Neural Document Modeling and Summarization

Yang Liu

Doctor of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2020

Abstract

Document summarization is the task of automatically generating a shorter version of a document or multiple documents while retaining the most important information. The task has received much attention in the natural language processing community due to its potential for various information access applications. Examples include tools that digest textual content (e.g., news, social media, reviews), answer questions, or provide recommendations. Summarization approaches are dedicated to processing single or multiple documents as well as creating extractive or abstractive summaries. In extractive summarization, summaries are formed by copying and concatenating the most important spans (usually sentences) from the input text, while abstractive approaches are able to generate summaries using words or phrases that are not in the original text.

A core module within summarization is how to represent documents and distill information for downstream tasks (e.g., abstraction or extraction). Thanks to the popularity of neural network models and their ability to learn continuous representations, many new systems have been proposed for document modeling and summarization in recent years. This thesis investigates different approaches with neural network models to address the document summarization problem. We develop several novel neural models considering extractive and abstractive approaches for both single-document and multi-document scenarios.

We first investigate how to represent a single document with a randomly initialized neural network. Contrary to previous approaches that ignore document structure when encoding the input, we propose a structured attention mechanism, which can impose a structural bias of document-level dependency trees when modeling a document, generating more powerful document representations. We first apply this model to the task of document classification, and subsequently to extractive single-document summarization using an iterative refinement process to learn more complex tree structures. Experimental results on both tasks show that the structured attention mechanism achieves competitive performance.

Very recently, pretrained language models have achieved great success on several natural language understanding tasks by training large neural models on an enormous corpus with a language modeling objective. These models learn rich contextual information and to some extent are able to learn the structure of the input text. While summarization systems could in theory also benefit from pretrained language models, there are some potential obstacles to applying these pretrained models to document summarization tasks. The second part of this thesis focuses on how to represent a single document with pretrained language models. Beyond previous approaches that learn solely from the summarization dataset, this thesis proposes a framework for using pretrained language models as encoders for both extractive and abstractive summarization. The framework achieves state-of-the-art results on three datasets.

Finally, in the third part of this thesis, we move beyond single documents and explore approaches for using neural networks for summarizing multiple documents. We analyze why the application of existing neural summarization models to this task is challenging and develop a novel modeling framework. More concretely, we propose a ranking-based pipeline and a hierarchical neural encoder for processing multiple input documents. Experiments on a large-scale multi-document summarization dataset, show that our system can achieve promising performance.

Acknowledgements

My PhD study in Edinburgh is wonderful, cheerful and unforgettable, and I want to thank everyone who has been with me during this journey. Furthermore, I want to express my appreciation to those who have been especially helpful to me.

I want to thank my supervisor Mirella Lapata who has always supported me with her knowledgeable guidance and passionate encouragement. I learned a lot from those weekly meetings and I appreciate the inspirations from Mirella during both my highs and lows. It will always be a honor and fortune for me to be her student.

I also want to thank my second supervisor Ivan Titov for helpful discussions and supervisions which greatly motivate my research. I am grateful to Rico Sennrich for helpful feedbacks on my first-year review. I would like to thank my examiners, Shay Cohen and Dragomir Radev, for their efforts to examine my thesis and viva, and for those valuable comments.

I am grateful to Matt Gardner at Allen Institute for AI for his insightful mentoring during that wonderful internship. I would like to acknowledge Google for supporting my study with the PhD fellowship and Tom Kwiatkowski at Google for being my fellowship mentor.

I feel honored to conduct my research in the Edinburgh NLP group, and I would like to thank Alex Lascarides, Shay Cohen, Sharon Goldwater, Kenneth Heafield, Frank Keller, Adam Lopez, Walid Magdy, Rico Sennrich, Mark Steedman, Ivan Titov and Bonnie Webber for their knowledgeable comments on my presentations in the group meetings. Special thanks to my coauthors, Bowen Li, Jianpeng Cheng and Laura Perez-Beltrachini, for their excellent research ideas. I will definitely miss the outstanding researchers in Mirella's group. Many thanks to Tom Sherborne, Reinald Kim Amplayo, Hao Zheng, Yumo Xu, Jiangming Liu, Rui Cai, Ratish Puduppully, Jonathan Mallinson, Li Dong, Jianpeng Cheng, Philip Gorinski and Xingxing Zhang. I learned a great deal from the research discussions and the weekly group meetings. I want to also thank all the Edinburgh Informatics members, Arthur, Bailin, Biao, Bowen, Craig, Chunchuan, Esma, He, Jiangming, Laura, Shashi, Sicong, Tom, Xinchi, Yanpeng, etc . I would like to specially thank my office mates, Bowen Li, Kai Xu, Yumo Xu, Joachim Fainberg, Marco Damonte and Sam Ribeiro. We created a productive and enjoyable working environment together, and those daily discussions inspired me a lot.

Finally, I want to express my deepest appreciation to my parents for their unconditional love and support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Yang Liu*)

Table of Contents

Chapter 1

Introduction

The increasing availability of content on the Internet has changed the way people accessing information. The number of online documents is huge. In 2016, it is estimated that Google search engine has indexed over 45 billion webpages (van den Bosch et al., 2016). The sizable amount of information has led to the information overload problem (Feldman and Sanger, 2007). To fulfil the requirements of efficiently browsing these documents and finding useful information, many challenges are proposed. How to extract important information from one long document? How to identify related information from multiple documents? How to generate a human-readable summary that contains important information? Document Summarization, the task of using computers to automatically generate a shortened but informative version of one or multiple documents, is one core method to tackle these problems.

Radev et al. (2002b) formally defines a summary as:

> *A text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that.*

From this definition, we can deduce two main properties for a summary:

1. Being Informative: A summary should contain the most important information of the original text(s).

2. Being Compressive: A summary should not contain redundant information, and should be significantly shorter than the original text(s).

Using computers and programs to automatically generate summaries has been formulated as the task of Document Summarization. Research on this topic has a long history, starting from early frequency-based methods (Luhn, 1958; Baxendale, 1958) and feature-based machine learning methods (Aone et al., 1997; Lin, 1999; Jones et al., 1999; Kupiec et al., 1999). Recently, neural network-based models have achieved promising results on many Natural Language Processing (NLP) tasks (Collobert et al., 2011; Devlin et al., 2019). In these neural network models, words are first mapped to continuous vectors called word embeddings, and then processed by non-linear transformations. One advantage of neural networks is the flexibility of these models. Different neural architectures can be designed for different tasks, and multiple neural network layers can be stacked or combined, to form a large network with powerful expressivity. Gradient-descent based optimization methods are usually used to tune the network parameters. Additionally, thanks to the encoder-decoder architecture (Bahdanau et al., 2015), neural network models have shown impressive performance on language generation tasks like Machine Translation (Bahdanau et al., 2015), Image Captioning (Xu et al., 2015) and Sentence Simplification (Zhang and Lapata, 2017). Their potential to generate fluent and abstractive text has led to many advances in natural language generation tasks and summarization is no exception. Because of these advantages, in this thesis we will explore how to use neural networks to build and improve document summarization systems.

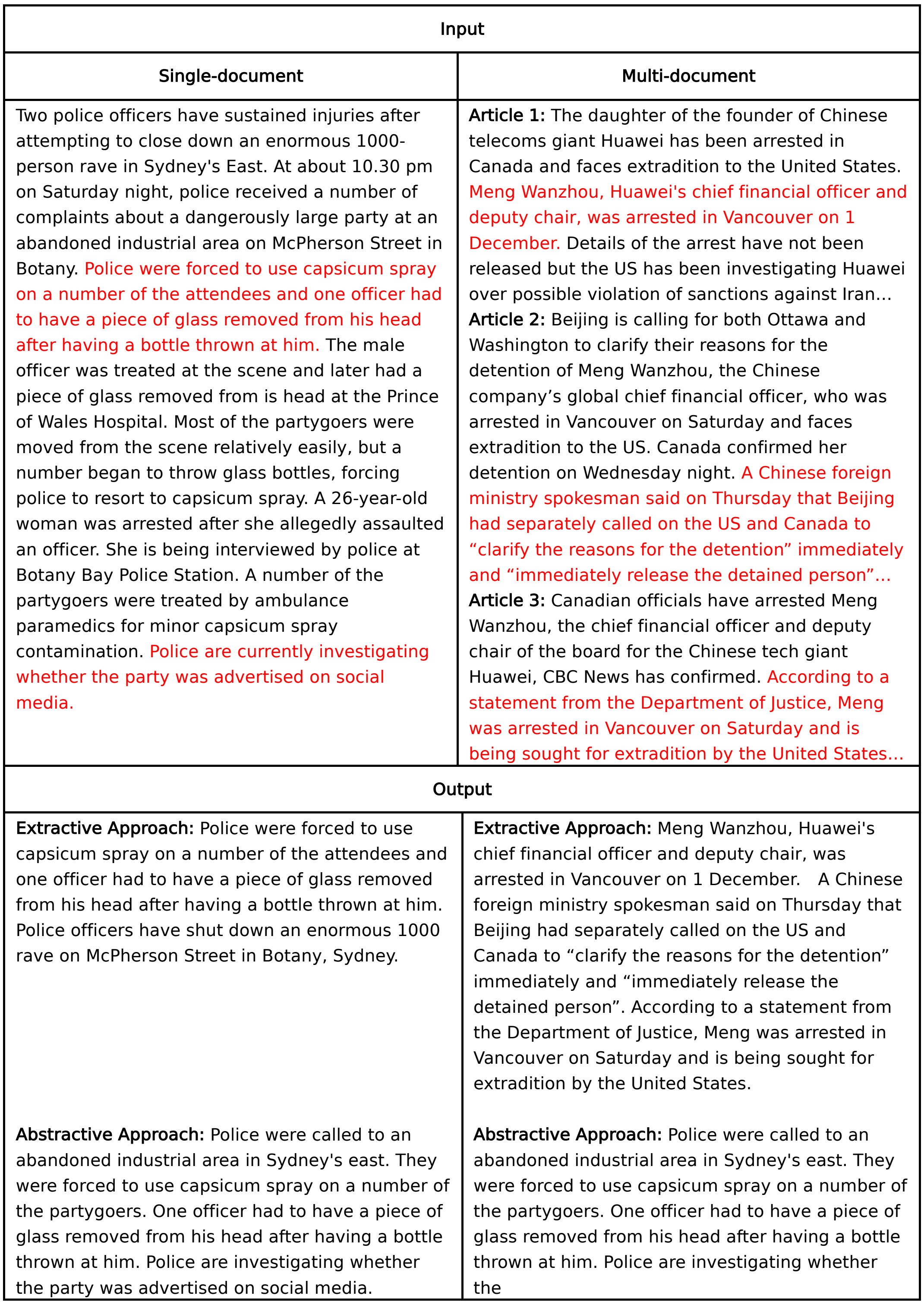
1.1 History of Document Summarization

Document summarization can be classified into different subtasks by different factors (Jones et al., 1999). Following are two main factors:

- **Single-document vs. Multi-document**: This factor simply means the number of input documents to the summarization system. Single-document summarization is the task of generating a summary based on one input document, while in multi-document summarization the input consists of multiple documents.
- **Extractive vs. Abstractive**: This factor concerns the form of the generated summaries. Extractive summarization is the task of generating summaries by selecting text spans (words, phrases or sentences) from the input document(s). Abstractive summarization requires the summarizer to be capable of generating novel text spans that did not appear in the input document(s).

In Figure1.1, we present examples of different types of the summarization task.

Starting from IBM's first attempt to build a document summarization system in 1958 (Luhn, 1958), research on automatic document summarization has a long his-



Input	
Single-document	**Multi-document**
Two police officers have sustained injuries after attempting to close down an enormous 1000-person rave in Sydney's East. At about 10.30 pm on Saturday night, police received a number of complaints about a dangerously large party at an abandoned industrial area on McPherson Street in Botany. Police were forced to use capsicum spray on a number of the attendees and one officer had to have a piece of glass removed from his head after having a bottle thrown at him. The male officer was treated at the scene and later had a piece of glass removed from is head at the Prince of Wales Hospital. Most of the partygoers were moved from the scene relatively easily, but a number began to throw glass bottles, forcing police to resort to capsicum spray. A 26-year-old woman was arrested after she allegedly assaulted an officer. She is being interviewed by police at Botany Bay Police Station. A number of the partygoers were treated by ambulance paramedics for minor capsicum spray contamination. Police are currently investigating whether the party was advertised on social media.	**Article 1:** The daughter of the founder of Chinese telecoms giant Huawei has been arrested in Canada and faces extradition to the United States. Meng Wanzhou, Huawei's chief financial officer and deputy chair, was arrested in Vancouver on 1 December. Details of the arrest have not been released but the US has been investigating Huawei over possible violation of sanctions against Iran… **Article 2:** Beijing is calling for both Ottawa and Washington to clarify their reasons for the detention of Meng Wanzhou, the Chinese company’s global chief financial officer, who was arrested in Vancouver on Saturday and faces extradition to the US. Canada confirmed her detention on Wednesday night. A Chinese foreign ministry spokesman said on Thursday that Beijing had separately called on the US and Canada to “clarify the reasons for the detention” immediately and “immediately release the detained person”… **Article 3:** Canadian officials have arrested Meng Wanzhou, the chief financial officer and deputy chair of the board for the Chinese tech giant Huawei, CBC News has confirmed. According to a statement from the Department of Justice, Meng was arrested in Vancouver on Saturday and is being sought for extradition by the United States…
Output	
Extractive Approach: Police were forced to use capsicum spray on a number of the attendees and one officer had to have a piece of glass removed from his head after having a bottle thrown at him. Police officers have shut down an enormous 1000 rave on McPherson Street in Botany, Sydney.	**Extractive Approach:** Meng Wanzhou, Huawei's chief financial officer and deputy chair, was arrested in Vancouver on 1 December. A Chinese foreign ministry spokesman said on Thursday that Beijing had separately called on the US and Canada to “clarify the reasons for the detention” immediately and “immediately release the detained person”. According to a statement from the Department of Justice, Meng was arrested in Vancouver on Saturday and is being sought for extradition by the United States.
Abstractive Approach: Police were called to an abandoned industrial area in Sydney's east. They were forced to use capsicum spray on a number of the partygoers. One officer had to have a piece of glass removed from his head after having a bottle thrown at him. Police are investigating whether the party was advertised on social media.	**Abstractive Approach:** Police were called to an abandoned industrial area in Sydney's east. They were forced to use capsicum spray on a number of the partygoers. One officer had to have a piece of glass removed from his head after having a bottle thrown at him. Police are investigating whether the

Figure 1.1: Examples of different types of summaries. The left input is for single-document summarization, while the right is for multi-document summarization with three different articles as input. The output shows both extractive and abstractive summaries. The extractive summary selects important text spans from the original article (marked in red), while the abstractive summary is more coherent with novel words and phrases.

tory. Early work on summarization involved frequency-based and rule-based methods. In the 1990s, with the advent of machine learning techniques in NLP, a series of papers were proposed to use statistical-based methods to produce document summaries and many feature-based machine learning systems were developed (Aone et al., 1997; Lin, 1999; Jones et al., 1999; Kupiec et al., 1999). Graph-based methods have attracted much attention in the beginning of the twenty-first century (Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Wan and Yang, 2008; Wan, 2008). Recently, with the success of neural network-based models in multiple NLP tasks, neural network-based summarizers have become the center of attention in the research community (Rush et al., 2015; Cheng and Lapata, 2016; Nallapati et al., 2017; See et al., 2017; Paulus et al., 2018; Gehrmann et al., 2018; Narayan et al., 2018a).

Early Work In 1958, in an attempt to reduce the overload of information in text data, Hans Peter Luhn developed a program (Luhn, 1958) for scoring the importance of sentences in scientific and technical documents. Luhn proposed the idea that the occurrence of some specific words in a sentence can indicate that this sentence is more significant than others that do not contain these words. Luhn implemented several rules to remove stop-words and non-content words from the documents, and measure the importance of each sentence by counting the number of occurrences of the remaining content words. All sentences were processed by his program and ranked by their importance factors. The top-ranked sentences were then selected and concatenated to form the summary of the input article.

Related work (Baxendale, 1958) found that the position of the sentence within the document could be another inspiring feature for selecting important content. By analyzing 200 paragraphs in the corpus, Baxendale found that most summary sentences occur as the first or the last sentence of the paragraph. Based on this assumption, a widely-used baseline method for modern summarization systems was developed: the LEAD baseline that extracts the first several sentences of a document as the summary.

In 1969, Edmundson (1969) proposed the idea that instead of using one single feature, several factors could jointly indicate the importance of a sentence within a document. Four major features were identified :

1. The frequency of a word that appears in the article.
2. The position of a sentence in the article and in the section.
3. The number of words that also appear in the article title or the section heading.

4. The frequency of some specific cue-words.

Edmundson applied a linear summation of these features to calculate the importance score of each sentence. These four features have also been used by most machine-learning based summarizers, and also in modern extractive summarization systems. Another important contribution of Edmundson (1969) was the creation of a summarization corpus consists of 400 document-summary pairs of scientific and technical articles. 200 pairs were used to decide the weight of the linear summation, and the other 200 pairs were used to evaluate system performance. The same schema sets the direction for later development of machine learning based summarization systems and has been employed by subsequent empirically-based research on document summarization.

Feature-based Machine Learning Methods Kupiec et al. (1999) proposed a learning-based method for extracting sentences from input documents as summaries. A naive-Bayes classifier was implemented to identify whether a sentence should be included in the summary or not. The feature set was an extended version of Edmundson (1969), with two more features added: whether the length of a sentence is longer than a pre-defined threshold and the presence of uppercase words. Experiments were done on a corpus of scientific and technical documents, and the summarization results were evaluated as a classification task, where each sentence in the reference summary was matched to a sentence in the source document. Aone et al. (1997) developed the Dim-Sum system with richer features for the summarization task. These novel features included term frequency and inverse document frequency of words, the occurrence of named-entities and phrases composed of two nouns. In Lin and Hovy (1997), the sentence position feature was thoroughly analysed and proved to be very important for the summarization task. In later work, Lin (1999) pointed out that the features should not be learned individually for text summarization. Instead, he started to use a decision tree classifier with an even richer feature set to extract summary sentences. Osborne (2002) also proposed to model features jointly by using a maximum entropy classifier and showed empirically that the system produced better summaries than a naive-Bayes classifier.

From 2001 to 2002, the Document Understanding Conference (DUC), held by National Institute of Standards and Technology, designed the task of single-document news summarization. The task required a participating system to automatically generate a 100-word summary of a single news article. In this task, the LEAD baseline that extracts the first *k* sentences from a news article to form the summary was found to be a surprisingly effective method despite its simplicity (Nenkova, 2005). Svore et al. (2007) implemented a summarization system which learned features from an external summarization corpus, in combination with a ranking based algorithm, successfully outperforming the LEAD baseline with statistical significance on the DUC datasets. From the years of 2007 to 2013, more machine learning methods for selecting words or sentences were proposed, and more features were found useful for the summarization task. For selecting words, new features include proper nouns (Fattah and Ren, 2009), phrase structures and dependency structures (Woodsend and Lapata, 2010), subjective words and phrases (Carenini et al., 2008), word co-occurrence (Liu et al., 2009) and lexical similarity (Barrera and Verma, 2012). For selecting sentences, new features include cue-phrases (Ferreira et al., 2013) and sentence centrality (Abuobieda et al., 2012).

Graph-based Methods An impactful thread in the history of document summarization is the application of unsupervised graph-based methods. These methods are influenced by the PageRank algorithm. They theoretically assume that a document (or multiple documents) can be represented as a graph, where each node is a text span (usually is a sentence or a paragraph) and graph edges indicate the similarity between the connected node pairs. Figure 1.2 presents an example graph representation of a document. After normalizing the edge weights of the graph to a Markov chain (where edge weights correspond to the probability of transitioning from one state to another), PageRank-like algorithms can be applied on this graph to generate the probabilities of staying at each node, which will converge to a stationary distribution after a few iterations. These probabilities can be considered as the degree of centrality of each node within this graph, indicating the importance of the corresponding text spans. An example of document graph is shown in Figure 1.2. LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004) were two early graph-based summarization systems proposed with different ways of building the graph, where the former focuses on multi-document summarization and the latter focuses on single-document summarization. Many methods attempted to design different graph representations for the summarization task. Wan and Xiao (2008) proposed to incorporate external cross-document relationships of sentence pairs to build the graph for a single document. Guinaudeau and Strube (2013) introduced a bipartite graph based on discourse entities. Parveen et al. (2015) designed a method for building a topical graph with Latent

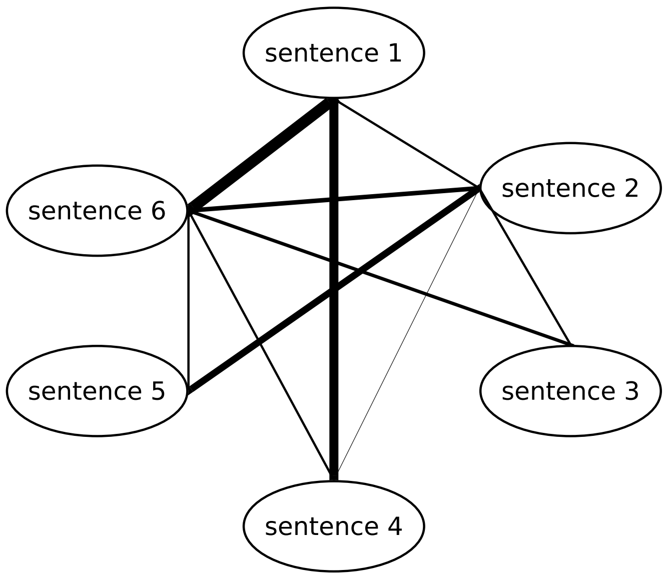


Figure 1.2: Example of a graph representation of a document consisting of six sentences. The nodes are sentences, and the weights on the edges are similarity scores. The thickness of lines indicates high or low similarity values. A threshold is applied to remove edges with low scores.

Drichlet Allocation (Blei et al., 2003).

Neural-based Methods The success of neural network models in several Natural Language Processing tasks including sentiment classification (Socher et al., 2013), machine translation (Bahdanau et al., 2015) and syntactic parsing (Bowman et al., 2016) has also led to the development of neural network-based summarization systems. There are two major advantages of using neural network-based models for summarization. Firstly, the neural models eschew the need for expensive feature engineering, all parameters can be trained by gradient descent algorithms and more flexible architectures can be designed for representing the documents. Secondly, the neural models can generate texts in a more fluent and abstractive way by framing the abstractive summarization problem as a conditional language generation task.

Neural models usually consider extractive summarization as a sentence classification problem: a neural encoder creates sentence representations and a classifier predicts which sentences should be selected as summaries. SUMMARUNNER (Nallapati et al., 2017) is one of the earliest neural approaches adopting an encoder based on Recurrent Neural Networks. REFRESH (Narayan et al., 2018b) is a reinforcement learning-based system trained by globally optimizing the ROUGE (Lin, 2004) metric. More recent work achieved higher performance with more sophisticated neural structures. LATENT (Zhang et al., 2018c) frames extractive summarization as a latent variable inference problem; instead of maximizing the likelihood of "gold" standard labels, their latent model directly maximizes the likelihood of human summaries given selected sentences. BANDITSUM (Dong et al., 2018) presents a contextual bandit learning method trained by reinforcement learning algorithms. NEUSUM (Zhou et al., 2018) scores and selects sentences jointly. DEEPCHANNEL (Shi et al., 2019) applies a neural network-based channel model, combined with an iterative process for extracting summaries. Xu and Durrett (2019) designed a syntax-based compression model which can decide whether to remove certain phrases or words in the predicted extracts.

