propose an unsupervised sentence clustering model. Extensive experimental results demonstrate that the proposed model is simple, effective, and outperforms several popular clustering methods when tested on two public datasets.

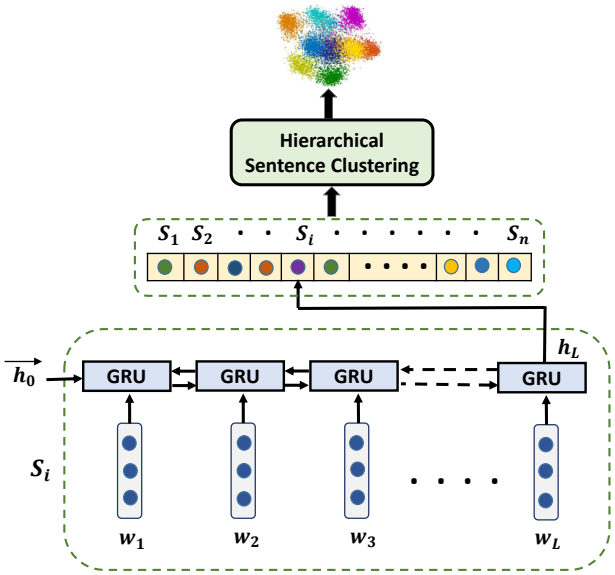

Figure 3.2: Sentence Clustering Model

3.2 Model

3.2.1 Sentence Embedding

A sentence is a sequence of words $\mathbf{S} = (w_1, w_2, ..., w_L)$, where L is the length of the sentence. We chose to encode a sentence using bi-directional GRUs (Cho et al., 2014a). The GRU (Cho et al., 2014a) achieves similar performance as an LSTM (Hochreiter and Schmidhuber, 1997) but it is faster, computationally efficient 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 using:

$$h_t = \mathbf{GRU}(h_{t-1}, e(w_t)) \tag{3.1}$$

where, $h_t \in \mathbb{R}^n$ encodes all content seen so far at time t which is computed from h_{t-1} and $e(w_t)$. Here, $e(w_t) \in \mathbb{R}^m$ is the m-dimensional embedding of the current word w_t. We can use any pre-trained word vectors as input to the GRUs.

In our work, we apply bi-directional GRUs (bi-GRUs), which we found achieve better results consistently than single directional GRUs. As shown in Figure 3.2, Bi-GRU processes the input sentence in both forward and backward direction with two separate hidden layers calculated with GRUs. It obtains the forward hidden states ($\overrightarrow{h_1}, \ldots, \overrightarrow{h_L}$) and the backward hidden states ($\overleftarrow{h_1}, \ldots, \overleftarrow{h_L}$). For each position t, we simply concatenate both the forward and backward states into the final hidden state:

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{3.2}$$

where, the operator $\oplus$ indicates concatenation. $\overrightarrow{h_t}$ is calculated using equation 3.2 and $\overleftarrow{h_t}$ is calculated using the following equation.

$$\overleftarrow{h_t} = \mathbf{GRU}(\overleftarrow{h_{t+1}}, e(w_t)) \tag{3.3}$$

where, $\overrightarrow{h_0}$ is initialized as a zero vector, and the output sentence embedding x_i for the sentence S_i is the last hidden state:

$$S_i = x_i = h_L \tag{3.4}$$

Inspired from Murtagh and Legendre (2014), we use a hierarchical clustering algorithm

with a complete linkage criteria. This algorithm proceeds incrementally, starting with each sentence considered as a cluster, and merging pairs of similar clusters after each step using a bottom up approach. The complete linkage criteria determines the metric used for the merge strategy, which calculates largest distance between a sentence in one cluster and a sentence in the other candidate cluster. In building the clusters, we use the cosine similarity between the sentence embeddings obtained from equation 3.4. We set a similarity threshold ($\tau = 0.5$) to stop the clustering process by using a hold out dataset SICK[23] of SemEval-2014 (Marelli et al., 2014) for getting optimal performance. If we cannot find any cluster pair with a similarity above the threshold ($\tau = 0.5$), the process stops, and the clusters are released.

3.3 Experiments

3.3.1 Datasets

In our work, following datasets have been used.

- **StackOverflow**[24] We use the challenge data published in Kaggle.com[25]. This dataset consists of 3,370,528 samples from July 31st, 2012 to August 14, 2012. In our experiments, we randomly select 20,000 question titles from 20 different tags.

- **SearchSnippets**[26] This dataset was constructed from the different predefined phrases of web search transaction results of 8 different domains (Phan et al., 2008).

3.3.2 Pre-trained Word Vectors

The word embeddings are low dimensional vector representations of words such as **word2vec** (Mikolov et al., 2013b) and **GloVe** (Pennington et al., 2014) which recently gained much attention in various natural language processing tasks. Recently, Bojanowski

[23]http://clic.cimec.unitn.it/composes/sick.html
[24]https://github.com/jacoxu/StackOverflow
[25]https://www.kaggle.com/c/predict-closed-questions-onstack- overflow/download/train.zip
[26]http://jwebpro.sourceforge.net/data-web-snippets.tar.gz

et al. (2017) propose a simple method named **fastText** to learn word representations by taking into account sub-word information. All these are already explained in Chapter 2.

3.3.3 Experiments

We conduct extensive experiments for our model (**HierGRU**) on two public datasets with these word embeddings which is presented in Table 3.1. We evaluate the performance using **Homogeneity** (each cluster contains only members of a single class) and **Completeness** (all members of a given class are assigned to the same cluster) from Rosenberg and Hirschberg (2007). As seen from Table 3.1, **fastText** performs very well when the number of clusters are large compare to other embeddings.

3.3.4 Baselines

Following baselines have been considered to compare our model's performance:

- **K-means** (Wagstaff et al., 2001) on original keyword features which are weighted with Term Frequency-Inverse Document Frequency (TF-IDF).
- **Spectral Clustering** (Belkin and Niyogi, 2001) uses Laplacian Eigenmaps (LE) and subsequently employ K-means algorithm on weighted Term Frequency (TF) of a word in a sentence.
- **Average Embedding:** We take the pre-trained word embeddings (Bojanowski et al., 2017) of all the non stopwords in a sentence and take the weighted vector average according to the term-frequency (TF) of a word in a sentence then run K-means on it.
- **STCC** (Xu et al., 2015) integrate the ability of convolutional filters to capture local features for high-quality text representation into a self-taught learning framework (Zhang et al., 2010) to cluster short texts.
- **STC-2** (Xu et al., 2017) incorporate some semantic features and learn non-biased deep text representation in an unsupervised manner using self-taught Convolutional

Table 3.1: Results of Homogeneity and Completeness with different pre-trained word embeddings.

Word Embedding	**StackOverflow**		**SearchSnippets**	
	Homogeneity (%)	**Completeness (%)**	**Homogeneity (%)**	**Completeness (%)**
Word2Vec	26.3	26.3	57.7	58.7
GloVe	44.1	47.0	**62.3**	**58.8**
fastText	**66.6**	**70.0**	57.0	57.6

Table 3.2: Comparison of ACC and NMI of clustering methods on two public datasets.

Method	**StackOverflow**		**SearchSnippets**	
	ACC (%)	**NMI (%)**	**ACC (%)**	**NMI (%)**
K-means (Wagstaff et al., 2001)	20.31	15.64	33.77	21.40
Spectral Clustering (Belkin and Niyogi, 2001)	27.55	21.03	63.90	48.44
Average Embedding	37.22	38.43	64.63	50.59
STCC (Xu et al., 2015)	51.13	49.03	77.09	63.16
STC-2 (Xu et al., 2017)	51.20	49.09	77.08	62.99
HierGRU + GloVe	54.5	48.8	81.0	60.3
HierGRU + fastText	**81.2**	**65.0**	**82.5**	**64.7**

Neural Networks (CNN).

3.3.5 Results

We present our experimental results compared to different simple and state-of-the-art baselines in Table 3.2. We evaluate clustering performance using the accuracy (**ACC**) and the normalized mutual information metric (**NMI**) (Cai et al., 2005). According to Table 3.2, our model achieves the best clustering performance on all the metrics for both datasets using **fastText** word embeddings.

3.4 Summary

Our clustering method has been explained in this chapter along with state-of-the art results. To achieve this result we have proposed a novel method. In the following chapter our proposed neural fusion model is discussed.