Neural approaches to abstractive summarization conceptualize the task as a sequence-to-sequence problem, Rush et al. (2015) and Nallapati et al. (2016) were among the first to apply the neural encoder-decoder architecture to text summarization. See et al. (2017) enhanced this model with a pointer-generator network which allows it to copy words from the source text, and a coverage mechanism which keeps track of words that have been summarized. More recently, Celikyilmaz et al. (2018) proposed an abstractive system where multiple agents (encoders) represent the document together with a hierarchical attention mechanism (over the agents) for decoding. Their Deep Communicating Agents model is trained end-to-end with reinforcement learning. Paulus et al. (2018) also presented a deep reinforced model for abstractive summarization which handles the coverage problem with an intra-attention mechanism where the decoder attends over previously generated words. Gehrmann et al. (2018) followed a bottom-up approach; a content selector first determines which phrases in a source document should be part of the summary, and a copy mechanism is applied only to preselected phrases during decoding. Narayan et al. (2018a) proposed an abstractive model which is particularly suited to extreme summarization (i.e., single sentence summaries), based on convolutional neural networks and additionally conditioned on topic distributions. You et al. (2019) proposed a saliency-selection network in the decoder for better modeling the salient words in the input text. Lebanoff et al. (2019) analyzed the importance of scoring sentence singletons and pairs, which are summary-worthy sentences, before generating the abstractive summaries.

Multi-document Summarization In multi-document summarization, where the source texts are from multiple different articles, the redundancy of information plays an important role. Information occurring in multiple input articles tends to be more important and should be more likely to be included in the summary. Based on this assumption, centroid-based clustering methods (Radev et al., 2002a, 2004; Wan and Yang, 2008; Wang et al., 2009) are popular for the task of multi-document summarization. These approaches collect sentences into multiple clusters, where sentences in the same cluster should be similar to each other. These clusters represent different topics of the input articles and the clusters formed with more sentences are assumed to be more important. Then, one or multiple representative sentence are selected from the top clusters to form the summary. The Maximum Marginal Relevance (MMR; Carbonell and Goldstein 1998 is also widely used in multi-document summarization. This method greedily selects summary sentences based on their importance scores, but also penalize sentences that lead to redundancy. SumBasic (Nenkova and Vanderwende, 2005) is a method developed to reduce the redundancy in the generated summaries by taking into account the context of previous selected summary sentences. Graph-based methods are also popular models when summarizing multiple documents. They can leverage the similarity of sentence pairs, while not constraining each sentence to belong to only one cluster. For example, in LexRank (Erkan and Radev, 2004), the similarity of sentence pairs are indicated by the weights of the edges connecting two sentences, and the importance score of each sentence is calculated by using a PageRank algorithm.

1.2 Challenges

Summarizing documents is a challenging task both on account of the document understanding and the language generation. In what follows we discuss challenges in more details, focusing on neural document summarization.

Document Structure Most neural-based approaches to (single-document) extractive summarization frame the task as a sequence labeling problem. The idea is to predict a label for each sentence specifying whether it should be included in the summary (Cheng and Lapata, 2016; Nallapati et al., 2017). In these systems, inter-sentential relations are usually captured in a sequential manner, without taking the structure of the document into account, although the latter has been shown to correlate with what readers perceive as important in a text (Marcu, 1999). Another problem in neural-based extractive models is the lack of interpretability. While capable of identifying summary sentences, these models are not able to rationalize their predictions (e.g., a sentence is in the summary because it describes important content upon which other related sentences elaborate).

The summarization literature offers examples of models which exploit the structure of the underlying document, inspired by existing theories of discourse such as Rhetorical Structure Theory (RST; Mann and Thompson 1988). Most approaches produce summaries based on tree-like document representations obtained by a parser trained on discourse annotated corpora (Carlson et al., 2001; Prasad et al., 2008). For instance, Marcu (1999) argues that a good summary can be generated by traversing the RST discourse tree structure top-down, following nucleus nodes (discourse units in RST are characterized regarding their text importance; nuclei denote central units, whereas satellites denote peripheral ones). Other work (Hirao et al., 2013a; Yoshida et al., 2014) extends this idea by transforming RST trees into dependency trees and generating summaries by tree trimming. Gerani et al. (2014) summarize product reviews; their system aggregates RST trees representing individual reviews into a graph, from which an abstractive summary is generated. Zhang et al. (2002) shows that using structures as posited in Cross-document Structure Theory is helpful to multi-document summarization. However, incorporating structural information into neural summarization systems is still challenging, not only because of the reliance on a parser which is expensive to obtain (since it must be trained on labeled data), using document structure within a pipeline-style architecture will unavoidably lead to error prone, presenting a major obstacle to its widespread use. To better take advantages of document structures, a potential solution is to model them as latent variables, and learn them with the task objective. In this manner, document structures could be learned in an end-to-end fashion and without recourse to external parsers.

Deep Understanding of Documents Although extractive summarization has been so far modeled without relying on deep semantic analysis of the input documents, recent studies (Nenkova and McKeown, 2012) find that models which produce summaries based on surface features like word frequencies and sentence positions still show a large gap in both automatic and human evaluations compared to human-authored summaries. Neural network-based models also fall behind reference summaries by a large margin in human evaluation (Narayan et al., 2018b). Deeper understanding of the input document may be needed for further boosting extractive summarization performance. It is usually assumed that abstractive summarization requires deep understanding and reasoning in both the encoding and the decoding phases (Nenkova and McKeown, 2012). When encoding the input documents, abstractive summarizers should be able to determine explicit or implicit meaning for words, sentences and documents, making global inferences, deciding which information should be used to generate the summary. In the decoding process, the abstractive summarizer should be able to generate fluent text containing the most important information, while avoiding redundancy and learning to abstract over concepts and expressions in the source documents. Many neural models have designed different neural architectures to achieve better summarization performance (Nallapati et al., 2017; Narayan et al., 2018a), and recent advances on pretrained language models (Peters et al., 2018; Devlin et al., 2019) may offer an effective alternative for the goal of better encoding the documents.

Modelling Multiple Documents A major obstacle to the application of end-to-end models to multi-document summarization is the sheer size and number of source documents which can be very large. As a result, it is practically infeasible (given memory limitations of current hardware) to train a model which encodes all of them into vectors and subsequently generates a summary from them. Another challenge for multi-document summarization is the hierarchical structure of the input. Different from a single document input, the input to multi-document summarization is formed first from sentences to documents, and then from multiple documents to one meta-input. Meanwhile, the relations that might exist among multiple documents should also be captured by the summarization system to better model the salient information in the input. For example, different web pages might repeat the same content, include additional content, present contradictory information, or discuss the same fact in a different light (Radev, 2000). The realization that cross-document links are important in isolating salient information, eliminating redundancy, and creating overall coherent summaries, has led to the widespread adoption of graph-based models for multi-document summarization (Erkan and Radev, 2004; Christensen et al., 2013; Wan, 2008; Parveen and Strube, 2014). Graphs conveniently capture the relationships between textual units within a document collection and can be easily constructed under the assumption that text spans represent graph nodes and edges are semantic links between them. To effectively leverage the power of neural methods for abstractive multi-document summarization, a new model which is capable of effectively processing multiple input documents and capturing the relation between these documents is needed.

1.3 Thesis Overview

In this thesis, we aim at investigating existing problems in neural document summarization and developing effective neural summarization models while addressing the challenges outlined in the previous section.

We first investigate one core module of the document summarization task, document modelling. Our goal is to incorporate document structural information into document modelling, generating better document representations. Recent work provides strong evidence that better document representations can be obtained by incorporating structural knowledge (Ji and Smith, 2017; Bhatia et al., 2015; Yang et al., 2016). Inspired by existing theories of discourse, representations of document structure have assumed several guises in the literature, such as trees in the style of Rhetorical Structure Theory (RST; Mann and Thompson, 1988), graphs (Lin et al., 2011; Wolf and Gibson, 2006), entity transitions (Barzilay and Lapata, 2008), or combinations thereof (Lin et al., 2011; Mesgar and Strube, 2015). The availability of discourse annotated corpora (Carlson et al., 2001; Prasad et al., 2008) has led to the development of off-the-shelf discourse parsers (e.g., Feng and Hirst, 2012a; Liu and Lapata, 2017), and the common use of trees as representations of document structure. For example, Bhatia et al. (2015) improve document-level sentiment analysis by reweighing discourse units based on the depth of RST trees, whereas Ji and Smith (2017) show that a recursive neural network built on the output of an RST parser benefits text categorization in learning representations that focus on salient content. Unfortunately, the reliance on labeled data, which is both difficult and highly expensive to produce, presents a major obstacle to the widespread use of discourse structure for document modeling. Moreover, despite recent advances in discourse processing, the use of an external parser often leads to pipeline-style architectures where errors propagate to later processing stages, affecting model performance.

Our first work focuses on learning deeper structure-aware document representations, drawing inspiration from efforts to empower neural networks with a structural bias (Cheng et al., 2016). Kim et al. (2017) introduce structured attention networks which are generalizations of the basic attention procedure, allowing to learn sentential representations while attending to partial segmentations or subtrees. We extend this idea by referring to the matrix-tree theorem (Kirchhoff, 1847; Tutte, 1984), and constrain the self-attention weights of neural models as non-projective dependency structures. In this way, without recourse to an external parser, our model is able to learn task-specific dependency structures, obtaining better document representations.

We then apply this structured attention model to the extractive summarization task by re-framing the task into a tree induction problem, instead of a sequence labeling problem. Drawing inspiration from existing discourse-informed summarization models (Marcu, 1999; Hirao et al., 2013a), our model represents documents as multi-root dependency trees where each root node is a summary sentence, and the subtrees attached are sentences whose content is related to and covered by the summary sentence. We proposed that structured attention can be used as both the objective and attention weights for extractive summarization, and document-level dependency trees can be induced while predicting the output summary. This leads to better summarization performance and brings more interpretability in the summarization process by helping explain how document content contributes to the model's decisions.

Modeling the tree structure provides deeper understanding of input document on the discourse aspect. Meanwhile, we find there are other aspects that could also be improved to further boost the summarization performance. Unlike previous studies that train document summarization models solely on annotated corpora with human-written summaries, we first pretrain the encoder on a large-scale unannotated corpus, to learn the rich linguistic features and complex contextual information. Pretrained language models have recently emerged as a key technology for achieving impressive gains in a wide variety of natural language tasks, ranging from sentiment analysis (Xu et al., 2019a), to question answering (Yang et al., 2019) and named entity recognition (Devlin et al., 2019). State-of-the-art pretrained models include ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and more recently Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) BERT combines both word and sentence representations in a single very large Transformer (Vaswani et al., 2017). With its deep neural architecture and pretraining on vast amounts of text, BERT is found to be able to obtain richer representations of sentences or documents and capture more long-tail features (Tenney et al., 2019).

In most cases, pretrained language models have been employed as encoders for sentence- and paragraph-level natural language understanding problems (Devlin et al., 2019) involving various classification tasks. In this thesis, we examine the influence of language model pretraining on text summarization. We explore the potential of BERT for text summarization under a general framework encompassing both extractive and abstractive modeling paradigms. We propose a novel document-level encoder based on BERT which is able to encode a document and obtain representations for its sentences.

Our extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers to capture document-level features for extracting sentences. Our abstractive model adopts an encoder-decoder architecture, combining the same pre-trained BERT encoder with a randomly-initialized Transformer decoder (Vaswani et al., 2017). We design a new training schedule which separates the optimizers of the encoder and the decoder in order to accommodate the fact that the former is pretrained while the latter must be trained from scratch. Finally, motivated by previous work showing that the combination of extractive and abstractive objectives can help generate better summaries (Gehrmann et al., 2018), we present a two-stage approach where the encoder is fine-tuned twice, first with an extractive objective and subsequently on abstractive summarization.

Most existing work on neural document summarization focused on single document summarization. Multi-document summarization, another important form of summarization task, although has a wide range of applications including summarizing related webpages, news articles of the same topic and product reviews, has been largely neglected. In this thesis, we expand the application of neural summarizers to the multi-document setting. We observe different challenges for multi-document summarization compared to single-document summarization. Firstly, the input to multi-document summarization is long and redundant; secondly, the input to multi-document summarization has a hierarchical structure where multiple interrelated documents are each composed by interrelated sentences which are composed by tokens; thirdly, the inter-document relations are important for summarizing multiple documents. We propose several solutions to these challenges and design a neural summarization model which can effectively process multiple input documents and distill abstractive summaries. Our model augments the previously proposed Transformer architecture with the ability to encode multiple documents in a hierarchical manner. We represent cross-document relationships via an attention mechanism which allows to share information across multiple documents as opposed to simply concatenating text spans and feeding them as a flat sequence to the model. In this way, the model automatically learns richer structural dependencies among textual units, thus incorporating well-established insights from earlier work of graph-based document representations.

The main contributions of this thesis are:

1. A new structured attention mechanism that can normalize the self-attention weights as the probabilities of non-projective dependency trees, incorporating more structural constraints into neural models.

2. A proposal of re-framing the single-document summarization task as a tree induction problem, generating more precise summary sentences.

3. A general framework and a training schedule for using pretrained language models as encoders for neural network based summarization models, under both extractive and abstractive settings.

4. A multi-document summarization framework for generating abstractive summaries of multiple input documents with hierarchical Transformer models.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 presents background knowledge with regard to neural network models. We introduce the general framework of using neural networks for Natural Language Processing tasks. Then two widely used neural networks: Long short-term memory network (Hochreiter and Schmidhuber, 1997) and the Transformer network (Vaswani et al., 2017) are described. We also discuss the related work of neural summarization models, including both extractive models and abstractive models.

- Chapter 3 presents the structured attention mechanism for modelling documents. We first introduce as background the self-attention mechanism (Parikh et al., 2016; Kim et al., 2017). We then describe our proposed model which is based on the tree-matrix theorem (Kirchhoff, 1847; Tutte, 1984). The model is tested on multiple document classification tasks and experimental results show the superiority of the structured attention mechanism.

- Chapter 4 presents our structured summarization model for single-document extractive summarization. We first propose the idea of re-framing the task as a tree induction problem, which could introduce more interpretability and structural constraints into the generated summaries. We then show that with an iterative process, we can use the structured attention mechanism for this task and gradually learn increasingly complex document structures. Experiments are performed on two large-scale summarization datasets: the CNN/DailyMail dataset (Hermann et al., 2015) and the New York Times (Sandhaus, 2008). Experimental results show that the proposed model can achieve competitive results.

- Chapter 5 introduces our general framework for using pretrained language models for text summarization tasks. Firstly, background on pretrained language models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) are presented. Secondly, we analyze both the advantages and challenges of using pretrained language models for text summarization. Then, we show that BERT can be modified and used as an encoder for modelling a document and generating sentence-level representations. Also, we propose a novel training schedule that can help narrow the gap between a pretrained encoder and a randomly-initialized decoder for abstractive summarization. The framework is named BERTSUM and we present implementation details of the model. Experiments are performed on three large-scale summarization datasets: the CNN/DailyMail dataset (Hermann et al., 2015), the New York Times (Sandhaus, 2008), and the XSum dataset (Narayan et al., 2018a). We show that BERTSUM outperforms previous models by a large margin across datasets under both extractive and abstractive settings.

- Chapter 6 focuses on the task of multi-document summarization. A new dataset called WIKISUM (Liu et al., 2018) is introduced as the first large-scale dataset for multi-document summarization. We outline three challenges in summarizing multiple documents and propose corresponding solutions. For the first challenge of long and redundant input, a paragraph ranker is designed to score each paragraph based on its usefulness for summarization. For the second challenge of processing multi-document input, a new Transformer model with a hierarchical architecture is designed. For the third challenge of incorporating information in external document graphs, we introduce a graph-informed attention mechanism into the Transformer model. Evaluation on the WIKISUM dataset shows that the proposed model can achieve better summarization results compared with previous systems.

- Chapter 7 concludes the thesis, and discusses directions for future work.

Chapter 2

Background

As introduced in Chapter 1, a typical neural document summarization model includes a document encoder to transform discrete words within source documents into continuous vector representations. For abstractive summarization, a decoder module additionally generate a human-readable abstractive summary based on source documents. Both the encoder and the decoder are based on neural networks (Bahdanau et al., 2015). In this chapter, we first provide background for two commonly used neural networks, namely Recurrent Neural Networks (RNNs) and Transformer models (Vaswani et al., 2017), for building the encoder and the decoder. RNN models process words in a time-dependent manner. Long Short-Term Memory network (LSTM; Hochreiter and Schmidhuber 1997) is an advanced variant of RNN with gating mechanisms. The Transformer model aims at reducing the fundamental constraint of sequential computation in RNNs in favor of applying a self-attention mechanism which directly models relationships between all words in a sentence. We will also discuss related work and the general framework of neural extractive and neural abstractive summarization systems, where the former is usually considered a sentence labelling task, while the latter typically incorporates a more sophisticated neural encoder-decoder architecture.

2.1 Neural Networks

2.1.1 Recurrent Neural Networks

Given a sequence of input vectors $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $\mathbf{x}_t \in \mathbb{R}^d$, Recurrent Neural Networks model this sequence in a temporal manner. More specifically, an RNN holds a hidden state vector $\mathbf{h}$ that is updated at each time step. Formally speaking, at time step

t, an RNN takes the t-th input vector $\mathbf{x}_t$ as network input and computes a hidden state vector $\mathbf{h}_t \in \mathbb{R}^d$ by non-linearly transforming the combination of $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{2.1}$$

A more commonly used version is the Elman network (Elman, 1990) or simple recurrent network:

$$\mathbf{h}_t = \sigma(\mathbf{W}_1\mathbf{h}_{t-1} + \mathbf{W}_2\mathbf{x}_t + \mathbf{b}) \tag{2.2}$$

where $\sigma(\cdot)$ is the non-linear activation function, $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ are transformation weighs, $\mathbf{b} \in \mathbb{R}^d$ is the bias. The output of simple recurrent networks will be $[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n]$, where $\mathbf{h}_t$ is the updated vector of $\mathbf{x}_t$ with contextual information.

Long Short-Term Memory Networks (LSTMs; Hochreiter and Schmidhuber 1997) are a special kind of RNNs, designed for solving the vanishing gradient problem in simple RNNs. When propagating the gradients through time in training simple RNNs with back-propagation, gradients can become extremely small. LSTMs try to solve this problem by introducing gating mechanisms and a memory cell. At each time step t, an input gate $\mathbf{i}_t \in \mathbb{R}^d$ is introduced to control how much information from the input will be fed to the memory cell, a forget gate $\mathbf{f}_t \in \mathbb{R}^d$ is used to decide how much information in the memory cell $\mathbf{c} \in \mathbb{R}^d$ will be reserved from last time step, and an output gate $\mathbf{o}_t \in \mathbb{R}^d$ is used to decide how much information should flow into the output hidden state. The calculation at time step t will be:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \tilde{\mathbf{c}}_t \end{bmatrix} = \begin{bmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}\right) \tag{2.3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.5}$$

where tanh and sigm are element-wise hyperbolic tangent operator and sigmoid operator, and $\odot$ is element-wise multiplication. Matrix $W \in \mathbb{R}^{4d\times 2d}$ represents the transformation weights and $b \in \mathbb{R}^{4d}$ is the bias. A detailed illustration of an LSTM cell at time step t is shown in Figure 2.1. The outputs of LSTMs are the same as those of simple RNNs.

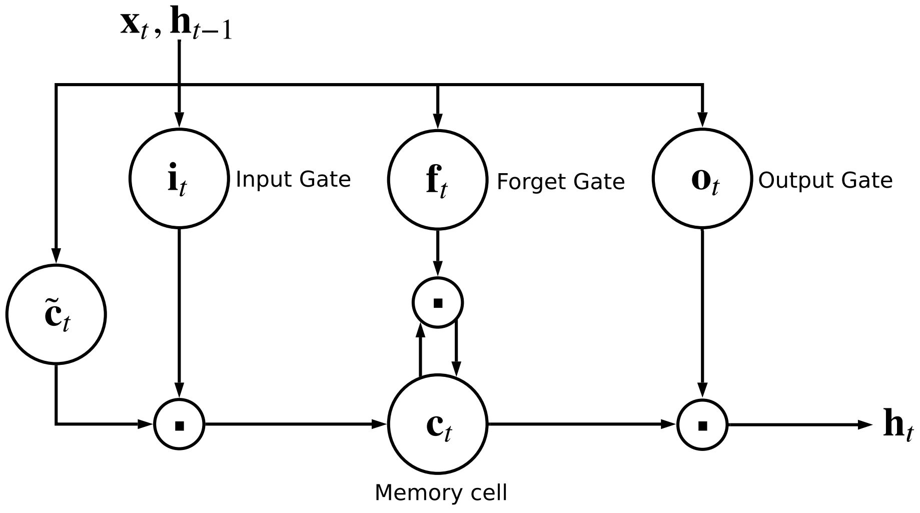


Figure 2.1: Illustration of the Long Short-Term Memory (LSTM) cell. At time step t, $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ are input, $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are input gates, forget gates and output gates, and $\mathbf{c}_t$ is memory cell. $\odot$ indicates element-wise multiplication operation.

2.1.2 Transformer Models

One drawback of RNNs is that the input sequence must be modelled in a temporal order, which makes parallelization of the model hard. Aiming at reducing the fundamental constraint of sequential computation which underlies most architectures based on RNNs, Vaswani et al. (2017) proposed a novel architecture called Transformer for modeling texts. Instead of relying on the recurrent structure, Transformer applies a self-attention mechanism, where each word can collect information from all other contextual words simultaneously.

More formally, given a sequence of input vectors $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $\mathbf{x}_t \in \mathbb{R}^d$, the Transformer is composed of a stack of N identical layers, each of which has two sub-layers:

$$\tilde{H}^l = \text{LayerNorm}(H^{l-1} + \text{MHAtt}(H^{l-1})) \tag{2.6}$$

$$H^l = \text{LayerNorm}(\tilde{H}^l + \text{FFN}(\tilde{H}^l)) \tag{2.7}$$

where $H^l = [\mathbf{h}_1^l, \cdots, \mathbf{h}_n^l]$, and the superscript l indicates layer depth. H^0 is the sequence of input vectors. Next, we will explain FFN, LayerNorm and MHAtt sequentially.

Positional Embeddings As in the Transformer model, there is no recurrent mechanism, the position of each input element needs to be explicitly distinguished by positional embeddings:

$$\mathbf{h}_t^0 = \mathbf{x}_t + \mathbf{PE}_t \tag{2.8}$$

where $\mathbf{PE}_t \in \mathbb{R}^d$ is the positional embedding for the t-th element in the input, formed by sine and cosine functions of different frequencies to indicate the position of each element in the sequence:

$$\mathbf{PE}_t[i] = \sin(t/10000^{2i/d}) \tag{2.9}$$

$$\mathbf{PE}_t[2i+1] = \cos(t/10000^{2i/d}) \tag{2.10}$$

where $\mathbf{PE}_t[i]$ indicates the i-th dimension of the vector. Because each dimension of the positional encoding corresponds to a sinusoid, for any fixed offset o, $\mathbf{PE}_{p+o}$ can be represented as a linear transformation of $\mathbf{PE}_p$, which enables the model to distinguish the positions of input elements.

Layer Normalization and Feed-forward Networks LayerNorm is the layer normalization operation proposed in Ba et al. (2016). Given a vector $\mathbf{h} \in \mathbb{R}^d$, the LayerNorm operation is calculated as:

$$a = \frac{1}{d}\sum_{i=1}^{d}\mathbf{h}[i];\ b = \frac{1}{d}\sum_{i=1}^{d}(\mathbf{h}[i]-a)^2 \tag{2.11}$$

$$LayerNorm(h) = \frac{\mathbf{h}-a}{\sqrt{b+\varepsilon}} \tag{2.12}$$

where a is the mean value of all dimensions of the input vector, and b is the variance of all dimensions of the input vector. ε is a small value to prevent division by zero.

FFN is a two-layer feed-forward network with ReLU as hidden activation function. Given a vector $\mathbf{h} \in \mathbb{R}^d$, the FFN operation is calculated as:

$$\mathrm{FFN}(\mathbf{h}) = \mathbf{W}_2\mathrm{max}(0, \mathbf{W}_1\mathbf{h} + \mathbf{b}_1) + \mathbf{b}_2 \tag{2.13}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{ff}\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times d_{ff}}$ are transformation weights; $\mathbf{b}_1 \in \mathbb{R}^{d_{ff}}$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are biases. $\max(\cdot)$ is the element-wise maximum operation.