Chapter 4

Neural Sentence Fusion

Multi-sentence compression (**MSC**) usually takes a group of related sentences and produces an output sentence through the merging of sentences about the same topic. MSC is a text-to-text generation process in which a novel sentence is produced as a result of summarizing a set of similar sentences, this process was originally called sentence fusion (Barzilay and McKeown, 2005). The recent success of neural sequence-to-sequence (seq2seq) models provide an effective way for text generation. This achieved huge success in the case of abstractive sentence summarization which can perform deletion based compression from a single source sentence (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; Suzuki and Nagata, 2017; Zhou et al., 2017; Ma et al., 2017). Moreover, there are some recent attempts which uses the CNN/Daily Mail corpus (Hermann et al., 2015) as a supervised training data to generate multi-sentence summary from a single document using neural architectures (See et al., 2017; Li et al., 2017b; Paulus et al., 2017; Fan et al., 2017). In this work, we investigate applying the **seq2seq** encoder-decoder models to the MSC task. Our task is completely different from them, our model takes a related ordered set of sentences and produces an output sentence by fusing or merging the input sentences instead of encoding a single sentence or a document. To the best of our knowledge, our work is the first work to investigate adapting a neural encoder-decoder models to the sentence fusion task.

Table 4.1: Performance of different systems compare to our proposed Neural Sentence Fusion (**NeuFuse**) model.

Dataset	Models	BLEU	METEOR	CR	Copy Rate	GMS	EACS
SFC	Filippova (2010)	42.07	34.10	57.57	99.84	84.3	88.94
	Boudin and Morin (2013)	44.64	35.12	37.95	100	80.0	86.79
	NeuFuse_sent (ours)	**61.39**	**38.49**	**66.93**	**90.30**	**90.37**	**92.81**
MSR-ATC	Filippova (2010)	40.95	35.91	67.04	99.91	85.31	88.47
	Boudin and Morin (2013)	43.74	36.62	41.00	100	82.15	90.76
	NeuFuse_sent (ours)	**52.49**	**37.48**	**69.96**	**86.28**	**89.67**	**93.97**

4.1 Preliminaries

Given a related set of source sentences about a same topic $\mathbf{X} = (X_1, X_2, ..., X_N)$, our model learns to predict its abstractive multi-sentence compression target $Y = (y_1, y_2, ..., y_M)$, where $N > 1$ and $M < |X_1| + |X_2| + ... + |X_N|$. In this work, we use the **Transformer** model (Vaswani et al., 2017) which has shown significant improvements over state-of-the-art models for a wide variety of applications, such as machine translation, parsing, and image captioning. The **Transformer** follows the overall architecture for a standard encoder-decoder model, replacing the complex recurrent or convolutional layers most commonly used in encoder-decoder architectures with multi-headed self-attention. The natural ability of a multi-head attention mechanism to jointly attend to similar phrases from different positions of a sequence makes this an ideal choice for our model. We use the implementation provided by the authors[27]. We keep the exact same settings which was suggested for summarization.

4.2 Datasets

Training Set: Neural **seq2seq** encoder-decoder models are usually trained with lots of human-generated references. However, there are very few gold reference available for the multi-sentence compression task, such as those provided by McKeown et al. (2010) and Toutanova et al. (2016), both of which are largely insufficient for training our Neural Sentence Fusion model. Therefore, we use the CNN/DailyMail corpus (Hermann et al., 2015)

[27] https://github.com/tensorflow/tensor2tensor

Table 4.2: Training dataset statistics.

Dataset	Total Generated Sample	Average Source Length	Average Target Length	Average Source to Target ratio
CNN-DailyMail	680367	23.25	12.71	3.05

to automatically construct our training set. It has been extensively used as supervised training data to generate a multi-sentence summary from a single document (See et al., 2017; Li et al., 2017b; Paulus et al., 2017; Narayan et al., 2018a,b; Fan et al., 2017; Celikyilmaz et al., 2018). The CNN/DailyMail dataset (Hermann et al., 2015) contains almost 312K documents, each with 3-4 highlight sentences that summarize the contents of the article. We take each highlight sentence and map it with the document sentences using word overlap based Jaccard Similarity. We set a similarity threshold ($t = 0.25$) by using a hold out dataset SICK[28] of SemEval-2014 (Marelli et al., 2014). We take only the many-to-one mappings which involves multiple source source sentences from a document and filter out the rest. Our resulting training set contains 680,367 pairs of multiple source sentence to one target sentence pairs. Table 4.2 shows in-detail the statistics of the generated training data from the CNN-DailyMail corpus (Hermann et al., 2015) .

SFC Test Set: We use 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. We filtered the sentences which have no main verbs. The resulting set contains 296 pairs of sentences.

MSR-ATC Test Set: Toutanova et al. (2016) introduced a manually-created, multi-reference dataset for abstractive sentence and short paragraph compression. It contains approximately 6,000 source texts with multiple references accompanied by up to five crowd-sourced rewrites. We filtered out the pairs which contain only single source sentence. We obtained 2,405 multiple source sentence pairs with five human reference variations for our testing.

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

4.3 Baselines

Most of the previous MSC approaches rely on the syntactic parsing to build the dependency tree for each related sentence in a cluster for producing grammatical compressions (Filippova and Strube, 2008). Unfortunately, syntactic parsers are not available for every language. As an alternative, word graph-based approaches that only require a 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. Hence, compressed sentences are generated by finding the k-shortest paths in the word graph. Boudin and Morin (2013) improved Filippova (2010)'s approach by re-ranking the fusion candidate paths according to keyphrases. However, they reported that the generated sentences were missing important information and were not perfectly grammatical. With the exceptions of Filippova (2010) and Boudin and Morin (2013), we did not find any recent competitive baseline for this specific task to compare with our model.

4.4 Evaluation Metric

We evaluate our system automatically using various automatic metrics as described below.

BLEU (Papineni et al., 2002) is the most commonly used metric for the Machine Translation evaluation. BLEU relies on exact matching of *n*-grams and has no concept of synonymy or paraphrasing. We used the implementation provided in NLTK[29] considering up to 4-gram matching.

METEOR (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 look-up table paraphrases.

Compression Ratio (CR) 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

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

uncompressed.

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

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

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

Furthermore, we also use the Embedding Average Cosine Similarity (**EACS**) and the Greedy Matching Score (**GMS**)[30] from Sharma et al. (2017) to measure the abstractiveness of our generated outputs which have a stronger correlation with human reference.

4.5 Experimental Results

We report the performance of our system when compared with the baselines in terms of different evaluation metrics in Table 4.1. Our model jointly improves the information coverage (BLEU, GMS) and complete abstractiveness (METEOR, Copy Rate, EACS) with a balanced compression ratio(CR). The copy Rate scores of other baseline systems clearly indicate the fact that they are performing completely deletion-based compression with no new words or words with morphological variation being generated in the process. We present some randomly selected outputs generated by our model for both the datasets in Appendix B : Supplemental Material.

[30]https://github.com/Maluuba/nlg-eval

4.6 Summary

Our proposed neural sentence fusion method has been explained in this chapter. This method also achieved state-of-the art results. In the next chapter multi-document summarization method is explained. Since our proposed method is a hierarchical method, in the next chapter we show the procedure to combine the clustering and fusion method to solve multi-doc summarization.

Chapter 5

Abstractive Multi-Document Summarization

We use our sentence clustering technique to group related sentences from the document set on a given topic. We then order the clusters and the sentences inside the clusters using a heuristic sentence ordering technique. For each cluster of related ordered sentences, we use our neural sentence fusion model to generate fused abstractive versions of the multiple related sentences extracted form the document set. Finally, we use our ILP based abstractive sentence selection mechanism to select the best subset of sentences which simultaneously considers importance, coverage and diversity under a desired length limit. The overall process is presented in this chapter.

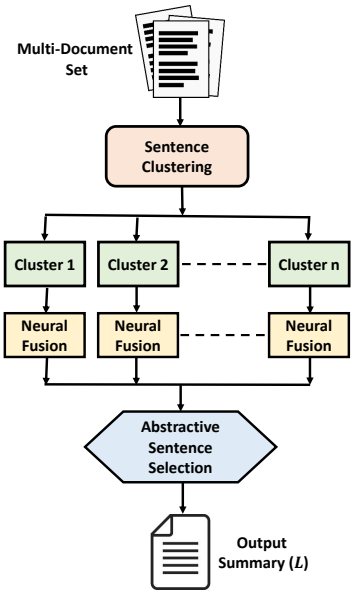

Figure 5.1: Proposed final method to solve Multi-Document Summarization Model

5.1 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 can 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 from the original document. In multi-document summarization, we can not directly use the sentence position as the sentences are coming from multiple documents. Therefore we implement two cluster ordering techniques that reorder clusters based on the original position of the sentences in the documents.