Multi-head Attention MHAtt represents the multi-head attention mechanism which allows the model to jointly attend to information from different representation subspaces (at different positions). For the t-th vector $\mathbf{h}_t \in \mathbb{R}^d$ in a sequence $[\mathbf{h}_1, \cdots, \mathbf{h}_n]$, the single-head operation is firstly calculated as:

$$\begin{pmatrix} \mathbf{q}_t \\ \mathbf{k}_t \\ \mathbf{v}_t \end{pmatrix} = \begin{bmatrix} \mathbf{W}_q \\ \mathbf{W}_k \\ \mathbf{W}_v \end{bmatrix} \mathbf{h}_t \tag{2.14}$$

$$\mathbf{head}_t = \text{softmax}(\frac{\mathbf{q}_t \mathbf{k}_t^T}{\sqrt{d_k}})\mathbf{v} \tag{2.15}$$

$$\tag{2.16}$$

where $\mathbf{W}_q \in \mathbb{R}^{d_k \times d}$, $\mathbf{W}_k \in \mathbb{R}^{d_k \times d}$ and $\mathbf{W}_v \in \mathbb{R}^{d_k \times d}$ are transformation weights for obtaining the query vector $\mathbf{q}_t$, key vector $\mathbf{k}_t$ and value vector $\mathbf{v}_t$. Then a scaled dot operation is applied between $\mathbf{q}_k$ and all n key vectors of the input sequence, normalized by the softmax function. The obtained normalized distribution is used as the weights to sum the all value vectors, generating the final head vector $\mathbf{head}_t \in \mathbb{R}^{d_k}$.

MHAtt is the operation that simultaneously applies K single-head operations with different parameters, which will output K head vectors. These head vectors are then concatenated and linearly transformed:

$$\text{MHAtt}(\mathbf{h}_t) = \mathbf{W}_o \text{Concat}(\mathbf{head}_t^1, \cdots, \mathbf{head}_t^K) \tag{2.17}$$

The illustration of one layer of the Transformer model is shown in Figure 2.2. The output of Transformer models will be $[\mathbf{h}_1^N, \cdots, \mathbf{h}_n^N]$ from the top layer, where N is the number of stacked layers.

2.2 Neural Document Summarization

Starting from Kågebäck et al. (2014)'s work that showed sentences can be represented by continuous vectors and used for document summarization, neural network-based models have become popular architectures for the document summarization task. In the last five years, many novel neural network-based models have been proposed for document summarization as well as datasets for training these models. This section briefly reviews this recent work and distills common characteristics underlying these models. Although there is some common ground between the techniques used in extractive summarization and abstractive summarization, they still follow different research threads, and we will describe models for these two settings separately.

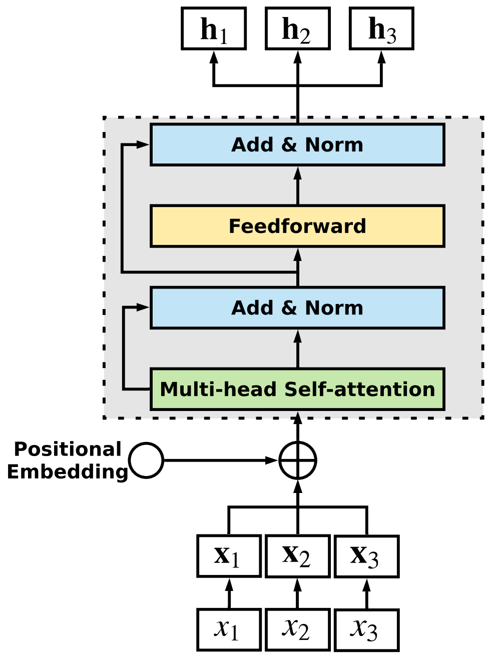


Figure 2.2: The illustration of a Transformer model with one layer. With three tokens as input, positional embeddings are first added to the input vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Then, as described in Section 2.1.2, the vectors go through a pipeline of operations including multi-head self-attention, layer normalization and feed-forward network.

2.2.1 Neural Extractive Summarization

Problem Formulation Let D denote a document containing sentences $[s_1, \cdots, s_n]$, where s_i is the i-th sentence in the document. Extractive summarization can be defined as the task of selecting a subset of sentences $[u_1, u_2 \cdots, u_m]$, where $u_m \in D$ and $m < n$, as the summary sentences. It is assumed that summary sentences represent the most important content of the document. Extractive summarizers usually have two basic modules: a module to build sentence representations and a module to select summary sentences based on their representations, taking into account the coverage and redundancy.

Sentence Representation Many previous studies focus on proposing new neural architectures for improving sentence representations. In early early work (Kågebäck et al., 2014), the sentences are simply represented by the summation of all word embeddings. Yin and Pei (2015) use convolutional neural networks (CNNs) over word embeddings to obtain sentence representations. In Cheng and Lapata (2016), sentence representations are obtained by using a CNN followed by an RNN. The CNN is applied as a sentence-level encoder to obtain sentence vectors, and then an LSTM is applied over these sentence vectors as a document-level encoder to model document-level contextual information, generating final sentence vectors. SummaRuNNer (Nallapati et al., 2017) employs a similar neural network with an RNN-based sentence-level encoder. Xiao and Carenini (2019) focus on long document summarization and propose to combine sentence representations with document representations, along with the LSTM-minus feature to represent local context.

Sentence Selection For selecting sentences, extractive summarizers usually follow two paradigms, sequence labelling and auto-regressive selection. Sequence labeling models (Cheng and Lapata, 2016; Nallapati et al., 2017) are equipped with a non-autoregressive classifier, assigning label $y_i \in \{0,1\}$ to each sentence s_i, indicating whether the sentence should be included in the summary. Auto-regressive selection models (Zhou et al., 2018; Narayan et al., 2018b) select the sentences in an autoregressive manner. When selecting a new summary sentence u_j, the model also takes selected summary sentences $[u_1, \cdots, u_{j-1}]$ as part of the input.

2.2.2 Neural Abstractive Summarization

Problem Formulation As many other generation problems in NLP (e.g., machine translation, text rewriting, dialogue generation), abstractive summarization can be framed as a sequence-to-sequence task, where the input is a sequence of words and the output is another sequence of words conditioned on the input. The neural encoder-decode model has proved extremely powerful when tackling these tasks: An encoder encodes the input text into a sequence of source vector representations, while a decoder generates the output text conditioned on these source vectors.

More formally, suppose we have a source-target pair (X,Y) (the source X is one or multiple documents, and the target Y is a summary in summarization tasks), where both the source and the target are represented as sequences of words $X = [x_1, \cdots, x_n]$ and $Y = [y_1, \cdots, y_m]$. Our goal in abstractive summarization task is to generate Y given X. Usually, the encoder will encode X to a sequence of continuous representations $H = [\mathbf{h}_1, \cdots, \mathbf{h}_m]$, and the decoder will generate the target summary token-by-token, in an auto-regressive manner, hence modeling the conditional probability: $p(y_1, \cdots, y_m|x_1, \cdots, x_n)$.

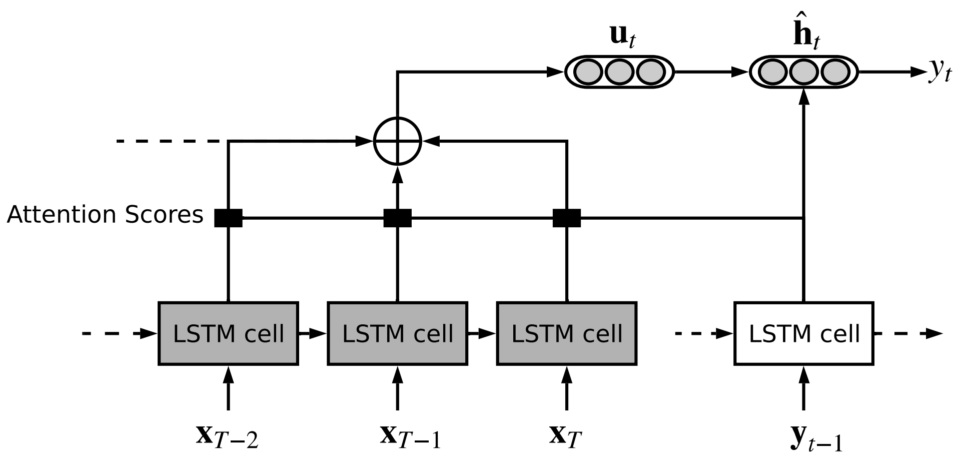


Figure 2.3: Illustration of the encoder-decoder attention module in an LSTM-based encoder-decoder model. The gray triangles are encoder LSTM cell and the white triangle is the decoder LSTM cell. The module computes a attention score for each source token based on its encoder hidden vector. The attention scores are then normalized and used as the weights of the weighted summation operation over all encoder hidden vectors, generating the context vector $\mathbf{u}_t$. And a new decoder hidden vector $\hat{\mathbf{h}}_t$ is computed based on $\mathbf{u}_t$ and $\mathbf{h}_t$ to calculate the probability of next decoded token y_t.

Transformer Encoder-Decoder Model The original encoder-decoder model was initially based on RNNs and applied in the machine translation task (Cho et al., 2014). After integrating the decoder-encoder attention mechanism (Bahdanau et al., 2015), it shows superiority over traditional translation models. For each decoding time step t, the decoder-encoder attention generates a context vector by applying a weighted summation over vectors of source tokens. As shown in Figure 2.3, with an LSTM-based encoder-decoder model, given a source input with T tokens, at decoding step t, the model computes the attention score $a_{t,k}$ of each source token x_k:

$$a_{t,k} = softmax(\mathbf{h}_k^e \cdot \mathbf{h}_t^d) \tag{2.18}$$

where $\mathbf{h}_k^e \in \mathbb{R}^d$ is the output hidden vector of the encoder and $\mathbf{h}_t^d \in \mathbb{R}^d$ is the output hidden vector of the decoder at last time step $t-1$, $a_{t,k}$ is the attention score indicating the importance of x_e at this decoding step. Then, the context vector $\mathbf{u}_t$ is calculated by a weighted summation over the encoder hidden vectors:

$$\mathbf{u}_t = \sum_{k=1}^{T} a_{t,k}\mathbf{h}_k^e \tag{2.19}$$

The probability of generating next token y_t from vocabulary V is computed as:

$$\hat{\mathbf{h}}_t^d = \tanh(\mathbf{W}_1 \mathbf{h}_t^d + \mathbf{W}_2 \mathbf{u}_t) \tag{2.20}$$

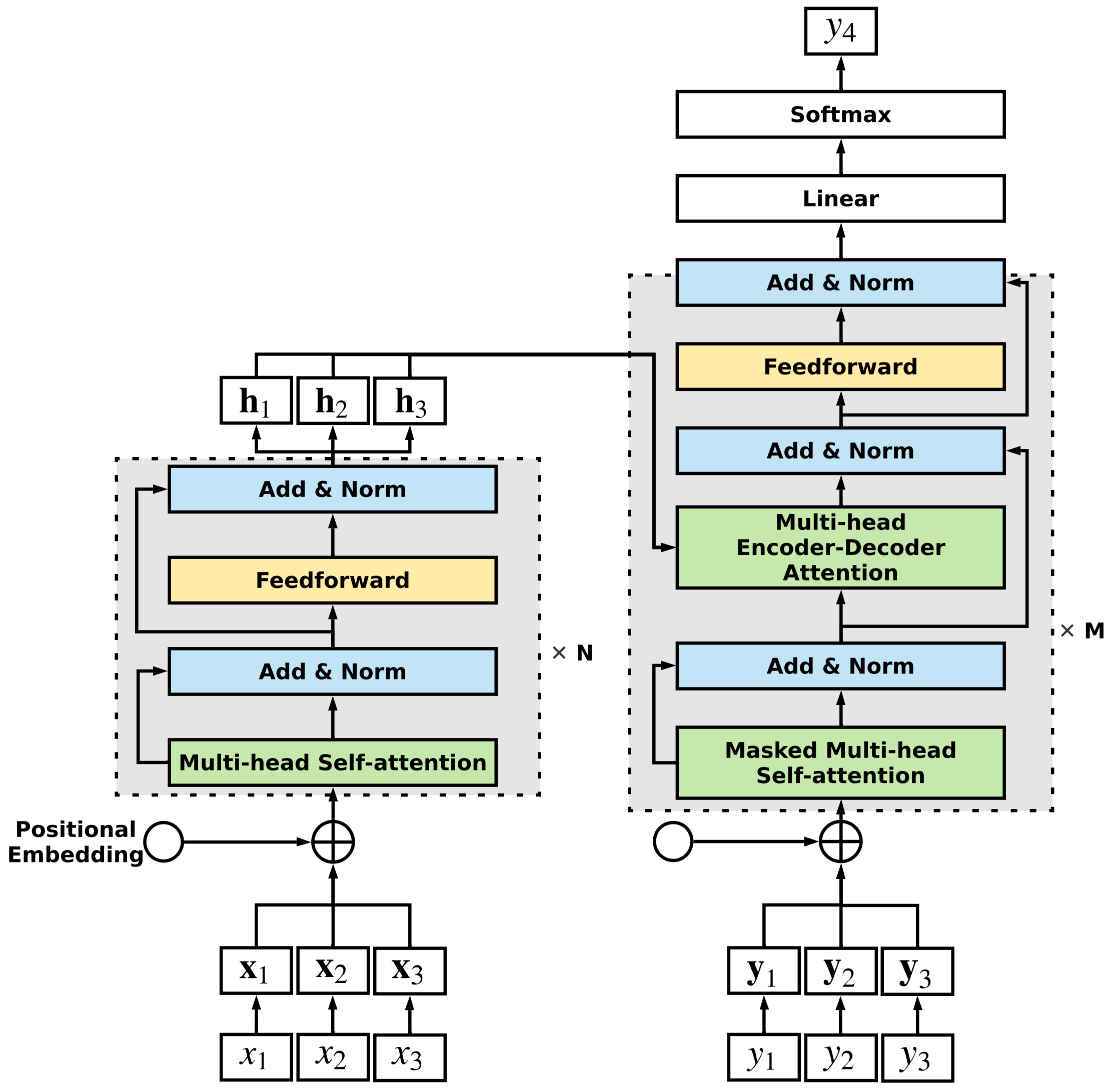
$$p(y_t | y_{<t-1}) = softmax(\mathbf{W}_o \hat{\mathbf{h}}_t^d) \tag{2.21}$$

where matrix $\mathbf{W}_o \in \mathbb{R}^{|V| \times d}$ and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are transformation weights.

Transformer encoder-decoder model (Vaswani et al., 2017), whose both encoder and decoder modules are based on Transformer models as described in previous sections, has been proved to be a even more powerful framework for generation tasks. In addition to the components in the vanilla Transformer layers, the Transformer decoder has an additional sub-layer which applied multi-head encoder-decoder attention over the top output vectors of the encoder. Also, since the target text is generated in an auto-aggressive fashion, the self-attention at time step t in the decoder only attend to words that have already been generated (before t), which is called masked self-attention, A detailed illustration of the Transformer encoder-decoder structure is shown in Figure 2.4.

2.3 Summary

In this chapter, we introduced the basis of neural network models for Natural Language Processing, and related models in neural document summarization. Two popular neural networks, namely the recurrent neural networks and Transformer models were introduced. We also formulated the extractive summarization task and the abstractive summarization task respectively, and introduced background knowledge of previous neural network-based summarizers. In the next chapter, we will explore how to learn structure-aware representations to better model a document.



y_4
Softmax
Linear
Add & Norm
Feedforward
Add & Norm
Multi-head Encoder-Decoder Attention
Add & Norm
Masked Multi-head Self-attention
× M
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$
Add & Norm
Feedforward
Add & Norm
Multi-head Self-attention
× N
Positional Embedding
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$
x_1 x_2 x_3
$\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$
y_1 y_2 y_3

Figure 2.4: Illustration of the Transformer encoder-decoder model. The encoder contains N Transformer layers and the decoder contains M Transformer layers. Input to the decoder are three tokens x_1, x_2, x_3. The decoder is running at time step $t = 3$, the input to the decoder are three already generated tokens y_1, y_2, y_3, and it is predicting the 4-th token y_4. Each head of the encoder-decoder attention generate a context vector of the source input.

Chapter 3

Document Modeling with Structured Attention

For summarizing one or multiple documents, one necessary process is to model the documents, generating representations that can be used for sentence extraction or text generation systems. Document modeling is also a fundamental task in Natural Language Processing, useful to various downstream applications including topic classification (Xie and Xing, 2013), summarization (Wolf and Gibson, 2006; Chen et al., 2016), sentiment analysis (Bhatia et al., 2015; He et al., 2018), question answering (Verberne et al., 2007), and machine translation (Meyer and Webber, 2013; Maruf and Haffari, 2018). In this chapter, we propose to model documents with structured attention mechanism, and generate better document representations. To show the model's effectiveness on document modeling, we experiment on the task of document classification with four datasets. And in the next chapter, we will show how this structured attention mechanism can be adapted to the summarization tasks.

Recent work provides strong evidence that better document representations can be obtained by incorporating structural knowledge (Bhatia et al., 2015; Yang et al., 2016; Ji and Smith, 2017; Mim et al., 2019). These structured representations are inspired by existing theories of discourse which have assumed several guises in the literature, such as trees in the style of Rhetorical Structure Theory (RST; Mann and Thompson, 1988), graphs (Lin et al., 2011; Wolf and Gibson, 2006), entity transitions (Barzilay and Lapata, 2008), or combinations thereof (Lin et al., 2011; Mesgar and Strube, 2015). The availability of discourse annotated corpora (Carlson et al., 2001; Prasad et al., 2008) has further led to the development of off-the-shelf discourse parsers (e.g., Feng and Hirst, 2012a; Liu and Lapata, 2017; Yu et al., 2018; Lin et al., 2019), and the common use of trees as representations of document structure. For example, Bhatia et al. (2015) improve document-level sentiment analysis by reweighing discourse units based on the depth of RST trees, whereas Ji and Smith (2017) show that a recursive neural network built on the output of an RST parser benefits text categorization in learning representations that focus on salient content.

Linguistically motivated representations of document structure rely on the availability of annotated corpora as well as a wider range of standard NLP tools (e.g., tokenizers, pos-taggers, syntactic parsers). Unfortunately, the reliance on labeled data, which is both difficult and highly expensive to produce, presents a major obstacle to the widespread use of discourse structure for document modeling. Moreover, despite recent advances in discourse processing, the use of an external parser often leads to pipeline-style architectures where errors propagate to later processing stages, affecting model performance.

It is therefore not surprising that there have been attempts to induce document representations directly from data without recourse to a discourse parser or additional annotations. The main idea is to obtain *hierarchical* representations by first building representations of sentences, and then aggregating those into a document representation (Tang et al., 2015a,b; Jiang et al., 2019). Yang et al. (2016) further demonstrate how to implicitly inject structural knowledge onto the representation using an attention mechanism (Bahdanau et al., 2015) which acknowledges that sentences are differentially important in different contexts. Their model learns to pay more or less attention to individual sentences when constructing the representation of the document.

Our work focus on learning deeper structure-aware document representations, drawing inspiration from recent efforts to empower neural networks with a structural bias (Cheng et al., 2016). Kim et al. (2017) introduce structured attention networks which are generalizations of the basic intra-sentential procedure, allowing to learn sentential representations while attending to partial segmentations or subtrees. Specifically, they take into account the dependency structure of a sentence by viewing the attention mechanism as a graphical model over latent variables. They first calculate unnormalized pairwise attention scores for all tokens in a sentence and then use the inside-outside algorithm (Baker, 1979) to normalize the scores with the marginal probabilities of a dependency tree. Without recourse to an external parser, their model learns meaningful task-specific dependency structures, achieving competitive results in several sentence-level tasks. More details about the intra-sentential attention mechanism and Kim et al. (2017)'s structured attention network will be presented in Section 3.1.

1 The next time you hear a Member of Congress moan about the deficit, consider what Congress did Friday.
2 The Senate, 84-6, voted to increase to $124,000 the ceiling on insured mortgages from the FHA, which lost $4.2 billion in loan defaults last year.
3 Then, by voice vote, the Senate voted a porkbarrel bill, approved Thursday by the House, for domestic military construction.
4 Compare the Bush request to what the Senators gave themselves:

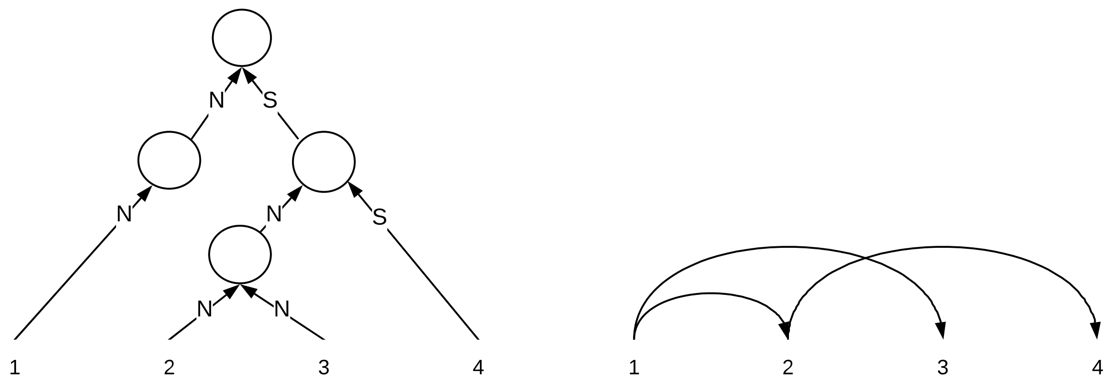


Figure 3.1: An example document with splitted sentences. Below the text, the left subfigure is the tree structure of the document analyzed in the style of Rhetorical Structure Theory (Mann and Thompson, 1988), and the right part represents a converted dependency tree following the conversion algorithm of Hayashi et al. (2016).

However, for document modeling, this approach has two drawbacks. Firstly, it does not consider non-projective dependency structures, which are relatively common in document-level discourse analysis (Lee et al., 2006; Hayashi et al., 2016). As illustrated in Figure 3.1, when converted from a RST tree structure, the dependency tree structure of a document can be flexible and the dependency edges may cross. Secondly, the inside-outside algorithm involves a dynamic programming process which is difficult to parallelize, making it impractical for modeling long documents.

In this chapter, we propose a new model for representing documents while automatically learning richer structural dependencies. Using a variant of Kirchhoff's Matrix-Tree Theorem (Tutte, 1984), our model implicitly considers non-projective dependency tree structures. We keep each step of the learning process differentiable, so the model can be trained in an end-to-end fashion and induce discourse information that is helpful to specific tasks without an external parser. The inside-outside model of Kim et al. (2017) and our model both have a $O(n^3)$ worst case complexity. However, major operations in our approach can be parallelized efficiently on GPU computing hardware. Although our primary focus is on document modeling, there is nothing inherent in our model that prevents its application to individual sentences. Advantageously, it can induce non-projective structures which are required for representing languages with free or flexible word order (McDonald and Satta, 2007).

Our contributions in this chapter are threefold: 1) a model for learning document representations whilst taking structural information into account; 2) an efficient training procedure which allows to compute document level representations of arbitrary length; 3) and a large scale evaluation study showing that the proposed model performs competitively against strong baselines while having the potential to induce meaningful intermediate structures.

3.1 Related Work

In this section, we describe how previous work uses the intra-sentential attention mechanism for representing individual sentences. The key idea is to capture the interaction between tokens within a sentence, generating a context representation for each word with weak structural information. This type of intra-sentential attention encodes relationships between words within each sentence and differs from *inter-sentence* attention which has been widely applied to sequence transduction tasks like machine translation (Bahdanau et al., 2015) and learns the latent alignment *between* source and target sequences. Different from the multi-head self-attention mechanism in Transformer, this intra-sentential attention is simpler, with only one head, and without the linear transformation to query, key and value (see Section 2.1.2 for more details.).