5.1.1 Intra-Cluster Ordering

The sentences $\{S_1, S_2, .., S_i, .., S_n\}$ in any cluster C_i are assigned a normalized score. For example, the normalized score of S_i is computed as the ratio of the original position of the sentence to the total number of sentences in document D_i (here, S_i belongs to document D_i). We then pass this ordered related set of sentences to our neural sentence fusion model.

5.1.2 Inter-Cluster Ordering

When ordering two different clusters, the cluster that has the lower score obtained by averaging the normalized scores of all the sentences in that particular cluster is ranked higher than the others.

5.2 Abstractive Sentence Selection

In this work, we use the concept-based ILP framework introduced by Gillick and Favre (2009) with some suitable changes to select the best subset of sentences. This approach aims to select sentences that cover as many important concepts as possible, while ensuring the summary length is within a given budgeted constraint. We propose an ILP based sentence selection mechanism which integrates three important measures namely importance,

coverage, and diversity to extract the sentences for the summary under a certain length limit.

5.2.1 Importance

One of the basic requirements of a good summary is that it should contain the most important information across multiple documents. To model this property, we use bi-grams as concepts. Bi-grams are the phrases that represent the main topics of a document. Sentences containing the most relevant phrases are important for the summary generation. We assign a weight to each bi-gram using its document frequency. Bi-grams consisting of two stop-words or one punctuation mark are pruned. Let w_i be the weight of bi-gram i and b_i a binary variable that indicates the presence of bi-gram i in the extracted sentences. We try to maximize the weight of the bi-grams in all the selected sentences for summary generation as follows,

$$S_{imp} = \sum_i w_i b_i \tag{5.1}$$

5.2.2 Coverage

A good summary has the capability to cover most of the important aspects of a document set. To formulate this, we select at most one sentence from the cluster of related sentences to increase the information coverage from the document side. In order to ensure at most one sentence per cluster in the extracted sentences we add an extra constraint in our overall ILP formulation using the following equation, where g_c is a cluster of sentences that corresponds to the set of similar sentences, S_j:

$$\sum_{j \in g_c} s_j \leq 1, \quad \forall g_c \tag{5.2}$$

5.2.3 Diversity

Maximizing diversity in the summary is another basic requirement in any summarization task. We define the degree of diversity of a generated summary by measuring the dissimilarity among the selected sentences. Let the generated summary be Y and $|Y|$ is the total number of sentences in the summary. We compute S_{div} as the mean of the pairwise dissimilarities among the selected sentences.

$$S_{div} = \frac{1}{|Y|\ (|Y|-1)} \sum_{i \in Y} \sum_{j \in Y} d(S_i, S_j) \tag{5.3}$$

where $d(.\ ,.)$ is the dissimilarity function calculated by

$$d\ (S_i, S_j) = 1 - \frac{S_i \cdot S_j}{||S_i||\ ||S_j||} \tag{5.4}$$

Intuitively, the more diverse (or more dissimilar) the selected sentences to each other, the higher the diversity. The right part of the equation is simply the $1 - cosineSimilarity(S_i, S_j)$.

5.2.4 Summary Length Limit

One of the essential properties of text summarization systems is the ability to generate a summary with a fixed length, which has a common commercial use case (e.g., 160 to 300 characters for search result and news article summarization by news aggregators, especially on mobile devices). All the recent models for document summarization either extractive or abstractive do not consider this issue at all in the case of multi-document summarization. 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, where models generate a single sentence headline of a document.

Very recently, Fan et al. (2017) presented a neural model that enables users to specify a desired length in order to control the shape of the final summaries which is only limited to single document summarization. In this thesis, we address this issue in multi-document setting, our model can generate summaries given a desired length.

Finally, we propose an ILP formulation which considers the above mentioned aspects in the context of multi-document summarization. The ILP problem is then solved exactly using an off-the-shelf ILP solver[31]. The final summaries are generated by assembling the optimally selected sentences. Let l_j be the number of words in sentence j, s_j be a binary variable that indicates the presence of sentence j in the extracted sentence set and L be the length limit for the summary. Let Occ_{ij} indicate the occurrence of bi-gram i in sentence j. The final ILP formulation is:

$$Maximize : S_{imp} + S_{div} \tag{5.5}$$

$$Subject\ to : \sum_j l_j s_j \leq L \tag{5.6}$$

$$s_j Occ_{ij} \leq b_i, \quad \forall i, j \tag{5.7}$$

$$\sum_j s_j Occ_{ij} \geq b_i, \quad \forall i \tag{5.8}$$

$$\sum_{j \in g_c} s_j \leq 1, \quad \forall g_c \tag{5.9}$$

$$b_i \in \{0,1\} \quad \forall i \tag{5.10}$$

[31] We use Gurobi, http://www.gurobi.com

$$s_j \in \{0,1\} \quad \forall j \tag{5.11}$$

We try to maximize the importance score as well as the diversity in the output summary sentences, while avoiding repetition of those bi-grams and staying under the maximum number of words allowed for the summary. We select at most one sentence from the cluster of related sentences to increase the information coverage from the document point of view. In this process, we extract the optimal combination of sentences as the output summary.

5.3 Multi-Document Level Experiments

In this section our multi-document level experiments have been discussed.

5.3.1 Dataset

We consider the generic multi-document summarization dataset provided from the Document Understanding Conference (DUC 2004)[32] which is one of the main benchmark dataset in the multi-document summarization containing 50 document clusters. The Opinosis (Ganesan et al., 2010) is another dataset consisting of short user reviews in 51 different topics collected from TripAdvisor, Amazon, and Edmunds.

5.3.2 Evaluation Metric

We evaluate our summarization system using **ROUGE**[33] (Lin, 2004) on DUC 2004 (Task-2, Length limit (L) = 100 Words) and Opinosis 1.0 (L = 15 Words). However, ROUGE scores are unfairly biased towards lexical overlap at the surface level. Taking this into account, we also evaluate our system with a recently proposed metric ROUGE-SU4. We report limited length recall performance for both the metrics, as our system generated summaries are forced to be concise through some constraints (such as a length limit constraint). Therefore, we considered using just the recall since precision is of less concern in

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

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

this scenario. We perform ablation experiments with our model having only Extract with a desired limit and our extract, abstract and select framework with a bigger lenth limit on the extractive side.

5.3.3 Baseline Systems

The summaries generated by the baseline **LexRank** (Erkan and Radev, 2004) and the state-of-the-art extractive summarizers **Submodular** (Lin and Bilmes, 2011) and **RegSum** (Hong and Nenkova, 2014) on the DUC 2004 dataset were collected from Hong et al. (2014). In the case of **ILPSumm**[34] (Banerjee et al., 2015) and **PDG*** (Yasunaga et al., 2017), we use the author provided implementation to generate a summary from their model. For the Opinosis 1.0 dataset, we use an open source implementation of **TextRank**(Mihalcea and Tarau, 2004)[35]. Moreover, we use the author provided implementation for the **Opinosis** (Ganesan et al., 2010) and **Biclique** (Muhammad et al., 2016) to generate summaries.

5.3.4 Results

According to the Table 5.1 & 5.2, our multi-document level model achieves the best summarization performance on all the ROUGE metrics for both the datasets. However, the ROUGE scores are unfairly biased towards lexical overlap at the surface level. Therefore, we are unable to measure the abstractiveness property. Taking this into account, we use the document level **EACS** (Sharma et al., 2017) which considers word embeddings to compute the semantic similarity of the words. Moreover, we verify the copy rate scores of the human summary and our system generated summary with the source documents. According to Table 5.3, our system generated summary is very close to human references in terms of both **EACS** and **Copy Rate** scores.

[34]https://github.com/StevenLOL/AbTextSumm
[35]https://github.com/davidadamojr/TextRank

Table 5.1: Comparison results on the **DUC 2004** test set.

DUC 2004			
Models	**R-1**	**R-2**	**R-SU4**
LexRank (2004)	35.95	7.47	12.48
Submodular (2011)	39.18	9.35	14.22
RegSum (2014)	38.57	9.75	13.81
ILPSumm (2015)	39.24	11.99	14.76
PDG* (2017)	38.45	9.48	13.72
NAMDS (2018)	36.7	7.83	12.4
NeuFuse_multidoc (ours)	**41.92**	**12.22**	**15.59**

Table 5.2: Comparison results on the **Opinosis 1.0** test set.

Opinosis 1.0			
Models	**R-1**	**R-2**	**R-SU4**
TextRank (2004)	27.56	6.12	10.53
Opinosis (2010)	32.35	9.13	14.35
Biclique (2016)	33.03	8.96	14.18
NeuFuse_multidoc (ours)	**43.98**	**17.31**	**22.19**

Table 5.3: Copy rate found in different data set.