Figure 3.2 provides a schematic view of the simple intra-sentential attention. Given a sentence represented as a sequence of n word vectors $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, for each word pair $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, the attention score $\mathbf{a}_{ij}$ is estimated as:

$$\mathbf{f}_{ij} = F(\mathbf{x}_i, \mathbf{x}_j) \tag{3.1}$$

$$\mathbf{a}_{ij} = \frac{\exp(\mathbf{f}_{ij})}{\sum_{k=1}^{n} \exp(\mathbf{f}_{ik})} \tag{3.2}$$

where $F(\cdot)$ is a function for computing the unnormalized score $\mathbf{f}_{ij}$ which is then normalized by calculating a probability distribution $\mathbf{a}_{ij}$. Individual words collect information from their context based on $\mathbf{a}_{ij}$ and obtain a context representation:

$$\mathbf{u}_i = \sum_{j=1}^{n} \mathbf{a}_{ij} \mathbf{x}_j \tag{3.3}$$

where attention score $\mathbf{a}_{ij}$ indicates the (dependency) relation between the i-th and the j-th words and how information from $\mathbf{x}_j$ should be fed into $\mathbf{x}_i$. $\mathbf{u}_i$ denotes the updated vector for the i-th token with context information.

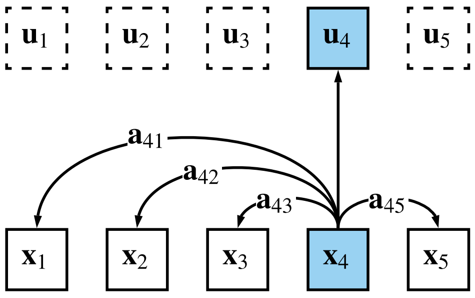


Figure 3.2: Intra-sentential attention mechanism; $\mathbf{a}_{ij}$ denotes the normalized attention score between token vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. $\mathbf{u}_i$ denotes the updated vector for the i-th token with context information.

Despite successful application of the above intra-sentential attention in sentiment analysis (Cheng et al., 2016) and entailment recognition (Parikh et al., 2016), the structural information under consideration is shallow, limited to word-word dependencies. Since attention is computed as a simple probability distribution, it cannot capture more elaborated structural dependencies such as trees (or graphs). Kim et al. (2017) induce richer internal structure by imposing structural constraints on the probability distribution computed by the intra-sentential attention mechanism. Specifically, they normalize $\mathbf{f}_{ij}$ into the marginal probabilities of projective dependency trees using the inside-outside algorithm (Baker, 1979), which considers the parsing process as a graph-based Conditional Random Field, implemented in a dynamic programming process:

$$\mathbf{f}_{ij} = F(\mathbf{x}_i, \mathbf{x}_j) \tag{3.4}$$

$$\mathbf{a} = \textit{inside-outside}(\mathbf{f}) \tag{3.5}$$

$$\mathbf{u}_i = \sum_{j=1}^{n} \mathbf{a}_{ij}\mathbf{x}_j \tag{3.6}$$

where $\mathbf{a}$ is a matrix that is of the same shape with $\mathbf{f}$, and its entry $\mathbf{a}_{ij}$ indicates the marginal probability of forming a dependency edge from i-th token to the j-th token in a projective dependency tree.

This process is differentiable, so the model can be trained end-to-end and learn structural information without relying on a parser. However, efficiency is a major issue, since the inside-outside algorithm has time complexity $O(n^3)$ (where n represents the number of tokens) and does not lend itself to easy parallelization.

3.2 Modeling Documents with Structural Bias

In this section we present our document representation model. We follow previous work (Tang et al., 2015a; Yang et al., 2016; Jiang et al., 2019) in modeling documents *hierarchically* by first obtaining representations for sentences and then composing those into a document representation. Structural information is taken into account while learning representations for both sentences and documents and an structured attention mechanism is applied on both words within a sentence and sentences within a document. The general idea is to force pair-wise attention between text units to form a non-projective dependency tree, and automatically induce this tree for different natural language processing tasks in a differentiable way. Instead of relying on the softmax function to obtain the attention weights as in intra-sentential attention, we refer to the Matrix-Tree Theorem (Kirchhoff, 1847; Tutte, 1984) to calculate the marginal probabilities of dependency trees as the attention weights, and with the normalized attention weights, we apply the same weighted summation operation as in the intra-sentential attention to obtain context represenations. With the new structured attention mechanism, we are able to incorporate structural bias into neural document modeling systems. In the following, we first describe how the structured attention mechanism is applied to sentences, and then move on to present our document-level model.

3.2.1 Sentence Model

Let $T = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$ denote a sentence containing a sequence of words, each represented by a vector $\mathbf{x}$, which can be pre-trained on a large corpus. As described in section 2.1.1, Long Short-Term Memory Neural Networks (LSTMs; Hochreiter and Schmidhuber, 1997) have been successfully applied to various sequence modeling tasks ranging from machine translation (Bahdanau et al., 2015; Wu et al., 2016), to speech recognition (Graves et al., 2013), and image caption generation (Xu et al., 2015). In this chapter we use bidirectional LSTMs as a way of representing elements in a sequence (i.e., words or sentences) together with their contexts, capturing the element and an "infinite" window around it. Specifically, we run a bidirectional LSTM over sentence T, and take the output vectors $[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n]$ as the representations of words in T, where $\mathbf{h}_i \in \mathbb{R}^d$ is the output vector for word $\mathbf{x}_i$ based on its context.

We then exploit the structure of T which we induce based on a structured attention mechanism detailed below to obtain more precise representations. Inspired by recent work (Daniluk et al., 2017; Miller et al., 2016), which shows that the conventional

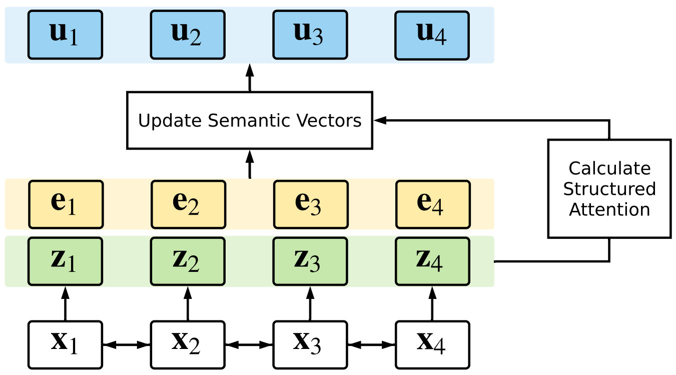


Figure 3.3: Sentence representation model: $\mathbf{x}_i$ is the input vector for the i-th word, $\mathbf{e}_i$ and $\mathbf{z}_i$ are semantic and structure vectors, respectively. $\mathbf{e}$ are used to calculate unnormalized attention scores, which are latter normalized by structured attention mechanism. $\mathbf{z}$ are used as semantic vectors, which are updated by the structured attention mechanism, generating the updated vectors $\mathbf{u}$ with context information.

way of using LSTM output vectors for calculating both attention and encoding word semantics is overloaded and likely to cause performance deficiencies, we decompose the LSTM output vector in two parts, where we take the first half of the dimensions as vector $\mathbf{e}_i$, and take the remaining half of the dimensions as vector $\mathbf{z}_i$:

$$[\mathbf{e}_i, \mathbf{z}_i] = \mathbf{h}_i \tag{3.7}$$

where $d_h = \frac{d}{2}$; $\mathbf{e}_i \in \mathbb{R}^{d_h}$ is the semantic vector, encoding semantic information for specific tasks, and $\mathbf{z}_i \in \mathbb{R}^{d_h}$, the structure vector, is used to calculate structured attention.

We use a series of operations based on the Matrix-Tree Theorem (Kirchhoff, 1847; Tutte, 1984) to incorporate the structural bias of non-projective dependency trees into the attention weights. We constrain the probability distributions $\mathbf{a}_{ij}$ (see Equation (3.2)) to be the posterior marginals of a dependency tree structure. We then use the normalized structured attention, to build a context vector for updating the semantic vector of each word, obtaining new representations $[\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n]$. An overview of the model is presented in Figure 3.3. We describe the structured attention mechanism in detail in the following section.

3.2.2 Structured Attention Mechanism

Dependency representations of natural language are a simple yet flexible mechanism for encoding words and their syntactic relations through directed graphs. Much work in descriptive linguistics (Tesniére, 1959; Melčuk, 1988) has advocated their suitability for representing syntactic structure across languages. A primary advantage of dependency representations is that they have a natural mechanism for representing discontinuous constructions arising from long distance dependencies or free word order through non-projective dependency edges.

More formally, building a dependency tree amounts to finding latent variables z_{ij} for all $i \neq j$, where word i is the parent node of word j, under some global constraints, amongst which the single-head constraint is the most important, since it forces the structure to be a rooted tree. We use a variant of Kirchhoff's Matrix-Tree Theorem (Kirchhoff, 1847; Tutte, 1984; Koo et al., 2007) to calculate the marginal probability of each dependency edge $p(z_{ij} = 1)$ of a non-projective dependency tree, and this probability is used as the attention weight that decides how much information is collected from child unit j to the parent unit i.

We first calculate unnormalized attention scores $\mathbf{f}_{ij}$ with structure vector $\mathbf{z}$ (see Equation (3.7)) via a bilinear function:

$$\mathbf{t}_p = \tanh(\mathbf{W}_p \mathbf{z}_i) \tag{3.8}$$

$$\mathbf{t}_c = \tanh(\mathbf{W}_c \mathbf{z}_j) \tag{3.9}$$

$$\mathbf{f}_{ij} = \mathbf{t}_p^T \mathbf{W}_a \mathbf{t}_c \tag{3.10}$$

where $\mathbf{W}_p \in \mathbb{R}^{d_h \times d_h}$ and $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_h}$ are the weights for building the representation of parent and child nodes. $\mathbf{W}_a \in \mathbb{R}^{d_h \times d_h}$ is the weight for the bilinear transformation. $\mathbf{f} \in \mathbb{R}^{n \times n}$ can be viewed as a weighted adjacency matrix for a graph G with n nodes where each node corresponds to a word in a sentence. We also calculate the root score $\mathbf{f}_i^r$, indicating the unnormalized possibility of a node being the root:

$$\mathbf{f}_i^r = \mathbf{W}_r \mathbf{z}_i \tag{3.11}$$

where $\mathbf{W}_r \in \mathbb{R}^{1 \times d_h}$. We calculate $p(z_{ij} = 1)$, the marginal probability of the dependency edge, following Koo et al. (2007):

$$\mathbf{A}_{ij} = \begin{cases} 0 & \text{if } i = j \\ \exp(\mathbf{f}_{ij}) & \text{otherwise} \end{cases} \tag{3.12}$$

$$\mathbf{L}_{ij} = \begin{cases} \sum_{i'=1}^{n} \mathbf{A}_{i'j} & \text{if } i = j \\ -\mathbf{A}_{ij} & \text{otherwise} \end{cases} \tag{3.13}$$

$$\bar{\mathbf{L}}_{ij} = \begin{cases} \exp(\mathbf{f}_i^r) & i = 1 \\ \mathbf{L}_{ij} & i > 1 \end{cases} \tag{3.14}$$

$$\begin{aligned} p(z_{ij} = 1) &= (1 - \delta_{1,j})\mathbf{A}_{ij}[\bar{\mathbf{L}}^{-1}]_{jj} \\ &\quad - (1 - \delta_{i,1})\mathbf{A}_{ij}[\bar{\mathbf{L}}^{-1}]_{ji} \\ p(root(i)) &= \exp(f_r^i)[\bar{\mathbf{L}}^{-1}]_{i1} \end{aligned} \tag{3.15}$$

where $1 \leq i \leq n, 1 \leq j \leq n$. $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix for graph G and $\bar{\mathbf{L}} \in \mathbb{R}^{n \times n}$ is a variant of $\mathbf{L}$ that takes the root node into consideration, and δ is the Kronecker delta. The key for the calculation to hold is for $\mathbf{L}^{ii}$, the minor of the Laplacian matrix $\mathbf{L}$ with respect to row i and column i, to be equal to the sum of the weights of all directed spanning trees of G which are rooted at i. $p(z_{ij} = 1)$ is the marginal probability of the dependency edge between the i-th and j-th words. $p(root(i) = 1)$ is the marginal probability of the i-th word headed by the root of the tree. Details of the proof can be found in Koo et al. (2007).

We denote the marginal probabilities $p(z_{ij} = 1)$ as $\mathbf{a}_{ij}$ and $p(root(i))$ as $\mathbf{a}_i^r$. This can be interpreted as attention scores which are constrained to converge to a structured object, a non-projective dependency tree, in our case. We update the semantic vector $\mathbf{e}_i$ of each word with structured attention:

$$\mathbf{p}_i = \sum_{k=1}^{n} \mathbf{a}_{ki}\mathbf{e}_k + \mathbf{a}_i^r \mathbf{e}_{root} \tag{3.16}$$

$$\mathbf{c}_i = \sum_{k=1}^{n} \mathbf{a}_{ik}\mathbf{e}_i \tag{3.17}$$

$$\mathbf{u}_i = \tanh(\mathbf{W}_r[\mathbf{e}_i, \mathbf{p}_i, \mathbf{c}_i]) \tag{3.18}$$

where $\mathbf{p}_i \in \mathbb{R}^{d_h}$ is the context vector gathered from possible parents of u_i and $\mathbf{c}_i \in \mathbb{R}^{d_h}$ the context vector gathered from possible children, and $\mathbf{e}_{root}$ is a special embedding for the root node. The context vectors are concatenated with $\mathbf{e}_i$ and transformed with weights $\mathbf{W}_r \in \mathbb{R}^{d_h \times 3d_h}$ to obtain the updated semantic vector $\mathbf{u}_i \in \mathbb{R}^{d_h}$ with rich structural information (see Figure 3.3).

3.2.3 Document Model

We build document representations hierarchically: sentences are composed of words and documents are composed of sentences. Composition on the document level also makes use of structured attention in the form of a dependency graph. Dependency-based representations have been previously used for developing discourse parsers (Li et al., 2014b; Hayashi et al., 2016) and in applications such as summarization (Hirao et al., 2013b).

As illustrated in Figure 3.4, given a document with n sentences $[s_1, s_2, \cdots, s_n]$, for each sentence s_i, the input is a sequence of word embeddings $[\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{im}]$, where m is the number of tokens in s_i. By feeding the embeddings into a sentence-level bi-LSTM and applying the proposed structured attention mechanism, we obtain the updated semantic vector $[\mathbf{u}_{i1}, \mathbf{u}_{i2}, \cdots, \mathbf{u}_{im}]$. Then an average pooling operation produces a fixed-length vector $\mathbf{s}_i$ for each sentence. Analogously, we view the document as a sequence of sentence vectors $[\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n]$ whose embeddings are fed to a document-level bi-LSTM. Application of the structured attention mechanism creates new semantic vectors $[\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]$ and another pooling operation yields the final document representation $\mathbf{y}$.

3.2.4 End-to-End Training

Our model can be trained in an end-to-end fashion since all operations required for computing structured attention and using it to update the semantic vectors are differentiable. In contrast to in Kim et al. (2017), training can be done efficiently. The major complexity of our model lies in the computation of the gradients of the inverse matrix. Let $\mathbf{A}$ denote a matrix depending on a real parameter x; assuming all component functions in $\mathbf{A}$ are differentiable, and $\mathbf{A}$ is invertible for all possible values, the gradient of $\mathbf{A}$ with respect respect to x is:

$$\frac{d\mathbf{A}^{-1}}{dx} = -\mathbf{A}^{-1}\frac{d\mathbf{A}}{dx}\mathbf{A}^{-1} \tag{3.19}$$

Multiplication of the three matrices and matrix inversion can be computed efficiently on modern parallel hardware architectures such as GPUs. In our experiments, computation of structured attention takes only 1/10 of training time (on a Nvidia GTX 1080 machine).

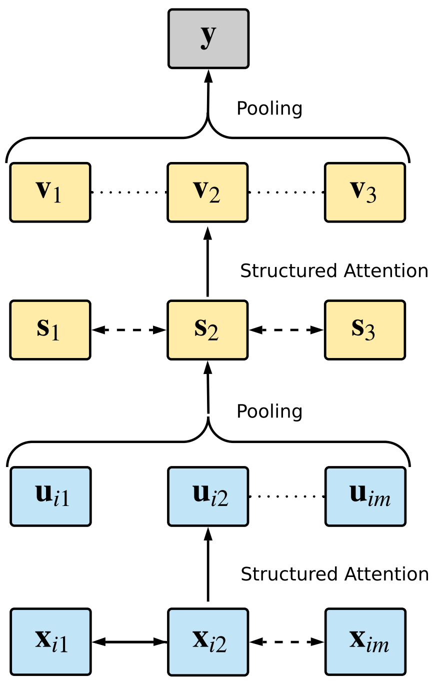


Figure 3.4: Document representation model with structured attention. The document is encoded first from tokens to sentences and then from sentences to the complete document. $\mathbf{x}_{ij}$ and $\mathbf{u}_{ij}$ are input vector and hidden vector for the j-th token in the i-th sentence. $\mathbf{s}_i$ and $\mathbf{v}_i$ are input vector and hidden vector for the i-th sentence.

3.3 Experiments and Analysis

In this section we present our experiments for evaluating the performance of our model. Since sentence representations constitute the basic building blocks of our document model, we first evaluate the performance of structured attention on a sentence-level task, namely natural language inference. We then assess the document-level representations obtained by our model on a variety of classification tasks representing documents of different length, subject matter, and language. Our code is available at `https://github.com/nlpyang/structured`.

In this section, we evaluate our document-level model on a variety of classification tasks. We selected four datasets which we describe below. Table 3.1 summarizes some statistics for each dataset. Examples of input texts and labels are shown in Table 3.2.

Yelp reviews were obtained from the 2013 Yelp Dataset Challenge. This dataset contains restaurant reviews, each associated with human ratings on a scale from 1 (negative) to 5 (positive) which we used as gold labels for sentiment classification; we

Dataset	#class	#docs	#s/d	#w/d
Yelp	5	335K	8.9	151.6
IMDB	10	348K	14.0	325.6
CZ Movies	3	92K	3.5	51.2
Debates	2	1.6K	22.7	519.2

Table 3.1: Dataset statistics; #class is the number of classes per dataset, #docs denotes the number of documents; #s/d and #w/d represent the average number of sentences and words per document.

followed the preprocessing introduced in Tang et al. (2015a) and report experiments on their training, development, and testing partitions (80/10/10).

IMDB reviews were obtained from Diao et al. (2014), who randomly crawled reviews for 50K movies. Each review is associated with user ratings ranging from 1 to 10.

Czech reviews were obtained from Brychcın and Habernal (2013). The dataset contains reviews from the Czech Movie Database[1] each labeled as positive, neutral, or negative. We include Czech in our experiments since it has more flexible word order compared to English, with non-projective dependency structures being more frequent. Experiments on this dataset perform 10-fold cross-validation following previous work (Brychcın and Habernal, 2013).

Congressional floor debates were obtained from a corpus originally created by Thomas et al. (2006) which contains transcripts of U.S. floor debates in the House of Representatives for the year 2005. Each debate consists of a series of speech segments, each labeled by the vote ("yea" or "nay") cast for the proposed bill by the speaker of each segment. We used the pre-processed corpus from Yogatama and Smith (2014).[2]

Following previous work (Yang et al., 2016), we only retained words appearing more than five times in building the vocabulary and replaced words with lesser frequencies with a special UNK token. Word embeddings were initialized by training word2vec (Mikolov et al., 2013) on the training and validation splits of each dataset. In our experiments, we set the word embedding dimension to be 200 and the hidden size for the sentence-level and document-level LSTMs to 100 (the dimensions of the

[1]`http://www.csfd.cz/`

[2]`http://www.cs.cornell.edu/~ainur/data.html`

Dataset	Text	Label
Yelp	Too bad, but you knew it would happen the douche bags would find this place out and now, the hoards and the masses of db's are roaming about this groovy, mid-century modern that tosses a pretty good pie but sadly the ol' db factor has ruined the place, for this guy it's no fault of the owners they did their very best but now the dicks and boners have descended from the east and the west side, in their true religions to prey on the cougars, et al who also have seeked out the parlor the kitty's have started to crawl around, looking for action made up, for making out and you know the rules of attraction will keep the c's and db's, turning out . I'll miss you, parlor pizza enjoyed our little affair, but i'm off, seeking douche-free pastures man !! ... i used to love it there ...	3
IMDB	This is a very good movie. Not the best of TNG's movies, but certainly still better then Kirk's movies, this one starts with a new race coming into the Federation ending up with a race that has people over 300 years old.Not much is there that I can tell you without giving away the story, but there is a lot of tension because the Federation wants to transport an entire race of people to another planet to get the particles that grant the long life that are around that planet. It turns out that Picard and his crew have to go against the very Federation they have pledged their life to, to save a planet full of people.With the opportunity to see Riker with a shaved face for possibly the last time, seeing Worf sing ”A British Tar” for the one and only time, this is truly a classic ST movie. rent it today if you can.	8
CZ Movies	Skrz naskrz uniktn pohled na to, jak me se lovkem zacloumat prakticky ze dne na den nabyt slva. A na Thoma Yorka se vichno ostatn z Radiohead vyrovnali s tm, e je lid miluj a jsou schopni pro jejich hudbu udlat takka cokoliv (j jsem jeden z nich). Nezvykl vizuln ztvrnn dokonale sed k celmu vtvarnmu kultu kolem Radiohead. Navc jsou zde i nkter ukzky z naten klip (pedevm m srdcovky No Surprises, co je podle m snad i nejlep videoklip vech dob) a nahrvn ve studiu. Zkrtka jak to asi vypad s kadou kapelou, kter dl to co dl s lskou. Zde konkrtn bhem tour po vydn desky OK Computer v roce 1997.	Negative

Table 3.2: Examples of input texts and labels for four experimented document classification datasets.

Debates	mr. speaker, i believe that section 122 of this bill is an important public policy statement that says corporate executives who are not properly funding the pension plans of their employees should not be feathering their own nests with overly generous retirement packages. currently, the bill penalizes employers who fund executive compensation if the sponsor's employee defined plans are less than 60 percent funded. my concern is that by setting this threshold too low, we are not discouraging them enough from being irresponsible with the retirement security of their employees while they take care of their own retirement packages. i ask the chairman to work with me in conference to increase the threshold to at least 80 percent so that we encourage executives to take their pension funding obligations more seriously, not leave their defined benefit plan beneficiaries and, indeed, the pbgc and taxpayers on the hook. mr. speaker, i thank the gentleman for his response.	Nay

Table 3.2 Continued

semantic and structure vectors were set to 75 and 25, respectively). We used a mini-batch size of 32 during training and documents of similar length were grouped in one batch. Parameters were optimized with Adagrad (Duchi et al., 2011), the learning rate was set to 0.05. We used L_2 regularization for all parameters except word embeddings with regularization constant set to $1e^{-4}$. Dropout was applied on the input and output layers with dropout rate 0.3.

Our results are summarized in Table 3.3. We compared our model against several related models covering a wide spectrum of representations including word-based ones (e.g., paragraph vector and CNN models) as well as hierarchically composed ones (e.g., a CNN or LSTM provides a sentence vector and then a recurrent neural network combines the sentence vectors to form a document level representation for classification). Previous state-of-the-art results on the three review datasets were achieved by the hierarchical attention network of Yang et al. (2016), which models the document hierarchically with two GRUs and uses an attention mechanism to weigh the importance of each word and sentence. On the debates corpus, Ji and Smith (2017) obtained best results with a recursive neural network model operating on the output of an RST parser. Table 3.3 presents three variants[3] of our model, one with structured attention

[3]We do not report comparisons with the inside-outside approach on document classification tasks

Models	Yelp	IMDB	CZ Movies	Debates	θ
Feature-based classifiers	59.8	40.9	78.5	74.0	—
Paragraph vector (Tang et al., 2015a)	57.7	34.1	—	—-	—
CNN (Tang et al., 2015a)	59.7	—	—	—	—
Convolutional gated RNN (Tang et al., 2015a)	63.7	42.5	—	—	—
LSTM gated RNN (Tang et al., 2015a)	65.1	45.3	—	—	—
RST-based NN (Ji and Smith, 2017)	—	—	—	75.7	—
75D HAN (Yang et al., 2016)	68.2	**49.4**	80.8	74.0	273K
75D No Attention	66.7	47.5	80.5	73.7	330K
100D Simple Attention	67.7	48.2	81.4	75.3	860K
100D Structured Attention (sentence-level)	68.0	48.8	81.5	74.6	842K
100D Structured Attention (document-level)	67.8	48.6	81.1	75.2	842K
100D Structured Attention (both levels)	**68.6**	49.2	**82.1**	**76.5**	860K

Table 3.3: Test accuracy on four datasets and number of parameters θ (excluding embeddings). Regarding feature-based classification methods, results on Yelp and IMDB are taken from Tang et al. (2015a), on CZ movies from Brychcın and Habernal (2013), and Debates from Yogatama and Smith (2014). Wherever available we also provide the size of the recurrent unit (LSTM or GRU).

on the sentence level, another one with structured attention on the document level and a third model which employs attention on both levels. As can be seen, the combination is beneficial achieving best results on three out of four datasets. Furthermore, structured attention is superior to the simpler word-to-word attention mechanism, and both types of attention bring improvements over no attention. The structured attention approach is also very efficient, taking only 20 minutes for one training epoch on the largest dataset.

3.3.1 Analysis of Induced Structures

To gain further insight on structured attention, we inspected the dependency trees it produces. Specifically, we used the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) to extract the maximum spanning tree from the attention scores. We report various statistics on the characteristics of the induced trees across different tasks

due to its prohibitive computation cost leading to 5 hours of training for one epoch.

		Yelp	IMDB	CZ Movie	Debates
Projective		79.6%	74.9%	82.8%	62.4%
Height		2.81	3.34	1.50	3.58
Nodes	depth 2	15.1%	13.6%	25.7%	12.8%
	depth 3	55.6%	46.8%	57.1%	30.2%
	depth 4	22.3%	32.5%	11.3%	40.8%
	depth 5	3.2%	4.1%	5.8%	14.8%

Table 3.4: Descriptive statistics for induced document-level dependency trees across four testsets.

and datasets.

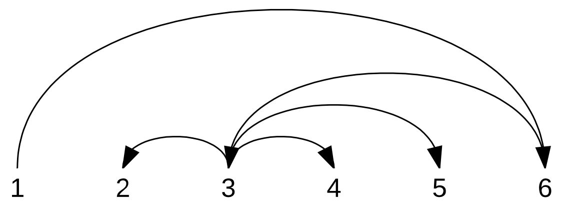
Table 3.4 summarizes various characteristics of these trees. For most datasets, document-level trees are not very deep, they mostly contain up to nodes of depth 3. This is not surprising as the documents are relatively short (see Table 3.1) with the exception of debates which are longer and the induced trees more complex. The fact that most documents exhibit simple discourse structures is further corroborated by the large number (over 70%) of projective trees induced on Yelp, IMDB, and CZ Movies datasets.

Figure 3.5 shows examples of document-level trees taken from Yelp dataset. In the tree of Figure 3.5a, most edges are examples of the "elaboration" discourse relation, i.e., the child presents additional information about the parent. The tree of Figure 3.5b is non-projective, the edges connecting sentences 1 and 4 and 3 and 5 cross. We also show an example that is not with well-formed tree structures in Figure 3.5c, where sentence 2 is the root and all other sentences are direct children of the root. This is especially common when the document length is long, since the model is learning the distribution of trees but not concrete tree structures and long inputs will smoothen the marginal probability distribution.

3.4 Summary

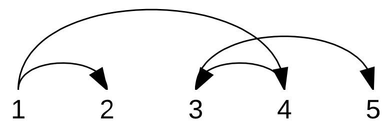
In this chapter, we proposed a new model for representing documents while automatically learning rich structural dependencies. Our model normalizes intra-attention scores with the marginal probabilities of a non-projective dependency tree based on a matrix inversion process. Each operation in this process is differentiable and the

1 first of all, i did not expect to come into a cafeteria style eatery.
2 they serve the basics of bbq, nothing too fancy.
3 a few appetizers and side options, like cheesy potatoes, baked mac 'n' 4 cheese, fresh corn bread, etc..
4 all were very tasty.
5 for entree, they have a wide variety of meats and combos and samplers.
6 overall, this is a great place,... meat was well prepared, a little pricey for what i was expecting.



(a)

1 great instruction by ryan
2 clean workout facility and friendly people
3 they have a new student membership for 60 per month and classes are mon , weds and fri 6pm 7pm
4 it 's definitely worth money if you want to learn brazilian jiu jitsu
5 i usually go to classes on mondays and fridays , and it 's the best workout i 've had in years

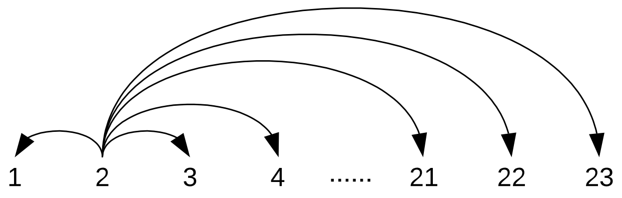


(b)

1 I used to love this yogurtology (see : glowing review below.)
2 But now?
3 I'll never be back.
4 Over the past month and a half, I've gone to yogurtology 3 times (I'd be there more, but i live a solid 20 min.

⋮

21 Given all the other frozen yogurt shops closer to where I live, this is an incredibly stupid move for yogurtology.
22 Hire some real workers and fire this clown who obviously doesn't care enough to do his job right.
23 I will never, ever be back.



(c)

Figure 3.5: Example tree outputs generated by the Chu-Liu-Edmonds algorithm from Yelp testset.

model can be trained efficiently end-to-end, while inducing structural information. We applied this approach to model documents hierarchically, incorporating both sentence and document-level structure. Experiments on sentence and document modeling tasks show that the representations learned by our model achieve competitive performance against strong comparison systems. Analysis of the induced tree structures revealed that they are meaningful, albeit different from linguistics ones, without ever exposing the model to linguistic annotations or an external parser.

Document modeling, as a fundamental component of document-level NLP tasks, can be usefully employed in document summarization models. In this next chapter, we will describe how to adapt structured attention for extractive document summarization.

Chapter 4

Single-Document Summarization as Tree Induction

As introduced in the previous chapter, the structured attention mechanism could be an effective tool for modelling documents with structural constraints by neural models without recourse to an external parser. For text summarization, document structure analysis has also been proven useful. Previous studies (Marcu, 1999) have shown that, document structure plays an important role in conveying important content as a writer and perceiving important content as a reader. And the structural information of the input document could help a neural summarizer to generate more precise summaries (Hirao et al., 2013a; Yoshida et al., 2014; Gerani et al., 2014). However, as we discussed in Chapter 1, the reliance on a parser has presented obstacles and challenges to the wide application of document structure analysis in summarization tasks.

In this chapter, we attempt to introduce structure analysis into neural network models for extractive summarization. And the structured attention mechanism could be a useful framework to achieve this. To this end, we explore and adapt structured attention to the task of single document extractive summarization. Unlike previous models which usually consider the task as a sequential labelling problem, we propose to view it as a multi-root dependency tree induction problem. The proposed model, SUMO (**S**tructured S**um**marization M**o**del), uses the structured attention mechanism as both the objective and attention weights. For summarization, deeper and more complex tree structures need to considered. Therefore, we design a new iterative structure refinement algorithm that can repeatedly refine the trees predicted by previous iterations.

Experiments on two datasets demonstrate that our model outperforms competitive summarization systems, including a Transformer baseline and a reinforcement learning based system. We found that the structured attention mechanism is also helpful to extractive summarization, and the refinement process can further boost its performance.

4.1 Introduction

As described in Chapter 1, single-document summarization is the task of automatically generating a shorter version of a document while retaining its most important information. Extractive summarization approaches form summaries by copying and concatenating the most important spans (usually sentences) in a document. Recent approaches to (single-document) extractive summarization frame the task as a sequence labeling problem taking advantage of the success of neural network architectures (Bahdanau et al., 2015). The idea is to predict a label for each sentence specifying whether it should be included in the summary. Existing systems mostly rely on recurrent neural networks (Hochreiter and Schmidhuber, 1997) to model the document and obtain a vector representation for each sentence (Nallapati et al., 2017; Cheng and Lapata, 2016). Inter-sentential relations are captured in a sequential manner, without taking the structure of the document into account, although the latter has been shown to correlate with what readers perceive as important in a text (Marcu, 1999). Another problem in neural-based extractive models is the lack of interpretability. While capable of identifying summary sentences, these models are not able to rationalize their predictions (e.g., a sentence is in the summary because it describes important content upon which other related sentences elaborate).

The summarization literature offers examples of models which exploit the structure of the underlying document, inspired by existing theories of discourse such as Rhetorical Structure Theory (RST; Mann and Thompson 1988). Most approaches produce summaries based on tree-like document representations obtained by a parser trained on discourse annotated corpora (Carlson et al., 2001; Prasad et al., 2008). For instance, Marcu (1999) argues that a good summary can be generated by traversing the RST discourse tree structure top-down, following nucleus nodes (discourse units in RST are characterized regarding their text importance; nuclei denote central units, whereas satellites denote peripheral ones). Other work (Hirao et al., 2013a; Yoshida et al., 2014) extends this idea by transforming RST trees into dependency trees and generating summaries by tree trimming. Gerani et al. (2014) summarize product reviews; their system aggregates RST trees representing individual reviews into a graph, from which

1. One wily coyote traveled a bit too far from home, and its resulting adventure through Harlem had alarmed residents doing a double take and scampering to get out of its way Wednesday morning.
2. Police say frightened New Yorkers reported the coyote sighting around 9:30 a.m., and an emergency service unit was dispatched to find the animal.
3. The little troublemaker was caught and tranquilized in Trinity Cemetery on 155th street and Broadway, and then taken to the Wildlife Conservation Society at the Bronx Zoo, authorities said.
4. "The coyote is under evaluation and observation," said Mary Dixon, spokesperson for the Wildlife Conservation Society.
5. She said the Department of Environmental Conservation will either send the animal to a rescue center or put it back in the wild.
6. According to Adrian Benepe, New York City Parks Commissioner, coyotes in Manhattan are rare, but not unheard of.
7. "This is actually the third coyote that has been seen in the last 10 years," Benepe said.
8. Benepe said there is a theory the coyotes make their way to the city from suburban Westchester.
9. He said they probably walk down the Amtrak rail corridor along the Hudson River or swim down the Hudson River until they get to the city.

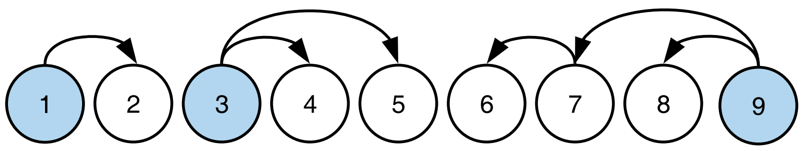


Figure 4.1: Dependency discourse tree for a document from the CNN/DailyMail dataset (Hermann et al., 2015). Blue nodes indicate the roots of the tree (i.e., summary sentences) and parent-child links indicate dependency relations.

an abstractive summary is generated. Xu et al. (2019b) propose a summarization system by applying Graph Convolutional Networks over the RST graph and co-reference graph of input documents.

Despite the intuitive appeal of discourse structure for the summarization task, the reliance on a parser which is both expensive to obtain (since it must be trained on labeled data) and error prone[1], presents a major obstacle to its widespread use.

In the previous chapter, we introduced the structured attention mechanism which can be used to learn structure-aware representations for documents. Drawing inspiration from structured attention and existing discourse-informed summarization models (Marcu, 1999; Hirao et al., 2013a), in this chapter, we propose to frame extractive summarization as a tree induction problem. Our model represents documents as multi-root dependency trees where each root node is a summary sentence, and the subtrees

[1]The macro-F1 score (on span structures) of the best RST discourse parsers is around 85% (Feng and Hirst, 2012b; Ji and Eisenstein, 2014; Li et al., 2014a; Morey et al., 2017; Yu et al., 2018), and the performance beyond sentence boundaries is considered to be much worse than that (Joty et al., 2015).

attached to it are sentences whose content is related to and covered by the summary sentence. An example of a document and its corresponding tree is shown in Figure 4.1; tree nodes correspond to document sentences; blue nodes represent sentences which should be in the summary, dependent nodes relate to or are subsumed by the parent summary sentence.

The proposed model uses a multi-root variant of the structured attention as both the objective and attention weights for extractive summarization. The model is trained end-to-end, it induces document-level dependency trees while predicting the output summary, and brings more interpretability in the summarization process by helping explain how document content contributes to the model's decisions. As illustrated in Figure 3.5 in previous chapter, the induced structures of the vanilla structured attention mechanism is meaningful but relatively shallow. For the summarization task, the document tree structure is usually considered to be deeper and more complex (Marcu, 1999; Yoshida et al., 2014). Therefore, we design a new iterative structure refinement algorithm, which learns to induce document-level structures through repeatedly refining the trees predicted by previous iterations and allows the model to infer complex trees which go beyond simple parent-child relations (Liu and Lapata, 2018; Kim et al., 2017). The idea of structure refinement is conceptually related to recently proposed models for solving iterative inference problems (Marino et al., 2018; Putzky and Welling, 2017; Lee et al., 2018). It is also related to structured prediction energy networks (Belanger et al., 2017) which approach structured prediction as iterative minimization of an energy function.

Our contributions in this chapter are three-fold: a novel conceptualization of extractive summarization as a tree induction problem; a model which capitalizes on the notion of structured attention to learn document representations based on iterative structure refinement; and large-scale evaluation studies (both automatic and human-based) which demonstrate that our approach performs competitively against state-of-the-art methods while being able to rationalize model predictions.

4.2 Model Description

Given a document containing several sentences $[s_1, s_2, \cdots, s_n]$, where s_i is the i-th sentence in the document, extractive summarization can be defined as the task of assigning a label $y_i \in \{0, 1\}$ to each s_i, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

4.2.1 Baseline Model

Most extractive models frame summarization as a classification problem. Recent approaches (Cheng and Lapata, 2016; Nallapati et al., 2017; Zhang et al., 2018c; Dong et al., 2018) incorporate a neural network-based encoder to build representations for sentences and apply a binary classifier over these representations to predict whether the sentences should be included in the summary. Given predicted scores **r** and gold labels **y**, the loss function can be defined as:

$$L = -\sum_{i=1}^{n} (\mathbf{y}_i \ln(\mathbf{r}_i) + (1-\mathbf{y}_i)\ln(1-\mathbf{r}_i)) \tag{4.1}$$

For our extractive summarization task, we design a baseline system which is composed of a sentence-level Transformer ($\mathcal{T}_\mathrm{S}$) and a document-level Transformer ($\mathcal{T}_\mathrm{D}$), which have the same structure (for more details about the Transformer models, please see Section 2.1.2). For each sentence $s_i = [\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{im}]$ in the input document, $\mathcal{T}_\mathrm{S}$ is applied to obtain a contextual representation for each word:

$$[\mathbf{u}_{i1}, \mathbf{u}_{i2}, \cdots, \mathbf{u}_{im}] = \mathcal{T}_\mathrm{S}([\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{im}]) \tag{4.2}$$

And the representation of a sentence is acquired by applying weighted-pooling:

$$\mathbf{a}_{ij} = \mathbf{W}_0 \mathbf{u}_{ij}^T \tag{4.3}$$

$$\mathbf{s}_i = \frac{1}{m} \sum_{j=1}^{m} \mathbf{a}_{ij} \mathbf{u}_{ij} \tag{4.4}$$

Document-level transformer $\mathcal{T}_\mathrm{D}$ takes $\mathbf{s}_i$ as input and yields a contextual representation for each sentence:

$$[\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n] = \mathcal{T}_\mathrm{D}([\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n]) \tag{4.5}$$

Following previous work (Nallapati et al., 2017), we use a sigmoid function after a linear transformation to calculate the probability $\mathbf{r}_i$ of selecting s_i as a summary sentence:

$$\mathbf{r}_i = \sigma(\mathbf{W}_1 \mathbf{v}_i^T) \tag{4.6}$$

4.2.2 Structured Summarization Model

In the Transformer model sketched above, inter-sentence relations are modeled by multi-head attention based on softmax functions, which only capture shallow structural information. Our summarizer, which we call SUMO as a shorthand for **S**tructured S**um**marization **Mo**del classifies sentences as summary-worthy or not, and simultaneously induces the structure of the source document as a multi-root tree. An overview of SUMO is illustrated in Figure 4.2. The model has the same sentence-level encoder $\mathcal{T}_S$ as the baseline Transformer model, but the document-level encoder differs from the baseline system in two important ways: (a) it uses structured attention to model the roots (i.e., summary sentences) of the underlying tree; and (b) through iterative refinement it is able to progressively infer more complex structures from past guesses (see the second and third block in Figure 4.2).

Structured Attention Assuming document sentences have been already encoded, SUMO first calculates the unnormalized root score $\tilde{\mathbf{r}}_i$ for s_i to indicate the extent to which it might be selected as root in the document tree (red squares in Figure 4.2). It also calculates the unnormalized edge score $\tilde{\mathbf{e}}_{ij}$ for sentence pair $\langle s_i, s_j \rangle$ indicating the extent to which s_i might be the head of s_j in that tree (green squares in Figure 4.2). To inject structural bias, SUMO normalizes these scores as the marginal probabilities of forming edges in the document dependency tree. The leaves of this dependency tree are document sentences, while the edges connecting them indicate the dependency relationships between them and determine how information flows in the attention mechanism.

We use the Tree-Matrix-Theorem (TMT; Kirchhoff 1847; Koo et al. 2007; Tutte 1984) to calculate root marginal probability $\mathbf{r}_i$ and edge marginal probability $\mathbf{e}_{ij}$. Different from the method described in Chapter 3, here we use a multi-root variant of the Tree-Matrix-Theorem, where the induced structure is a multi-root dependency tree, since in our task the summary typically contains multiple sentences. As illustrated in Algorithm 1, we first build the Laplacian matrix $\bar{\mathbf{L}}$ based on unnormalized scores and calculate marginal probabilities by matrix inverse-based operations ($\bar{\mathbf{L}}^{-1}$). Given sentence vector $\mathbf{s}_i$ as input, SUMO computes:

$$\tilde{\mathbf{r}}_i = \mathbf{W}_r \mathbf{s}_i \tag{4.7}$$

$$\tilde{\mathbf{e}}_{ij} = \mathbf{s}_i \mathbf{W}_e \mathbf{s}_j^T \tag{4.8}$$

$$\mathbf{r}_i, \mathbf{e}_{ij} = \mathrm{TMT}(\tilde{\mathbf{r}}_i, \tilde{\mathbf{e}}_{ij}) \tag{4.9}$$

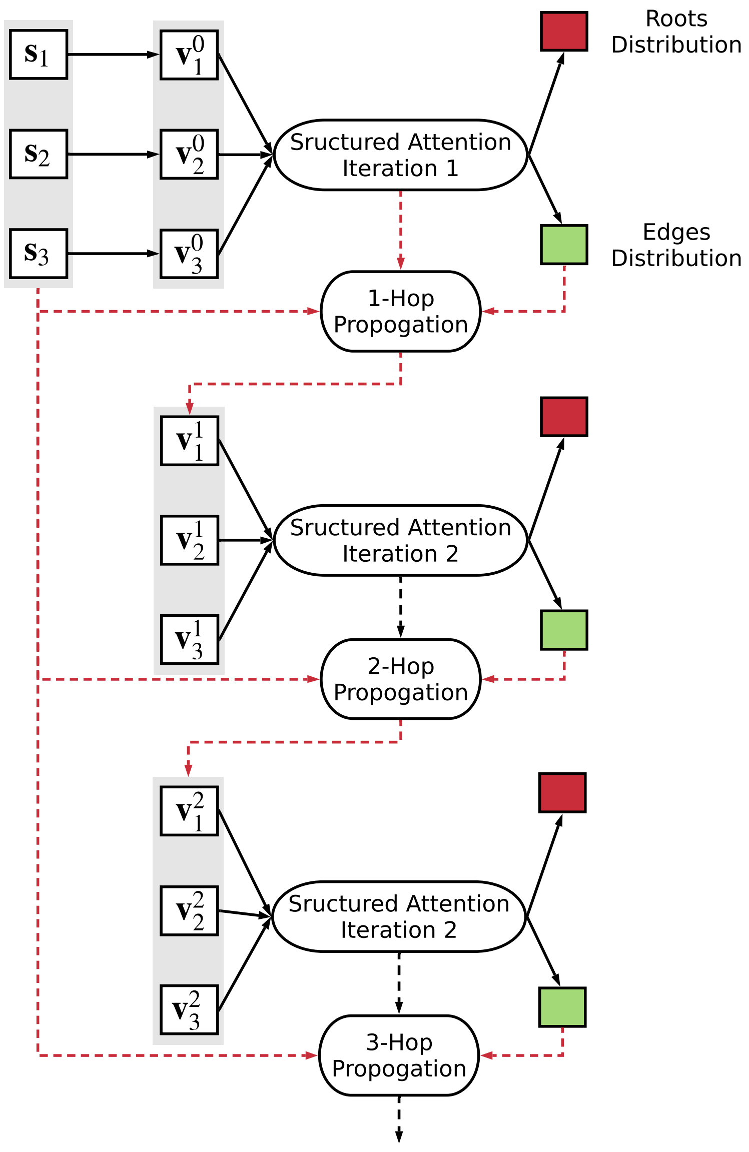


Figure 4.2: Overview of SUMO. A Transformer-based sentence-level encoder builds a vector $\mathbf{s}_i$ for each sentence s_i. The dotted lines indicate iterative application of structured attention, where at each iteration the model outputs a roots distribution and an edges distribution. Roots distribution is used to is calculate the extractive loss based on gold summary sentences. Edges distribution is used as the attention weights to build new sentence embeddings for next iteration. $\mathbf{v}_i^k$ indicates the sentence embedding for s_i after iteration k.

Algorithm 1: Calculate Tree Marginal Probabilities based on Tree-Matrix-Theorem

```
Input: unnormalized root score $\tilde{\mathbf{r}}_i$, unnormalized edge score $\tilde{\mathbf{e}}_{ij}$
Function TMT ($\tilde{\mathbf{r}}_i, \tilde{\mathbf{e}}_{ij}$):
    $\mathbf{A}_{ij} = \begin{cases} 0 & \text{if } i = j \\ \exp(\tilde{\mathbf{r}}_{ij}) & \text{otherwise} \end{cases}$
    $\mathbf{L}_{ij} = \begin{cases} \sum_{i'=1}^{n} \mathbf{A}_{i'j} & \text{if } i = j \\ -\mathbf{A}_{ij} & \text{otherwise} \end{cases}$
    $\bar{\mathbf{L}}_{ij} = \begin{cases} \mathbf{L}_{ij} + \exp(\tilde{\mathbf{r}}_i) & i = j \\ \mathbf{L}_{ij} & \text{otherwise} \end{cases}$
    $\mathbf{e}_{ij} = \mathbf{A}_{ij}[\bar{\mathbf{L}}^{-1}]_{jj} - \mathbf{A}_{ij}[\bar{\mathbf{L}}^{-1}]_{ji}$
    $\mathbf{r}_i = \exp(\tilde{\mathbf{r}}_i)[\bar{\mathbf{L}}^{-1}]_{ii}$
    return $\mathbf{r}_i, \mathbf{e}_{ij}$
```

Iterative Structure Refinement SUMO essentially reduces summarization to a rooted-tree parsing problem. However, accurately predicting a tree in one shot is problematic. Firstly, when predicting the dependency tree, the model has solely access to labels for the roots (aka summary sentences), while tree edges are latent and learned without an explicit training signal. And shown in section 3.3.1, a single application of TMT leads to shallow tree structures. Secondly, the calculation of $\tilde{\mathbf{r}}_i$ and $\tilde{\mathbf{e}}_{ij}$ would be based on first-order features alone, however, higher-order information pertaining to siblings and grandchildren has proved useful in discourse parsing (Carreras, 2007).

We address these issues with an inference algorithm which iteratively infers latent trees. In contrast to multi-layer neural network architectures like the Transformer or Recursive Neural Networks (Tai et al., 2015) where word representations are updated at every layer based on the output of previous layers, we refine only the tree structure during each iteration, word representations are not passed across multiple layers. Empirically, at early iterations, the model learns shallow and simple trees, and information propagates mostly between neighboring nodes; as the structure gets more refined, information propagates more globally allowing the model to learn higher-order features.