Dataset	**Copy Rate**		**EACS**
	Human Summary	**System Summary**	
DUC 2004	76.22	88.01	95.46
Opinosis 1.0	69.58	70.48	88.28

5.4 Summary

In this chapter, our final approach is explained, which is a combination of previous chapters. Also in this chapter, a data diversity driven sentence selection approach has been explained which is also within limited length. Finally, this chapter concludes with explaining our approach to solve multi-document summarization method. Our system achieves state-of-the-art result. In the next chapter, overall work and future direction of this work are discussed.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

We have developed a hierarchical approach in this thesis to solve multi-document summarization. In our approach we have developed techniques to solve both clustering and sentence fusion. The combined operation of these techniques achieve a state-of-the-art result. We implemented an ILP-based sentence selection along with our own ranking algorithm for abstractive multi-document summarization. We have conducted both sentence level and document level experiments in which competitive results are achieved. For sentence level tasks, our approach has been applied to several datasets and compared with several newly proposed methods. We have also evaluated our approach at the document level and for that the Document Understanding Conference (DUC) 2004 datasets along with ROUGE evaluation are used. We have also conducted our experiment in with the Opinosis 1.0 dataset. Our experiments demonstrate that our approach have achieved significant improvements over several latest best performances.

6.2 Future Work

Though the results we obtained have already shown the effectiveness of the proposed hierarchical approach, it could be further improved in a number of ways:

- Our clustering approach is completely unsupervised. This could also be done in supervised way using tensor2tensor.

- Our encoder-decoder model can be modified to encode a full document or to some extent a document set with our model.
- Our Neural Sentence Fusion model could be further modified to document level so that a single model performs both fusion and summarization.
- Our proposed model can be extended to produce a summary for different groups of people.

References

Charu C. Aggarwal and ChengXiang Zhai. 2012. *A Survey of Text Clustering Algorithms*, Springer US, Boston, MA, pages 77–128. https://doi.org/10.1007/978-1-4614-3223-4$_4$.

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

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Association for Computational Linguistics, pages 65–72. http://www.aclweb.org/anthology/W05-0909.

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

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. MIT Press, Cambridge, MA, USA, NIPS'01, pages 585–591. http://dl.acm.org/citation.cfm?id=2980539.2980616.

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.

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.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146. https://transacl.org/ojs/index.php/tacl/article/view/999.

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.

Deng Cai, Xiaofei He, and Jiawei Han. 2005. Document clustering using locality preserving indexing. *IEEE Trans. on Knowl. and Data Eng.* 17(12):1624–1637. https://doi.org/10.1109/TKDE.2005.198.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep Communicating Agents for Abstractive Summarization. In *Proceedings of the NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Yllias Chali, Moin Tanvee, and Mir Tafseer Nayeem. 2017. Towards abstractive multidocument summarization using submodular function-based framework, sentence compression and merging. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*. pages 418–424. https://aclanthology.info/papers/I17-2071/i17-2071.

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.

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.

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.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217* .

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 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.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*. volume 2.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 340–348. http://dl.acm.org/citation.cfm?id=1873781.1873820.

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.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1693–1701. http://dl.acm.org/citation.cfm?id=2969239.2969428.

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.

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.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 3294–3302. http://dl.acm.org/citation.cfm?id=2969442.2969607.

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.

Piji Li, Wai Lam, Lidong Bing, Weiwei Guo, and Hang Li. 2017a. Cascaded attention based unsupervised information distillation for compressive summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2081–2090. https://www.aclweb.org/anthology/D17-1221.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017b. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2091–2100. https://www.aclweb.org/anthology/D17-1222.

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.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 635–640. http://aclweb.org/anthology/P17-2100.

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, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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.

Azam Sheikh Muhammad, Peter Damaschke, and Olof Mogren. 2016. Summarizing online user reviews using bicliques. In *Proceedings of the 42Nd International Conference on SOFSEM 2016: Theory and Practice of Computer Science - Volume 9587*. Springer-Verlag New York, Inc., New York, NY, USA, pages 569–579. https://doi.org/10.1007/978-3-662-49192-$8_4$6.