Algorithm 2 provides the details of our refinement procedure. SUMO takes K iterations to learn the structure of a document. For each sentence, we initialize a structural vector $\mathbf{v}_i^0$ with sentence vector $\mathbf{s}_i$. At iteration k, we use sentence embeddings from

Algorithm 2: Structured Summarization Model

```
   Input: Document $d$
   Output: Root probabilities $\mathbf{r}^K$ after $K$ iterations
1  Calculate sentence vectors $\mathbf{s}$ using sentence-level Transformer $\mathcal{T}_S$
2  $\mathbf{v}^0 \leftarrow \mathbf{s}$
3  for $k \leftarrow 1$ to $K-1$ do
4      Calculate unnormalized root scores: $\tilde{\mathbf{r}}_i^k = \mathbf{W}_r^k \mathbf{v}_i^{k-1}$
5      Calculate unnormalized edge scores: $\tilde{\mathbf{e}}_{ij}^k = \mathbf{v}_i^{k-1} \mathbf{W}_e^k \mathbf{v}_j^{k-1^T}$
6      Calculate marginal root and edge probabilities: $\mathbf{r}^k, \mathbf{e}^k = \mathrm{TMT}(\tilde{\mathbf{r}}^k, \tilde{\mathbf{e}}^k)$
7      Update sentence representations: $\mathbf{v}^k = \text{k-Hop-Propogation}(\mathbf{e}^k, \mathbf{s}, k)$
8  end
9  Calculate final unnormalized root and edge scores: $\tilde{\mathbf{r}}_i^K = \mathbf{W}_r^K \mathbf{v}_i^{K-1}$,
   $\tilde{\mathbf{e}}_{ij}^K = \mathbf{v}_i^{K-1} \mathbf{W}_e^K \mathbf{v}_j^{K-1^T}$
10 Calculate final root and edge probabilities: $\mathbf{r}^K, \mathbf{e}^K = \mathrm{TMT}(\tilde{\mathbf{r}}^K, \tilde{\mathbf{e}}^K)$
11
12 Function k-Hop-Propogation($\mathbf{e}$, $\mathbf{s}$, $k$):
13     $\mathbf{z}^0 \leftarrow \mathbf{s}$
14     for $l \leftarrow 1$ to $k$ do
15         $\mathbf{p}_i^l = \frac{1}{n} \sum_{j=1}^n \mathbf{e}_{ji} \mathbf{z}_i^{l-1}$
16         $\mathbf{c}_i^l = \frac{1}{n} \sum_{j=1}^n \mathbf{e}_{ij} \mathbf{z}_i^{l-1}$
17         $\mathbf{z}_i^l = \tanh(\mathbf{W}_v^k [\mathbf{p}_i^{l-1}, \mathbf{c}_i^{l-1}, \mathbf{z}_i^{l-1}])$
18     end
19     return $\mathbf{v}^k$
```

the previous iteration $\mathbf{v}^{k-1}$ to calculate unnormalized root $\tilde{\mathbf{r}}_i^k$ and edge $\tilde{\mathbf{e}}_{ij}^k$ scores using a linear transformation with weight $\mathbf{W}_r^k$ and a bilinear transformation with weight $\mathbf{W}_e^k$, respectively. Marginal root and edge probabilities are subsequently normalized with the TMT to obtain $\mathbf{r}_i^k$ and $\mathbf{e}_{ij}^k$ (see lines 4–6 in Algorithm 2). Then, sentence embeddings are updated with *k-Hop Propagation*. The latter takes as input the initial sentence representations $\mathbf{s}$ rather than sentence embeddings $\mathbf{v}^{k-1}$ from the previous layer. In other words, new embeddings $\mathbf{v}^k$ are computed from scratch relying on the structure from the previous layer. Within the *k-Hop-Propagation* function (lines 12–19), edge probabilities $\mathbf{e}_{ij}^k$ are used as attention weights to propagate information from a sentence to all other sentences in k hops. $\mathbf{p}_i^l$ and $\mathbf{c}_i^l$ represent parent and child vectors, respectively, while vector $\mathbf{z}_i^l$ is updated with contextual information at hop l. At the final iteration (lines 9 and 10), the top sentence embeddings $\mathbf{v}^{K-1}$ are used to calculate the final root probabilities r^K.

We define the model's loss function as the summation of the losses of all iterations:

$$L = \sum_{k=1}^{K} [y \log(r_k) + (1-y)\log(1-r_k)] \tag{4.10}$$

SUMO uses the root probabilities of the top layer as the scores for summary sentences.

The *k-Hop-Propagation* function resembles the Graph Convolution Networks (Kipf and Welling, 2017a; Marcheggiani and Titov, 2017), which encode a graph representation with convolutional operations. GCNs have been been recently applied to latent trees (Corro and Titov, 2019), however not in combination with iterative refinement.

4.3 Experiments

In this section we present our experimental setup, describe the summarization datasets we used, discuss implementation details, our evaluation protocol, and analyze our results.

4.3.1 Summarization Datasets

We evaluated SUMO on two benchmark datasets, namely the CNN/DailyMail news highlights dataset (Hermann et al., 2015) and the New York Times Annotated Corpus (NYT; Sandhaus 2008). The CNN/DailyMail dataset contains news articles and associated highlights, i.e., a few bullet points giving a brief overview of the article. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents). We did not anonymize entities.

The NYT dataset contains 110,540 articles with abstractive summaries. Following Durrett et al. (2016), we split these into 100,834 training and 9,706 test examples, based on date of publication (test is all articles published on January 1, 2007 or later). We also followed their filtering procedure, documents with summaries that are shorter than 50 words were removed from the raw dataset. The filtered test set includes 3,452 test examples out of the original 9,706. Compared to CNN/DailyMail, the NYT dataset contains longer and more elaborate summary sentences.

Both datasets contain abstractive gold summaries, which are not readily suited to training extractive summarization models. A greedy algorithm similar to Nallapati et al. (2017) was used to generate an oracle summary for each document. Starting from an empty summary set, we select one sentence from the source document into the summary set incrementally at a time. At each selection, we maximize the ROUGE-1 and ROUGE-2 score of the current set with respect to the reference summary. The process is stopped when none of the remaining source sentences improves the score. The selected summary sentences are called oracle summary. We assigned label 1 to sentences in the oracle summary and 0 otherwise and trained SUMO on this data.

4.3.2 Implementation Details

We followed the same training procedure for SUMO and various Transformer-based baselines. The vocabulary size was set to 30K. We used 300D word embeddings which were initialized randomly from $\mathcal{N}(0, 0.01)$. The sentence-level Transformer has 6 layers and the hidden size of the feed-forward network was set to 512. The number of heads in multi-head attention was set to 4. Adam was used for training ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We adopted the learning rate schedule from Vaswani et al. (2017) with warming-up on the first 8,000 steps. SUMO and related Transformer models produced 3-sentence summaries for each document at test time (for both CNN/DailyMail and NYT datasets).

4.3.3 Automatic Evaluation

We evaluated summarization quality using ROUGE F_1 (Lin, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

Table 4.1 summarizes our results. We evaluated two variants of SUMO, with one and three structured-attention layers. We compared against a baseline which simply selects the first three sentences in each document (LEAD-3) and several incarnations of the basic Transformer model introduced in Section 4.2.1. These include a Transformer without document-level self-attention and two variants with document-level self attention instantiated with one and three layers. Several state-of-the-art models are also included in Table 4.1, both extractive and abstractive.

REFRESH (Narayan et al., 2018b) is an extractive summarization system trained by globally optimizing the ROUGE metric with reinforcement learning. The system of Marcu (1999) is another extractive summarizer based on RST parsing. It uses discourse structures and RST's notion of nuclearity to score document sentences in terms of their importance and selects the most important ones as the summary. Our re-implementation of Marcu (1999) used the parser of Zhao and Huang (2017) to obtain RST trees. Durrett et al. (2016) develop a summarization system which integrates a compression model that enforces grammaticality and coherence. See et al. (2017) present an abstractive summarization system based on an encoder-decoder architecture. Celikyilmaz et al.'s (Celikyilmaz et al., 2018) system is an abstractive summarization system using multiple agents to represent the document as well a hierarchical attention mechanism over the agents for decoding.

As far as SUMO is concerned, we observe that it outperforms a simple Transformer model without any document attention as well as variants with document attention. SUMO with three layers of structured attention overall performs best, confirming our hypothesis that document-level structure is beneficial for summarization. The results in Table 4.1 also reveal that SUMO and all Transformer-based models with document attention (doc-att) outperform LEAD-3 across metrics. SUMO (3-layer) is competitive or better than state-of-the-art approaches. Examples of system output are shown in Table 4.4. Finally, we should point out that SUMO is superior to Marcu (1999) even though the latter employs linguistically informed document representations.

4.3.4 Human Evaluation

In addition to automatic evaluation, we also assessed system performance by eliciting human judgments. Our first evaluation quantified the degree to which summarization models retain key information from the document following a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018b). We created a set of questions based on the gold summary under the assumption that it highlights the most important document content. We then examined whether participants were able to answer these questions by reading system summaries alone without access to the article. The more questions a system can answer, the better it is at summarizing the document as a whole.

We randomly selected 20 documents from the CNN/DailyMail and NYT datasets, respectively and wrote multiple question-answer pairs for each gold summary. We created 71 questions in total varying from two to six questions per gold summary. We asked participants to read the summary and answer all associated questions as best they could without access to the original document or the gold summary. Examples of

	CNN			DM			CNN+DM			NYT		
Model	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
LEAD-3	29.2	11.2	26.0	40.7	18.3	37.2	39.6	17.7	36.2	35.5	17.3	32.0
Narayan et al. (2018b)	**30.4**	11.7	**26.9**	41.0	18.8	37.7	40.0	18.2	36.6	41.3	22.0	37.8
Marcu (1999)	25.6	6.10	19.5	31.9	12.4	23.5	26.5	9.80	20.4	29.6	11.2	23.0
Durrett et al. (2016)	—	—	—	—	—	—	—	—	—	40.8	22.3	36.7
See et al. (2017)	—	—	—	—	—	—	39.5	17.3	36.4	**42.7**	22.1	38.0
Celikyilmaz et al. (2018)	—	—	—	—	—	—	**41.7**	19.5	**37.9**	—	—	—
Transformer (no doc-att)	29.2	11.1	25.6	40.5	18.1	36.8	39.7	17.0	35.9	41.1	21.5	37.0
Transformer (1-layer)	29.5	11.4	26.0	41.5	18.7	38.0	40.6	18.1	36.7	41.8	22.1	37.8
Transformer (3-layer)	29.6	11.8	26.3	41.7	18.8	38.0	40.6	18.1	36.9	42.0	22.3	38.2
SUMO (1-layer)	29.5	11.6	26.2	41.6	18.8	37.6	40.5	18.0	36.8	42.2	22.1	38.1
SUMO (3-layer)	29.7	**12.0**	26.5	**42.0**	**19.1**	**38.0**	41.0	18.4	37.2	42.3	**22.7**	**38.6**

Table 4.1: Test set results on the CNN/DailyMail and NYT datasets using ROUGE F_1 (R-1 and R-2 are shorthands for unigram and bigram overlap, R-L is the longest common subsequence).

questions and their answers are given in Table 4.4. We adopted the same scoring mechanism used in Clarke and Lapata (2010), i.e., a correct answer was marked with a score of one, partially correct answers with a score of 0.5, and zero otherwise. Answers were elicited using Amazon's Mechanical Turk platform. Participants evaluated summaries produced by the LEAD-3 baseline, our 3-layered SUMO model and multiple state-of-the-art systems. We elicited 5 responses per summary. Detailed instructions of human evaluation can be found in Appendix A.

Table 4.2 (QA column) presents the results of the QA-based evaluation. Based on the summaries generated by SUMO, participants can answer 65.3% of questions correctly on CNN/DailyMail and 57.2% on NYT. Summaries produced by LEAD-3 and comparison systems fare worse, with REFRESH (Narayan et al., 2018b) coming close to SUMO on CNN/DailyMail but not on NYT. Overall, we observe there is room for improvement since no system comes close to the extractive oracle, indicating that improved sentence selection would bring further performance gains to extractive approaches. Between-systems differences are all statistically significant (using a one-way ANOVA with posthoc Tukey HSD tests; $p < 0.01$) with the exception of LEAD-3 and See et al. (2017) in both CNN+DM and NTY, Narayan et al. (2018b) and SUMO in

	CNN+DM		NYT	
Model	Rank	QA	Rank	QA
LEAD	0.07	40.1	-0.18	36.3
Narayan et al. (2018b)	**0.21**	62.4	0.12	46.1
Durrett et al. (2016)	—	—	-0.11	40.1
See et al. (2017)	-0.23	36.6	-0.44	35.3
Celikyilmaz et al. (2018)	-0.64	37.5	—	—
SUMO (3-layer)	0.15	**65.3**	**0.33**	**57.2**
GOLD	0.11	—	-0.16	—
ORACLE	0.37	74.6	0.41	67.1

Table 4.2: System ranking according to human judgments on summary quality and QA-based evaluation.

both CNN+DM and NTY, and LEAD-3 and Durrett et al. (2016) in NYT.

Our second evaluation study assessed the overall quality of the summaries by asking participants to rank them taking into account the following criteria: *Informativeness* , *Fluency*, and *Succinctness*. The study was conducted on the Amazon Mechanical Turk platform using Best-Worst Scaling (Louviere et al., 2015), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2017). Participants were presented with a document and summaries generated from 3 out of 7 systems and were asked to decide which summary was better and which one was worse, taking into account the criteria mentioned above. We used the same 20 documents from each dataset as in our QA evaluation and elicited 5 responses per comparison.

The rating of each system was computed as the percentage of times it was chosen as best minus the times it was selected as worst. Ratings range from -1 (worst) to 1 (best). As shown in Table 4.2 (Rank column), participants overwhelming prefer the extractive oracle summaries followed by SUMO and REFRESH (Narayan et al., 2018b). Abstractive systems (Celikyilmaz et al., 2018; See et al., 2017; Durrett et al., 2016) perform relatively poorly in this evaluation; we suspect that humans are less forgiving to fluency errors and slightly incoherent summaries. Interestingly, gold summaries fare worse than the oracle and extractive systems. Albeit fluent, gold summaries naturally contain less detail compared to oracle-based ones; on virtue of being abstracts, they are written in a telegraphic style, often in conversational language while participants prefer the more lucid style of the extracts. All pairwise comparisons among systems are statistically significant (using a one-way ANOVA with post-hoc Tukey HSD tests; $p < 0.01$) except LEAD-3 and See et al. (2017) in both CNN+DM and NTY, Narayan et al. (2018b) and SUMO in both CNN+DM and NTY, and LEAD and Durrett et al. (2016) in NYT.

4.3.5 Evaluation of the Induced Structures

To gain further insight into the structures learned by SUMO, we inspected the trees it produces. As in the previous chapter, we used the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) to extract the maximum spanning tree from the attention scores. We report various statistics on the characteristics of the induced trees across datasets in Table 4.3. We also examine the trees learned from different SUMO variants (with different numbers of iterations) in order to establish whether the iterative process yields better structures.

Specifically, we compared the dependency trees obtained from our model to those produced by a discourse parser (Zhao and Huang, 2017) trained on a corpus which combines annotations from the RST treebank (Carlson et al., 2001) and the Penn Treebank (Marcus et al., 1993). Unlike traditional RST discourse parsers (Feng and Hirst, 2014), which first segment a document into Elementary Discourse Units (EDUs) and then build a discourse tree with the EDUs[2] as leaves, Zhao and Huang (2017) parse a document into an RST tree along with its syntax subtrees without segmenting it into EDUs. The outputs of their parser are ideally suited for comparison with our model, since we only care about document-level structures, and ignore the subtrees within sentence boundaries. We converted the constituency RST trees obtained from the discourse parser into dependency trees using Hayashi et al. (2016)'s algorithm by assigning each sentence a unique head in a bottom-up manner.

As can be seen in Table 4.3, the dependency structures induced by SUMO are simpler compared to those obtained from the discourse parser. Our trees are generally shallower, almost half of them are projective. This is because there are two main patterns in the induced trees, where the first pattern is a chain structure composed by consecutive nodes, and the second pattern is a tree node with a large number of children. These two patterns lead more induced trees to be projective.

We also calculated the percentage of head-dependency edges that are identical be-

[2]EDUs roughly correspond to clauses.

	CNN+DM			NYT		
	P(%)	H	EA(%)	P(%)	H	EA(%)
Parser	24.8	8.9	—	18.7	10.6	—
SUMO (1-layer)	69.0	2.9	23.1	54.7	3.6	20.6
SUMO (3-layer)	52.7	3.7	25.3	45.1	6.2	21.6
Left Branching	—	—	21.4	—	—	21.3
Right Branching	—	—		—	—	6.7

Table 4.3: Descriptive statistics Projectivity(P), Height(H) and EdgeAgreement(EA) for dependency trees produced by our model and the RST discourse parser of Zhao and Huang (2017). Results are shown on the CNN/DailyMail and NYT test sets.

tween learned trees and parser generated ones. Although SUMO is not exposed to any annotated trees during training, a number of edges agree with the outputs of the discourse parser. Moreover, we observe that the iterative process involving multiple structured attention layers helps generate better discourse trees, since the agreement with parser generated trees is higher. We also compare SUMO trees against a left- and right-branching baseline, where the document is trivially parsed into a left- and right-branching tree forming a chain-like structure. As shown in Table 4.3, SUMO outperforms these baselines (with the exception of the one-layered model on NYT).

4.4 Examples of System Output and Summary Evaluation Questions

Table 4.4 shows examples of system output. Specifically, we show summaries produced from our model, SUMO, REFRESH (Narayan et al., 2018b), a pointer-generator model (See et al., 2017), a baseline which simply selects the first three sentences in each document (LEAD-3), and gold standard summaries. The table also contains examples of questions (and their answers) used in our QA-based evaluation study. We can see compared with baseline systems, SUMO produces summaries that are more informative and can answer more questions in the question answering evaluation. For example, in the CNN/DM example in Table 4.4, SUMO is the only system that mentioned the main event in the news and name of the operating company, while other systems only summarized the background information.

4.5 Summary

In this chapter we provide a new perspective on extractive summarization, conceptualizing it as a tree induction problem. We present SUMO, a Structured Summarization Model, which induces a multi-root dependency tree of a document, where roots are summary-worthy sentences, and subtrees attached to them are sentences which elaborate or explain the summary content. SUMO generates complex trees following an iterative refinement process which builds latent structures while using information learned in previous iterations. Experiments on two datasets, show that SUMO performs competitively against state-of-the-art methods and induces meaningful tree structures.

Incorporating document structures into neural summarization systems can provide deeper understanding of the documents. In the next chapter, we will show an alternative encoder that does not emphasize structural information but is augmented by model pretraining can also boost the performance of a summarization system.

	CNN/DM	NYT
GOLD	A company called CyArk specializes in digital preservation of threatened ancient and historical architecture. Founded by an Iraqi-born engineer, it plans to preserve 500 World Heritage sites within five years.	Louisiana officials set July 31 deadline for applicants for the Road Home, grant program for homeowners who lost their houses to hurricanes Katrina and Rita. Program is expected to cost far more than $7.5 billion provided by Federal Government, in part because many more families have applied than officials anticipated. With cutoff date, State hopes to figure out how much more money it needs to pay for program. Shortfall is projected to be $2.9 billion.
Question-Answer Pairs	Which company specializes in digital preservation of threatened ancient and historical architecture? [CyArk] How many World Heritage sites does the company plan to preserve? [500]	What is Road Home? [the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita] When is the applicants' deadline for the Road Home? [July 31] Why is the program expected to cost far more than $7.5 billion? [many more families have applied than officials anticipated] What is the shortfall projected to be? [$2.9 billion]
LEAD-3	In 2001, the Taliban wiped out 1700 years of history in a matter of seconds, by blowing up ancient Buddha statues in central Afghanistan with dynamite. They proceeded to do so after an attempt at bringing down the 175-foot tall sculptures with anti-aircraft artillery had failed. Sadly, the event was just the first in a series of atrocities that have robbed the world of some of its most prized cultural heritage.	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the Federal Government, in part because many more families have applied than officials had anticipated. As a result, Louisiana officials on Tuesday night set a July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses. With the cutoff date, the State hopes to be able to figure out how much more money it needs to pay for the program.

Table 4.4: Examples of GOLD human authored summaries, questions based on them (answers shown in square brackets) and automatic summaries produced by the LEAD-3 baseline, the abstractive system of See et al. (2017), REFRESH (Narayan et al., 2018b), and SUMO for a CNN and NYT (test) article.

	CNN/DM	NYT
See et al. (2017)	The Taliban wiped out 1700 years of history in a matter of seconds. The thought of losing a piece of our collective history is a bleak one. But if loss can't be avoided, technology can lend a hand.	Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita is expected to cost far more than $7.5 billion provided by federal government. Louisiana officials set July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses.
Narayan et al. (2018b)	Sadly, the event was just the first in a series of atrocities that have robbed the world of some of its most prized cultural heritage. But historical architecture is also under threat from calamities which might well escape our control, such as earthquakes and climate change. The thought of losing a piece of our collective history is a bleak one.	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the federal government, in part because many more families have applied than officials had anticipated. With the cutoff date, the State hopes to be able to figure out how much more money it needs to pay for the program. The shortfall is projected to be $2.9 billion.
SUMO	In 2001, the Taliban wiped out 1700 years of history in a matter of seconds, by blowing up ancient Buddha statues in central Afghanistan with dynamite. Sadly, the event was just the first in a series of atrocities that have robbed the world of some of its most prized cultural heritage. Now Cyark, a non-profit company founded by an Iraqi-born engineer, is using groundbreaking laser scanning to ensure that – at the very least – incredibly accurate digital versions of the world's treasures will stay with us forever.	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the federal government, in part because many more families have applied than officials had anticipated. As a result, Louisiana officials on Tuesday night set a July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses. The shortfall is projected to be $2.9 billion.

Table 4.4 Continued

Chapter 5

Document Summarization with Pretrained Encoders

We have shown in previous chapters that imposing a structural bias is beneficial to summarization systems. Beyond structural information at the discourse-level, there are other factors that could potentially help build a better summarizer. In this chapter, we show that to achieve a better encoding of the documents, it is possible to distill general knowledge contained in a large corpus via model pretraining. Incorporating this encoder into a summarization system could bring substantial improvement to the summarization performance.

Recently, language model pretraining has advanced the state of the art in many NLP tasks. These models usually incorporate flexible but large neural architectures such as LSTMs or Transformers for encoding input texts, and are trained on vast amounts of text. Through experiments on various natural language understanding tasks (Devlin et al., 2019), pretrained language models are found to be able to obtain richer representations of sentences or documents and capture more long-tail features (Tenney et al., 2019).

However, directly applying pretrained language models to document summarization tasks faces several challenges. In this chapter, we will discuss these challenges and present how to use pretrained language models as encoders for both extractive and abstractive summarization. Our experiments show that our proposed model achieves new state-of-the-art results on multiple datasets.

5.1 Introduction

Language model pretraining has advanced the state of the art in many NLP tasks ranging from sentiment analysis, to question answering, natural language inference, named entity recognition, and textual similarity (Peters et al., 2018; Devlin et al., 2019). State-of-the-art pretrained models include ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and more recently Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019). BERT combines both word and sentence representations in a single very large Transformer (Vaswani et al., 2017); it is pretrained on vast amounts of text, with an unsupervised objective of masked language modeling and next-sentence prediction and can be fine-tuned with various task-specific objectives.

In most cases, pretrained language models have been employed as encoders for sentence- and paragraph-level natural language understanding problems (Devlin et al., 2019) involving various classification tasks (e.g., predicting whether any two sentences are in an entailment relationship; or determining the completion of a sentence among four alternative sentences). In this chapter, we examine the influence of language model pretraining on text summarization. Different from previous tasks, summarization requires wide-coverage natural language understanding going beyond the meaning of individual words and sentences. Furthermore, under abstractive modeling formulations, the task requires language generation capabilities in order to create summaries containing novel words and phrases not featured in the source text.

We explore the potential of BERT for text summarization under a general framework encompassing both extractive and abstractive modeling paradigms. We propose a novel document-level encoder based on BERT which is able to encode a document and obtain representations for its sentences. Our extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers to capture document-level features for extracting sentences. Our abstractive model adopts an encoder-decoder architecture (see Section 2.2.2 for details), combining the same pretrained BERT encoder with a randomly-initialized Transformer decoder (Vaswani et al., 2017). We design a new training schedule which separates the optimizers of the encoder and the decoder in order to accommodate the fact that the former is pretrained while the latter must be trained from scratch. Finally, motivated by previous work showing that the combination of extractive and abstractive objectives can help generate better summaries (Gehrmann et al., 2018), we present a two-stage approach where the encoder is fine-tuned twice, first with an extractive objective and subsequently on the abstractive summarization task.

We evaluate the proposed approach on three single-document news summarization datasets representative of different writing conventions (e.g., important information is concentrated at the beginning of the document or distributed more evenly throughout) and summary styles (e.g., verbose vs. more telegraphic; extractive vs. abstractive). Across datasets, we experimentally show that the proposed models achieve state-of-the-art results under both extractive and abstractive settings. Our contributions in this chapter are three-fold: a) we highlight the importance of document encoding for the summarization task; a variety of recently proposed techniques aim to enhance summarization performance via copying mechanisms (Gu et al., 2016; See et al., 2017; Nallapati et al., 2017), reinforcement learning (Narayan et al., 2018b; Paulus et al., 2018; Dong et al., 2018), and multiple communicating encoders (Celikyilmaz et al., 2018). We achieve better results with a minimum-requirements model without using any of these mechanisms; b) we showcase ways to effectively employ pretrained language models in summarization under both extractive and abstractive settings; we would expect any improvements in model pretraining to translate in better summarization in the future; and c) the proposed models can be used as a stepping stone to further improve summarization performance as well as baselines against which new proposals are tested.

5.2 Pretrained Language Models

Pretrained language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Zhang et al., 2019a; Dong et al., 2019; Zhang et al., 2019b) have recently emerged as a key technology for achieving impressive gains in a wide variety of natural language tasks. Pretrained language models are typically used to enhance performance in language understanding tasks. Very recently, there have been attempts to apply pretrained models to various generation problems (Edunov et al., 2019; Rothe et al., 2019). These models extend the idea of word embeddings by learning contextual representations from large-scale corpora using a language modeling objective. Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) is a new language representation model which is trained with a masked language modeling and a "next sentence prediction" task on a corpus of 3,300M words.

The general architecture of BERT is shown in the upper part of Figure 5.1. Input text is first preprocessed by inserting two special tokens. [CLS] is appended to the

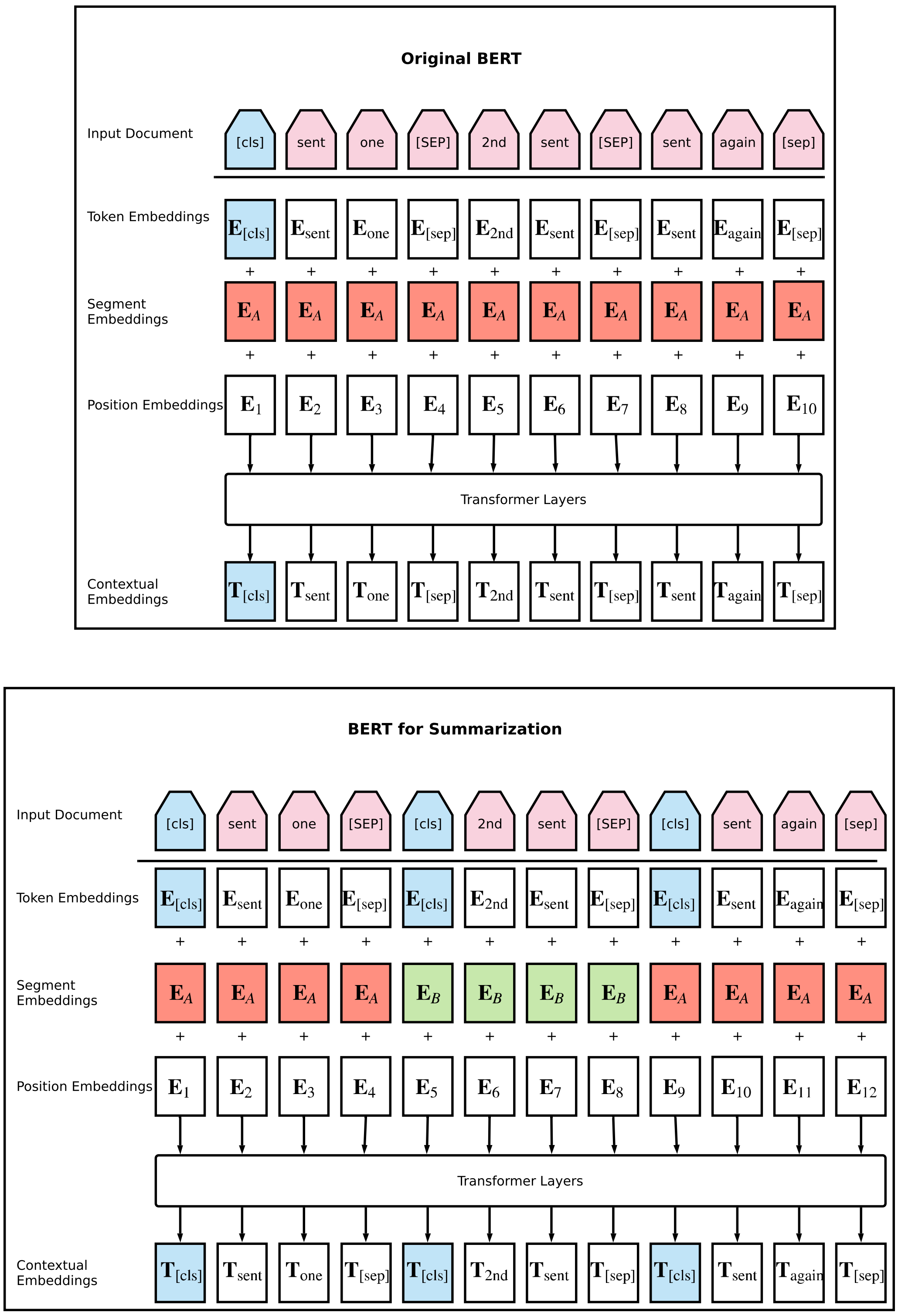

Original BERT
Input Document
[cls] sent one [SEP] 2nd sent [SEP] sent again [sep]
Token Embeddings
$E_{[cls]}$ E_{sent} E_{one} $E_{[sep]}$ E_{2nd} E_{sent} $E_{[sep]}$ E_{sent} E_{again} $E_{[sep]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_A E_A E_A E_A
Position Embeddings
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}
Transformer Layers
Contextual Embeddings
$T_{[cls]}$ T_{sent} T_{one} $T_{[sep]}$ T_{2nd} T_{sent} $T_{[sep]}$ T_{sent} T_{again} $T_{[sep]}$
BERT for Summarization
Input Document
[cls] sent one [SEP] [cls] 2nd sent [SEP] [cls] sent again [sep]
Token Embeddings
$E_{[cls]}$ E_{sent} E_{one} $E_{[sep]}$ $E_{[cls]}$ E_{2nd} E_{sent} $E_{[sep]}$ $E_{[cls]}$ E_{sent} E_{again} $E_{[sep]}$
Segment Embeddings
E_A E_A E_A E_A E_B E_B E_B E_B E_A E_A E_A E_A
Position Embeddings
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11} E_{12}
Transformer Layers
Contextual Embeddings
$T_{[cls]}$ T_{sent} T_{one} $T_{[sep]}$ $T_{[cls]}$ T_{2nd} T_{sent} $T_{[sep]}$ $T_{[cls]}$ T_{sent} T_{again} $T_{[sep]}$

Figure 5.1: Architecture of the original BERT model (above) and BERTSUM (below). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences.

beginning of the text; the output representation of this token is used to aggregate information from the whole sequence (e.g., for classification tasks). And token [SEP] is inserted after each sentence as an indicator of sentence boundaries. The modified text is then represented as a sequence of tokens $X = [x_1, x_2, \cdots, x_n]$. Each token x_i is assigned three kinds of embeddings: *token embeddings* indicate the meaning of each token, *segmentation embeddings* are used to discriminate between two sentences (e.g., during a sentence-pair classification task) and *position embeddings* indicate the position of each token within the text sequence. These three embeddings are summed to a single input vector $\mathbf{x}_i$ and fed to a bidirectional Transformer with multiple layers:

$$\tilde{h}^l = \mathrm{LN}(\mathbf{h}^{l-1} + \mathrm{MHAtt}(\mathbf{h}^{l-1})) \tag{5.1}$$

$$h^l = \mathrm{LN}(\tilde{\mathbf{h}}^l + \mathrm{FFN}(\tilde{\mathbf{h}}^l)) \tag{5.2}$$

where $\mathbf{h}^0 = \mathbf{x}$ are the input vectors; LN is the layer normalization operation (Ba et al., 2016); MHAtt is the multi-head attention operation (Vaswani et al., 2017); superscript l indicates the depth of the stacked layer. On the top layer, BERT will generate an output vector $\mathbf{t}_i$ for each token with rich contextual information. The detailed description of the Transformer model is presented in Section 2.1.2.

5.3 Fine-tuning BERT for Summarization

5.3.1 Summarization Encoder

Although BERT has been used to fine-tune various NLP tasks, its application to summarization is not as straightforward. Since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences, while in extractive summarization, as we introduced in Section 2.2, most models manipulate sentence-level representations. Although segmentation embeddings represent different sentences in BERT, they only apply to sentence-pair inputs, while in summarization we must encode and manipulate multi-sentential inputs. Figure 5.1 illustrates our proposed BERT architecture for SUMmarization (which we call BERTSUM).

In order to represent *individual* sentences, we insert external [CLS] tokens at the start of each sentence, and each [CLS] symbol collects features for the sentence preceding it. We also use *interval segment* embeddings to distinguish multiple sentences within a document. For the i-th sentence s_i we assign segment embedding $\mathbf{E}_A$ or $\mathbf{E}_B$ depending on whether i is odd or even. For example, for document $[s_1, s_2, s_3, s_4, s_5]$, where s_i is the i we would assign embeddings $[\mathbf{E}_A, \mathbf{E}_B, \mathbf{E}_A, \mathbf{E}_B, \mathbf{E}_A]$. This way, document representations are learned hierarchically where lower Transformer layers represent adjacent sentences, while higher layers, in combination with self-attention, represent multi-sentence discourse.

Position embeddings in the original BERT model have a maximum length of 512; we overcome this limitation by adding more position embeddings that are initialized randomly and fine-tuned with other parameters in the encoder.

5.3.2 Extractive Summarization

Let d denote a document containing sentences $[s_1, s_2, \cdots, s_m]$, where s_i is the i-th sentence in the document. As mentioned in Chapter 2.2.1, extractive summarization can be defined as the task of assigning a label $\mathbf{y}_i \in \{0, 1\}$ to each s_i, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

With BERTSUM, the vector of the i-th [CLS] symbol from the top layers can be used as the representation for s_i. Several inter-sentence Transformer layers are then stacked on top of BERT outputs, to capture document-level features for extracting summaries:

$$\tilde{\mathbf{h}}^l = \mathrm{LN}(\mathbf{h}^{l-1} + \mathrm{MHAtt}(\mathbf{h}^{l-1})) \tag{5.3}$$

$$\mathbf{h}^l = \mathrm{LN}(\tilde{\mathbf{h}}^l + \mathrm{FFN}(\tilde{\mathbf{h}}^l)) \tag{5.4}$$

where $\mathbf{h}^0 = \mathrm{PosEmb}(\mathbf{T})$; $\mathbf{T}$ denotes the sentence vectors output by BERTSUM, and function PosEmb adds sinusoid positional embeddings (see section 2.1.2 for details) to $\mathbf{T}$, indicating the position of each sentence.

The final output layer is a sigmoid classifier:

$$\hat{\mathbf{y}}_i = \sigma(\mathbf{W}_o \mathbf{h}_i^L + \mathbf{b}_o) \tag{5.5}$$

where $\mathbf{h}_i^L$ is the vector for s_i from the top layer (the L-th layer) of the Transformer. In experiments, we implemented Transformers with $L = 1, 2, 3$ and found that a Transformer with $L = 2$ performed best. We name this model BERTSUMEXT.

The loss of the model is the binary classification entropy of prediction $\hat{\mathbf{y}}_i$ against gold label $\mathbf{y}_i$. Inter-sentence Transformer layers are jointly fine-tuned with BERTSUM. We use the Adam optimizer with $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Our learning rate schedule follows (Vaswani et al., 2017) with warming-up (warmup $= 10,000$):

$$lr = 2\mathrm{e}^{-3} \cdot \min(\mathrm{step}^{-0.5}, \mathrm{step} \cdot \mathrm{warmup}^{-1.5})$$

Datasets	# docs (train/val/test)	avg. doc length		avg. summary length		% novel bi-grams
		words	sentences	words	sentences	in gold summary
CNN	90,266/1,220/1,093	760.50	33.98	45.70	3.59	52.90
DailyMail	196,961/12,148/10,397	653.33	29.33	54.65	3.86	52.16
NYT	96,834/4,000/3,452	800.04	35.55	45.54	2.44	54.70
XSum	204,045/11,332/11,334	431.07	19.77	23.26	1.00	83.31

Table 5.1: Comparison of summarization datasets: size of training, validation, and test sets and average document and summary length (in terms of words and sentences). The proportion of novel bi-grams that do not appear in source documents but do appear in the gold summaries quantifies corpus bias towards extractive methods.

5.3.3 Abstractive Summarization

We use a standard encoder-decoder framework, as described in Chapter 2, for abstractive summarization. The encoder is the pretrained BERTSUM and the decoder is a 6-layered Transformer initialized randomly. It is conceivable that there is a mismatch between the encoder and the decoder, since the former is pretrained while the latter must be trained from scratch. This can make fine-tuning unstable; for example, the encoder might overfit the data while the decoder underfits, or vice versa. To circumvent this, we design a new fine-tuning schedule which separates the optimizers of the encoder and the decoder.

We use two Adam optimizers with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the encoder and the decoder, respectively, each with different warmup-steps and learning rates:

$$lr_{\mathcal{E}} = \tilde{lr}_{\mathcal{E}} \cdot \min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}_{\mathcal{E}}^{-1.5}) \tag{5.6}$$

$$lr_{\mathcal{D}} = \tilde{lr}_{\mathcal{D}} \cdot \min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}_{\mathcal{D}}^{-1.5}) \tag{5.7}$$

where $\tilde{lr}_{\mathcal{E}} = 2\text{e}^{-3}, \text{warmup}_{\mathcal{E}} = 20,000$ for the encoder, and $\tilde{lr}_{\mathcal{D}} = 0.1, \text{warmup}_{\mathcal{D}} = 10,000$ for the decoder. This is based on the assumption that the pretrained encoder should be fine-tuned with a smaller learning rate and smoother decay (so that the encoder can be trained with more accurate gradients when the decoder becomes stable).

In addition, we propose a two-stage fine-tuning approach, where we first fine-tune the encoder on the extractive summarization task (Section 5.3.2) and then fine-tune it on the abstractive summarization task (Section 5.3.3). Previous work (Gehrmann et al., 2018; Li et al., 2018) suggests that using extractive objectives can boost the performance of abstractive summarization. Also notice that this two-stage approach is conceptually very simple, the model can take advantage of information shared between these two tasks, without fundamentally changing its architecture. We name the default abstractive model BERTSUMABS and the two-stage fine-tuned model BERTSUMEXTABS.

5.4 Experimental Setup

In this section, we describe the summarization datasets used in our experiments and discuss various implementation details.

5.4.1 Summarization Datasets

We evaluated our model on three benchmark datasets. Beyond the CNN/DailyMail news highlights dataset and the New York Times Annotated Corpus used in the previous chapter, we report experiments on one additional dataset: XSum (Narayan et al., 2018a). These datasets represent different summary styles ranging from highlights to very brief one sentence summaries. The summaries also vary with respect to the type of rewriting operations they exemplify (e.g., some showcase more cut and paste operations while others are genuinely abstractive). Table 5.1 presents statistics on these datasets (test set).

CNN/DailyMail contains news articles and associated highlights, i.e., a few bullet points giving a brief overview of the article. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents). We did not anonymize entities. We first split sentences with the Stanford CoreNLP toolkit (Manning et al., 2014) and pre-processed the dataset following See et al. (2017). Input documents were truncated to 512 tokens.

NYT contains 110,540 articles with abstractive summaries. Following Durrett et al. (2016), we split these into 100,834/9,706 training/test examples, based on the date of publication (the test set contains all articles published from January 1, 2007 onward). We used 4,000 examples from the training as validation set. We also followed their filtering procedure, documents with summaries less than 50 words were removed from the dataset. The filtered test set (NYT50) includes 3,452 examples. Sentences were split with the Stanford CoreNLP toolkit (Manning et al., 2014) and pre-processed following Durrett et al. (2016). Input documents were truncated to 800 tokens.

XSum contains 226,711 news articles accompanied with a one-sentence summary, answering the question “What is this article about?”. We used the splits of Narayan et al. (2018a) for training, validation, and testing (204,045/11,332/11,334) and followed the pre-processing introduced in their work. Input documents were truncated to 512 tokens.

Aside from various statistics on the datasets, Table 5.1 also reports the proportion of novel bi-grams in gold summaries as a measure of their abstractiveness. We would expect models with extractive biases to perform better on datasets with (mostly) extractive summaries, and abstractive models to perform more rewrite operations on datasets with abstractive summaries. CNN/DailyMail and NYT are somewhat extractive, while XSum is highly abstractive.

5.4.2 Implementation Details and Comparison Systems

For both extractive and abstractive settings, we used PyTorch, OpenNMT (Klein et al., 2017) and the ‘bert-base-uncased’[1] version of BERT to implement BERTSUM. Both source and target texts were tokenized with BERT’s subwords tokenizer.

Extractive Summarization All extractive models were trained for 50,000 steps on 3 GPUs (GTX 1080 Ti) with gradient accumulation every two steps. Model checkpoints were saved and evaluated on the validation set every 1,000 steps. We selected the top-3 checkpoints based on the evaluation loss on the validation set, and report the averaged results on the test set. We used a greedy algorithm similar to Nallapati et al. (2017) to obtain an oracle summary for each document to train extractive models. The algorithm generates an oracle consisting of multiple sentences which maximize the ROUGE-2 score against the gold summary.

When predicting summaries for a new document, we first use the model to obtain the score for each sentence. We then rank these sentences by their scores from highest to lowest, and select the top-3 sentences as the summary.

During sentence selection we use *Trigram Blocking* to reduce redundancy (Paulus et al., 2018). Given summary S and candidate sentence c, we skip c if there exists a

[1]`https://git.io/fhbJQ`

trigram overlapping between c and S. The intuition is similar to Maximal Marginal Relevance (MMR; Carbonell and Goldstein 1998); we wish to minimize the similarity between the sentence being considered and sentences which have been already selected as part of the summary.

We compared our model with several previously proposed systems. SUMMARUNNER (Nallapati et al., 2017) is one of the earliest neural approaches adopting an encoder based on Recurrent Neural Networks. REFRESH (Narayan et al., 2018b) is a reinforcement learning-based system trained by globally optimizing the ROUGE metric. LATENT (Zhang et al., 2018c) frames extractive summarization as a latent variable inference problem; instead of maximizing the likelihood of "gold" standard labels, their latent model directly maximizes the likelihood of human summaries given selected sentences. NEUSUM (Zhou et al., 2018) scores and selects sentences jointly and represents the state-of-the-art in extractive summarization. We also show results of the SUMO system described in the previous chapter.

Abstractive Summarization In all abstractive models, we applied dropout (with probability 0.1) before all linear layers; label smoothing (Szegedy et al., 2016) with smoothing factor 0.1 was also used. Our Transformer decoder has 768 hidden units and the hidden size for all feed-forward layers is 2,048. All models were trained for 200,000 steps on 4 GPUs (GTX 1080 Ti) with gradient accumulation every five steps. Model checkpoints were saved and evaluated on the validation set every 2,500 steps. We selected the top-3 checkpoints based on their evaluation loss on the validation set, and report the averaged results on the test set.

During decoding we use beam search with beam size 5 and length penalty (Wu et al., 2016). The length penalty adds a penalty term $lp(Y)$ to the score of each candidate summary Y:

$$lp(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}} \tag{5.8}$$

where $|Y|$ is the current length of the generated text, and α is the length normalization coefficient. We tuned α between 0.6 and 1 on the validation set; we decode until an end-of-sequence token is emitted and repeated trigrams are blocked (Paulus et al., 2018). It is worth noting that our decoder applies neither a copy nor a coverage mechanism (See et al., 2017), despite their popularity in abstractive summarization. This is mainly because we focus on building a minimum-requirements model and these mechanisms may introduce additional hyper-parameters to tune. Thanks to the subwords tokenizer, we also rarely observe issues with out-of-vocabulary words in the output; moreover, trigram-blocking produces diverse summaries managing to reduce repetitions.

We compared our model with several previously proposed systems. PTGEN (See et al., 2017) is a summarizer based on pointer-generator network which allows it to copy words from the source text, and it can be enhanced by a coverage mechanism (COV) which keeps track of words that have been summarized. Deep Communicating Agents (DCA) model (Celikyilmaz et al., 2018) uses multiple agents (encoders) to represent the document together, and is trained end-to-end with reinforcement learning. DRM (Paulus et al., 2018) is a deep reinforced mode which handles the coverage problem with an intra-attention mechanism where the decoder attends over previously generated words. BOTTOMUP (Gehrmann et al., 2018) follows a bottom-up approach; a content selector first determines which phrases in the source document should be part of the summary, and a copy mechanism is applied only to preselected phrases during decoding. TCONVS2S (Narayan et al., 2018a) is an abstractive model which is particularly suited to extreme summarization (i.e., single sentence summaries), based on convolutional neural networks and additionally conditioned on topic distributions.

5.5 Results

5.5.1 Automatic Evaluation

We evaluated summarization quality automatically using ROUGE (Lin, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

Table 5.2 summarizes our results on the CNN/DailyMail dataset. The first block in the table includes the results of an extractive ORACLE system, which generates summaries by maximizing the ROUGE-2 score, as an upper bound. We also present the LEAD-3 baseline (which simply selects the first three sentences in a document).

The second block in the table includes various extractive models trained on the CNN/DailyMail dataset, including the SUMO model as described in the previous chapter. For comparison to our own model, we also implemented a non-pretrained Transformer baseline (TransformerEXT) which uses the same architecture as BERTSUMEXT, but with fewer parameters. It is randomly initialized and only trained on the summa-

Model	ROUGE-1	ROUGE-2	ROUGE-L
ORACLE	52.59	31.24	48.87
LEAD-3	40.42	17.62	36.67
Extractive			
SUMMARUNNER (Nallapati et al., 2017)	39.60	16.20	35.30
REFRESH (Narayan et al., 2018b)	40.00	18.20	36.60
LATENT (Zhang et al., 2018c)	41.05	18.77	37.54
NEUSUM (Zhou et al., 2018)	41.59	19.01	37.98
SUMO (Liu et al., 2019)	41.00	18.40	37.20
TransformerEXT	40.90	18.02	37.17
Abstractive			
PTGEN (See et al., 2017)	36.44	15.66	33.42
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38
DRM (Paulus et al., 2018)	39.87	15.82	36.90
BOTTOMUP (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA (Celikyilmaz et al., 2018)	41.69	19.47	37.92
TransformerABS	40.21	17.76	37.09
BERT-based			
BERTSUMEXT	43.25	20.24	39.63
BERTSUMEXT w/o interval embeddings	43.20	20.22	39.59
BERTSUMEXT (large)	43.85	20.34	39.90
BERTSUMABS	41.72	19.39	38.76
BERTSUMEXTABS	42.13	19.60	39.18

Table 5.2: ROUGE F1 results on **CNN/DailyMail** test set (ROUGE-1 and ROUGE-2 are unigram and bigram overlaps; ROUGE-L is the longest common subsequence). Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

Model	ROUGE-1	ROUGE-2	ROUGE-L
ORACLE	49.18	33.24	46.02
LEAD-3	39.58	20.11	35.78
Extractive			
COMPRESS (Durrett et al., 2016)	42.20	24.90	—
SUMO (Liu et al., 2019)	42.30	22.70	38.60
TransformerEXT	41.95	22.68	38.51
Abstractive			
PTGEN (See et al., 2017)	42.47	25.61	—
PTGEN + COV (See et al., 2017)	43.71	26.40	—
DRM (Paulus et al., 2018)	42.94	26.02	—
TransformerABS	35.75	17.23	31.41
BERT-based			
BERTSUMEXT	46.66	26.35	42.62
BERTSUMABS	48.92	30.84	45.41
BERTSUMEXTABS	49.02	31.02	45.55

Table 5.3: ROUGE Recall results on **NYT** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software. Table cells are filled with — whenever results are not available.

rization task. TransformerEXT has 6 layers, the hidden size is 512, and the feed-forward filter size is 2,048. The model was trained with same settings as in Vaswani et al. (2017).