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.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Ranking Sentences for Extractive Summarization with Reinforcement Learning. In *Proceedings of the NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Shashi Narayan, Nikos Papasarantopoulos, Shay B. Cohen, and Mirella Lapata. 2018b. Neural extractive summarization with side information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Mir Tafseer Nayeem and Yllias Chali. 2017a. Extract with order for coherent multi-document summarization. In *Proceedings of TextGraphs@ACL 2017: the 11th Workshop on Graph-based Methods for Natural Language Processing, Vancouver, Canada, August 3, 2017*. pages 51–56. https://aclanthology.info/papers/W17-2407/w17-2407.

Mir Tafseer Nayeem and Yllias Chali. 2017b. Paraphrastic fusion for abstractive multi-sentence compression generation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. pages 2223–2226. https://doi.org/10.1145/3132847.3133106.

Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. 2018. Abstractive unsupervised multi-document summarization using paraphrastic sentence fusion. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 1191–1204. http://aclweb.org/anthology/C18-1102.

Mir Tafseer Nayeem et al. 2017. *Methods of sentence extraction, abstraction and ordering for automatic text summarization*. Master's thesis, Lethbridge, Alta.: Universtiy of Lethbridge, Department of Mathematics and Computer Science.

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

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.

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.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '08, pages 91–100. https://doi.org/10.1145/1367497.1367510.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. http://www.aclweb.org/anthology/D07-1043.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 157–162. http://dl.acm.org/citation.cfm?id=2390384.2390403.

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.

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.

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.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799* https://www.microsoft.com/en-us/research/publication/relevance-unsupervised-metrics-task-oriented-dialogue-evaluating-natural-language-generation/.

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.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 577–584. http://dl.acm.org/citation.cfm?id=645530.655669.

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.

jiaming Xu, peng wang, guanhua tian, bo xu, jun zhao, fangyuan wang, and hongwei hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Association for Computational Linguistics, pages 62–69. https://doi.org/10.3115/v1/W15-1509.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, Jun Zhao, and Bo Xu. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks* 88:22 – 31. https://doi.org/https://doi.org/10.1016/j.neunet.2016.12.008.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 452–462. http://aclweb.org/anthology/K17-1045.

Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '10, pages 18–25. https://doi.org/10.1145/1835449.1835455.

Jianmin Zhang, Jiwei Tan, and Xiaojun Wan. 2018. Towards a neural network approach to abstractive multi-document summarization. *CoRR* abs/1804.09010.

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.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Washington, DC, USA, ICCV '15, pages 19–27. https://doi.org/10.1109/ICCV.2015.11.

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 summaries

Table B.1: Randomly selected outputs for our **NeuFuse** model form **MSR-ATC** dataset (Toutanova et al., 2016). Green Shading intensity represents new word generation other than source input sentence words and Yellow Shading intensity represents the morphological variation generation from the source input sentence words.

MSR-ATC	
Input Sentences	Thank you for requesting material from the American Association of Kidney Patients. We hope you find the enclosed material beneficial.
Reference (*best*)	Thanks for requesting American Association of Kidney Patients materials. We hope it is beneficial.
System Output	The American Association of Kidney Patients aim to use the enclosed material.
Input Sentences	Will the administration live up to its environmental promises ? Can we save the last of our ancient forests from the chainsaw ?
Reference (*best*)	Will the administration live up to its environmental promises to save our ancient forests?
System Output	Officials could save the last of our ancient forests from the chainsaw.

Table B.2: Randomly selected outputs for our **NeuFuse** model form **SFC** dataset (McKeown et al., 2010). Green Shading intensity represents new word generation other than source input sentence words and Yellow Shading intensity represents the morphological variation generation from the source input sentence words.

SFC	
Input Sentences	Daschle, the former Senate Democratic leader, said he would have not been able to operate with the full faith of Congress and the American people. “This work will require a leader who can operate with the full faith of Congress and the American Daschle said in a statement released by the White House.
Reference (*best*)	Daschle said the work would require the full faith of Congress and the American people.
System Output	Daschle: It will require a leader who can operate with the full faith of Congress.
Input Sentences	Senators and Obama had stood by him, but Daschle withdrew today, saying he did not want to be a distraction. Asked about the stunning reversal, White House spokesman Robert Gibbs said Daschle made the decision because he did not want to be a distraction to Obama’s agenda.
Reference (*best*)	Daschle made the decision because he did not want to be a distraction.
System Output	Daschle said he did not want to be a distraction in Obama’s agenda.