The third block in Table 5.2 highlights the performance of several abstractive models on the CNN/DailyMail dataset. We also include an abstractive Transformer baseline (TransformerABS) which has the same decoder as our abstractive BERTSUM models; the encoder is a 6-layer Transformer with 768 hidden size and 2,048 feed-forward filter size.

The fourth block reports results with fine-tuned BERT models: BERTSUMEXT and its two variants (one without interval embeddings, and one with the large version of BERT), BERTSUMABS, and BERTSUMEXTABS. BERT-based models outperform the LEAD-3 baseline which is not a strawman; on the CNN/DailyMail corpus it is indeed superior to several extractive (Nallapati et al., 2017; Narayan et al., 2018b; Zhou et al., 2018) and abstractive models (See et al., 2017). BERT models collectively outperform

Model	ROUGE-1	ROUGE-2	ROUGE-L
ORACLE	29.79	8.81	22.66
LEAD	16.30	1.60	11.95
Abstractive			
PTGEN (See et al., 2017)	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	28.10	8.02	21.72
TCONVS2S (Narayan et al., 2018a)	31.89	11.54	25.75
TransformerABS	29.41	9.77	23.01
BERT-based			
BERTSUMABS	38.76	16.33	31.15
BERTSUMEXTABS	38.81	16.50	31.27

Table 5.4: ROUGE F1 results on the **XSum** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

all previously proposed extractive and abstractive systems, only falling behind the ORACLE upper bound. Among BERT variants, BERTSUMEXT performs best which is not entirely surprising; CNN/DailyMail summaries are somewhat extractive and even abstractive models are prone to copying sentences from the source document when trained on this dataset (See et al., 2017). Perhaps unsurprisingly we observe that larger versions of BERT lead to performance improvements and that interval embeddings bring only slight gains.

Table 5.3 presents results on the NYT dataset. Following the evaluation protocol in Durrett et al. (2016), we use limited-length ROUGE Recall, where predicted summaries are truncated to the length of the gold summaries. Again, we report the performance of the ORACLE upper bound and LEAD-3 baseline. The second block in the table contains previously proposed extractive models as well as our own Transformer baseline. COMPRESS (Durrett et al., 2016) is an ILP-based model which combines compression and anaphoricity constraints. The third block includes abstractive models from the literature, and our Transformer baseline. BERT-based models are shown in the fourth block. Again, we observe that they outperform previously proposed approaches. On this dataset, abstractive BERT models generally perform better compared to BERTSUMEXT, almost approaching ORACLE performance.

Table 5.4 summarizes our results on the XSum dataset. Recall that summaries

$\tilde{lr}_{\mathfrak{E}}$ \ $\tilde{lr}_{\mathfrak{D}}$	1	0.1	0.01	0.001
2e^{-2}	50.69	9.33	10.13	19.26
2e^{-3}	37.21	8.73	9.52	16.88

Table 5.5: Model perplexity (CNN/DailyMail; validation set) under different combinations of encoder and decoder learning rates.

in this dataset are highly abstractive (see Table 5.1) consisting of a single sentence conveying the gist of the document. Extractive models here perform poorly as corroborated by the low performance of the LEAD baseline (which simply selects the leading sentence from the document), and the ORACLE (which selects a single-best sentence in each document) in Table 5.4. As a result, we do not report results for extractive models on this dataset. The second block in Table 5.4 presents the results of various abstractive models taken from Narayan et al. (2018a) and also includes our own abstractive Transformer baseline. In the third block we show the results of our BERT summarizers which again are superior to all previously reported models (by a wide margin).

5.5.2 Model Analysis

Learning Rates Recall that our abstractive model uses separate optimizers for the encoder and decoder. In Table 5.5 we examine whether the combination of different learning rates ($\tilde{lr}_{\mathfrak{E}}$ and $\tilde{lr}_{\mathfrak{D}}$) is indeed beneficial. Specifically, we report model perplexity on the CNN/DailyMail validation set for varying encoder/decoder learning rates. We can see that the model performs best with $\tilde{lr}_{\mathfrak{E}} = 2\text{e}^{-3}$ and $\tilde{lr}_{\mathfrak{D}} = 0.1$.

Ablation Studies We conducted various ablation studies to examine the contribution of different components of BERTSUM. The results are shown in Table 5.6. In the extractive setting, interval segments and inter-sentence Transformer layers increase the performance of the base model. In the abstractive setting, interval segments also seem to enhance the quality of the output summaries. When BERTSUM does not employ separate optimizers, i.e., the encoder and decoder are optimized with the same learning rate and warmup-steps, ROUGE scores drop dramatically.

Position of Extracted Sentences In addition to the evaluation based on ROUGE, we also analyzed in more detail the summaries produced by our model. For the ex-

Model	ROUGE-1	ROUGE-2	ROUGE-L
BERTSUMEXT	43.25	20.24	39.63
−interval segments	43.14	20.01	39.22
−inter-sent Transformer	43.23	20.22	39.60
BERTSUMABS	41.72	19.39	38.76
−interval segments	41.54	19.31	38.76
−seperate optimizers	39.77	16.89	36.53

Table 5.6: Results for ablation studies of BERTSUMEXT and BERTSUMABS on the CNN/DailyMail test set using ROUGE F1.

tractive setting, we looked at the position (in the source document) of the sentences which were selected to appear in the summary. Figure 5.2 shows the proportion of selected summary sentences which appear in the source document at positions 1, 2, and so on. The analysis was conducted on the CNN/DailyMail dataset for Oracle summaries, and those produced by BERTSUMEXT and the TransformerEXT. We can see that Oracle summary sentences are fairly smoothly distributed across documents, while summaries created by TransformerEXT mostly concentrate on the first document sentences. BERTSUMEXT outputs are more similar to Oracle summaries, indicating that with the pretrained encoder, the model relies less on shallow position features, and learns deeper document representations.

Novel N-grams We also analyzed the output of abstractive systems by calculating the proportion of novel n-grams that appear in the summaries but not in the source texts. The results are shown in Figure 5.3. In the CNN/DailyMail dataset, the proportion of novel n-grams in automatically generated summaries is much lower compared to reference summaries, but in XSum, this gap is much smaller. We also observe that on CNN/DailyMail, BERTEXTABS produces less novel n-grams than BERTABS, which is not surprising. BERTEXTABS is more biased towards selecting sentences from the source document since it is initially trained as an extractive model.

5.5.3 Human Evaluation

In addition to automatic evaluation, we also evaluated system output by eliciting human judgments. We report experiments following a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018b) which quantifies the degree to which

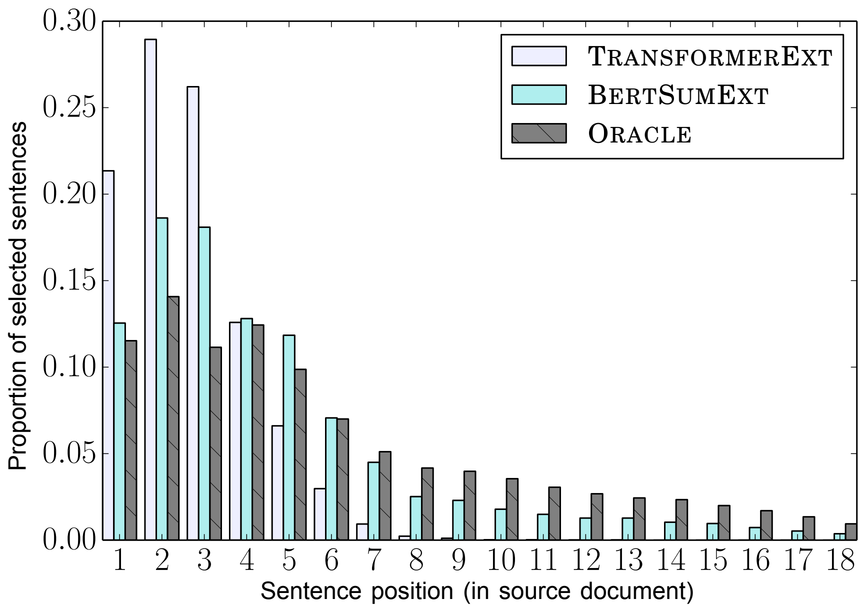
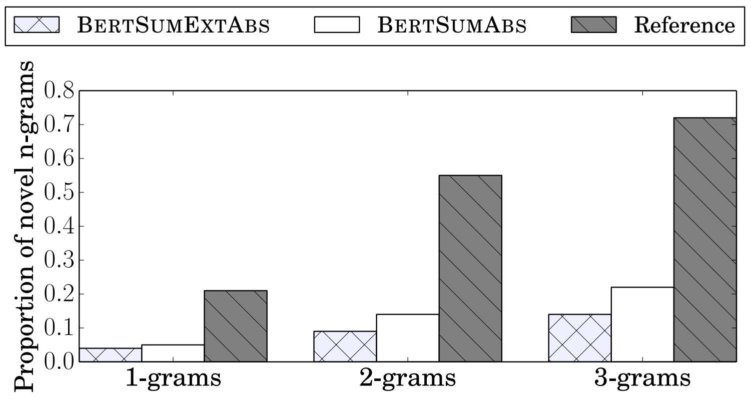


Figure 5.2: Proportion of extracted sentences according to their position in the original document.

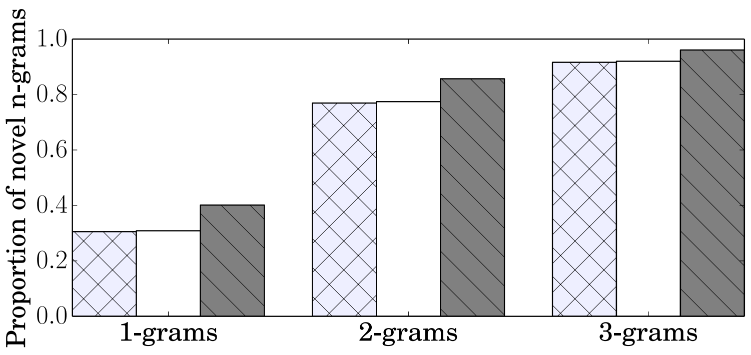
summarization models retain key information from the document. Under this paradigm, a set of questions is created based on the gold summary under the assumption that it highlights the most important document content. Participants are then asked to answer these questions by reading system summaries alone without access to the article. The more questions a system can answer, the better it is at summarizing the document as a whole.

Moreover, we also assessed the overall quality of the summaries produced by abstractive systems which due to their ability to rewrite content may produce disfluent or ungrammatical output. Specifically, we followed the Best-Worst Scaling (Kiritchenko and Mohammad, 2017) method where participants were presented with the output of two systems (and the original document) and asked to decide which one was better according to the criteria of *Informativeness*, *Fluency*, and *Succinctness*.

Both types of evaluation were conducted on the Amazon Mechanical Turk platform. Detailed instructions of human evaluation can be found in Appendix A. For the CNN/DailyMail and NYT datasets we used the same documents (20 in total) and questions from previous work (Narayan et al., 2018b; Liu et al., 2019). For XSum, we randomly selected 20 documents (and their questions) from the release of Narayan et al. (2018a). We elicited 3 responses per HIT. With regard to QA evaluation, we



(a) CNN/DailyMail Dataset



(b) XSum dataset

Figure 5.3: Proportion of novel n-grams in model generated summaries.

Extractive	CNN/DM	NYT
Lead	42.5†	36.2†
NeuSum	42.2†	—
Sumo	41.7†	38.1†
Transformer	37.8†	32.5†
BertSum	58.9	41.9

Table 5.7: QA-based evaluation. Models with † are significantly different from BertSum (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available.

adopted the scoring mechanism from Clarke and Lapata (2010); correct answers were marked with a score of one, partially correct answers with 0.5, and zero otherwise. For quality-based evaluation, the rating of each system was computed as the percentage of times it was chosen as better minus the times it was selected as worse. Ratings thus range from -1 (worst) to 1 (best).

	CNN/DM		NYT		XSum	
Abstractive	QA	Rank	QA	Rank	QA	Rank
LEAD	$42.5^\dagger$	—	$36.2^\dagger$	—	$9.20^\dagger$	—
PTGEN	$33.3^\dagger$	$-0.24^\dagger$	$30.5^\dagger$	$-0.27^\dagger$	$23.7^\dagger$	$-0.36^\dagger$
BOTTOMUP	$40.6^\dagger$	$-0.16^\dagger$	—	—	—	—
TCONVS2S	—	—	—	—	52.1	$-0.20^\dagger$
GOLD	—	$0.22^\dagger$	—	$0.33^\dagger$	—	$0.38^\dagger$
BERTSUM	56.1	0.17	41.8	-0.07	57.5	0.19

Table 5.8: QA-based and ranking-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available. GOLD is not used in the QA setting, and LEAD is not used in the Rank evaluation.

Results for extractive and abstractive systems are shown in Tables 5.7 and 5.8, respectively. We compared the best performing BERTSUM model in each setting (extractive or abstractive) against various state-of-the-art systems (whose output is publicly available), the LEAD baseline, and the GOLD standard as an upper bound. As shown in both tables participants overwhelmingly prefer the output of our model against comparison systems across datasets and evaluation paradigms. All differences between BERTSUM and comparison models are statistically significant ($p < 0.05$), with the exception of TCONVS2S (see Table 5.8; XSum) in the QA evaluation setting.

5.5.4 Examples of System Output and Evaluation Questions

Table 5.9 shows examples of system output on the CNN/DailyMail dataset. Specifically, we show summaries produced from our BERT-based models, NEUSUM (Zhou et al., 2018), the pointer-generator network (PTGEN; See et al. 2017) and the bottom-up (BOTTOMUP) summarization model of Gehrmann et al. (2018). Table 5.10 shows examples of system output on the NYT datasets. Specifically, we show summaries produced from our BERT-based models, SUMO and the pointer-generator network (See et al., 2017). Table 5.11 shows examples of system output on the XSum dataset. Specifically, we show summaries produced from our BERT-based models, the pointer-generator network (See et al., 2017) and the topic-conditioned convolutional model (TCONVS2S) of Narayan et al. (2018a). For extractive setting, we can see BERTSUMEXT produced summaries with more coverage of the source input. For example, in Table 5.9, BERTSUMEXT selected content not only about the women but also about the police's operations. For abstractive setting, BERTSUMABS produced more fluent and informative summaries than baseline systems and can better answer the questions in the question answering evaluation. For example, in Table 5.10, BERTSUMABS provided more background information of the grant program and the complete event chain, while PGNET only covers the grant program itself. And BERTSUMEXTABS tended to generate longer sentences with more copies from the source input compared with BERTSUMABS.

5.6 Summary

In this chapter, we showcased how pretrained BERT can be usefully applied in text summarization. We introduced a novel document-level encoder and proposed a general framework for both abstractive and extractive summarization. Experimental results across three datasets show that our model achieves state-of-the-art results across the board under automatic and human-based evaluation protocols.

CNN/DailyMail	
GOLD	Emergency services were called to the Kosciuszko bridge at about 11.50 am Monday, where a woman had climbed over the bridge's railing and was standing on a section of metal piping. Officers tried to calm her down as NYPD patrol boats cruised under the bridge on Newtown Creek, which connects Greenpoint in Brooklyn and Maspeth in Gueens. A witness said the woman was a 44-year-old Polish mother-of-one who was going through a tough divorce. She agreed to be rescued after police talked to her about her daughter and was taken to Elmhurst hospital.
QA	When were emergency services called to the Kosciuszko bridge? [11.50 am] What did the witness say about the woman? [44-year-old polish mother-of-one who was going through a tough divorce] Did the woman agreed to be rescued? [yes]
LEAD	A woman who threatened to jump to her death off a New York City bridge was saved by a group of hero cops who spent more than two hours talking the woman out of jumping. The unidentified woman from Greenpoint, Brooklyn, was apparently distraught over her troubled marriage. She walked to the Kosciuszko bridge from the Brooklyn side shortly before noon on Monday, climbed over a railing and stood on a section of metal piping barely wide enough to fit her feet.
Extractive Models	
NEUSUM	A woman walked on to New York city's Kosciuszko bridge from the Brooklyn side on Monday just before noon, climbed over the railing and stood on a section of metal piping 125 feet above Newtown Creek. A witness said the woman is a 44-year-old Polish mother-of-one who was going through a tough divorce with her husband. The unidentified woman from Greenpoint, Brooklyn, was apparently distraught over her troubled marriage.

Table 5.9: GOLD human authored summaries, questions based on them (answers shown in square brackets), and automatic summaries produced by our BERT-based models NEUSUM (Zhou et al., 2018), a pointer-generator network (See et al., 2017) and a bottom-up summarizer (Gehrmann et al., 2018) for a CNN/DailyMail article.

BERTSUMEXT	The unidentified woman from Greenpoint, Brooklyn, was apparently distraught over her troubled marriage. A woman walked on to New York City's Kosciuszko bridge from the Brooklyn side on Monday just before noon, climbed over the railing and stood on a section of metal piping 125 feet above Newtown Creek. Police arrived on scene at about 11.50 am and spent more than two hours talking to the woman and trying to calm her down.
Abstractive Models	
PTGEN	A woman who threatened to jump to her death off a New York bridge was apparently distraught over her troubled marriage. She walked to the Kosciuszko bridge from the Brooklyn side shortly before noon on Monday, climbed over the railing. Police arrived on scene at about 11.50am and spent more than two hours talking to the woman and trying to calm her down.
BOTTOMUP	The woman from Greenpoint, Brooklyn, was apparently distraught over her troubled marriage. She walked to the Kosciuszko bridge from the Brooklyn side shortly before noon on Monday, climbed over a railing and stood on a section of metal piping barely wide enough to fit her feet. After she was pulled to safety by police officers, the woman was taken by ambulance to Elmhurst hospital for observation.
BERTSUMABS	The unidentified woman from Greenpoint, Brooklyn , was apparently distraught over her troubled marriage. She walked to the Kosciuszko bridge from the Brooklyn side shortly before noon on Monday, climbed over a railing and stood on a section of metal piping 125 feet above Newtown Creek. At about 2pm the woman agreed to be rescued and NYPD officers were able to pull her to safety over the railing.
BERTSUMEXTABS	The unidentified woman from Greenpoint, Brooklyn, was apparently distraught over her troubled marriage. She walked to the Kosciuszko bridge from the brooklyn side shortly before noon on Monday, climbed over a railing and stood on a section of metal piping 125 feet above Newtown Creek. She agreed to be rescued after police officers spoke to her about her daughter, and was lifted off the bridge 'so she could be with her'. a witness said the woman is a 44-year-old polish mother-of-one who was going through a tough divorce with her husband.

Table 5.9 Continued

	NYT
GOLD	Louisiana officials set July 31 deadline for applicants for the Road Home, grant program for homeowners who lost their houses to hurricanes Katrina and Rita. Program is expected to cost far more than $7.5 billion provided by Federal Government, in part because many more families have applied than officials anticipated. With cutoff date, State hopes to figure out how much more money it needs to pay for program. Shortfall is projected to be $2.9 billion.
QA	What is Road Home? [The Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita] When is the applicants' deadline for the Road Home? [july 31] What is the shortfall projected to be? [$2.9 billion]
LEAD	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the Federal Government, in part because many more families have applied than officials had anticipated. As a result, Louisiana officials on Tuesday night set a july 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses. With the cutoff date, the State hopes to be able to figure out how much more money it needs to pay for the program.
	Extractive Models
SUMO	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the Federal Government, in part because many more families have applied than officials had anticipated. As a result, Louisiana officials on Tuesday night set a july 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses. The shortfall is projected to be $2.9 billion.

Table 5.10: GOLD human authored summaries, questions based on them (answers shown in square brackets) and automatic summaries produced by our BERT-based models, SUMO and the pointer-generator network (See et al., 2017) for a NYT article.

BERTSUMEXT	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the Federal Government, in part because many more families have applied than officials had anticipated. The shortfall is projected to be $2.9 billion. Only 22,000 families statewide, out of 140,000 applicants, have received grants so far, for a total of $1.3 billion.
Abstractive Models	
PGNET	Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita is expected to cost far more than $7.5 billion provided by Federal Government. Louisiana officials set July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses.
BERTSUMABS	Road Home, Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than $ 7.5 billion provided by Federal Government, in part because many more families have applied than officials had anticipated. State hopes to be able to figure out how much more money it needs to pay for program.
BERTSUMEXTABS	Road Home, Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than $7.5 billion provided by Federal Government, in part because many more families have applied than officials had anticipated. State hopes to be able to figure out how much more money it needs to pay for program. Financial woes of Road Home have set off frenzy of fingerpointing between Federal and State officials.

Table 5.10 Continued

XSum	
GOLD	A council plans to employ its own staff to help young people with mental health problems.
QA	Why are young people getting help? [mental health problems] Who is employing staff for this purpose? [the council]
LEAD	West Berkshire Council is setting up an emotional health academy to train psychology graduates and health professionals.
Abstractive Models	
PGNET	A mental health academy in Berkshire has been put up for sale in a bid to reduce the number of mental health patients.
TCONVS2S	A new academy for children with mental health problems is being launched in West Berkshire.
BERTSUMABS	A new mental health academy is to be launched in West Berkshire in a bid to raise awareness of mental health problems.
BERTSUMEXTABS	An academy to train mental health patients with mental health problems is being launched by West Berkshire Council.

Table 5.11: GOLD human authored summaries, questions based on them (answers shown in square brackets) and automatic summaries produced by BERT-based models, a pointer-generator Network (See et al., 2017) and TConvS2S (Narayan et al., 2018a) for a XSum article.

Chapter 6

Hierarchical Models for Multi-Document Summarization

In the previous chapters, we focused on the task of *single*-document summarization. As we discussed, with the popularity of neural network models and the availability of large-scale datasets, in recent years, single-document summarization has enjoyed extensive interest. On the other hand, another important summarization task, *multi*-document summarization — the task of producing summaries from clusters of thematically related documents — has received significantly less attention, partly due to the paucity of suitable data for the application of learning methods. Very recently, Liu et al. (2018) tap into the potential of Wikipedia and propose a methodology for creating a large-scale dataset (WikiSum) for multi-document summarization with hundreds of thousands of instances. However, introducing neural models for multi-document summarization still faces several challenges, including the large size of input texts, modeling relations across documents and their hierarchical structures. In this chapter, we first give more details on these challenges, then propose several solutions and extend more modeling efforts to neural multi-document summarization.

6.1 Introduction

Different from single-document summarization, multi-document summarization aims to produce a summary containing the most important information from multiple input documents. Therefore, multi-document summarization has different applications compared to the single-document setting. It can be applied to summarize multiple related news articles and product reviews. One advantage of multi-document summarization is that it can be integrated with a information retrieval system, like a search engine. In such situations, there will be a large set of related documents given a query, and the ability of processing and summarizing multiple documents becomes important. Although there are some techniques that could be shared by single-document summarization and multi-document summarization, there are several major differences:

1. The redundancy within the set of documents is higher than the redundancy in one single document, since each document could describe similar background information of the topic. Therefore, how to model and reduce redundancy is more crucial for multi-document summarization.

2. The compression ratio (i.e. the size of the summary against the size of the input texts) of multi-document summarization will be smaller than single-document summarization. (Goldstein et al., 2000)

3. Unlike single-document summarization, for multi-document summarization, the relations among documents is an important signal. For example, co-references (Azzam et al., 1999), entity relations (Christensen et al., 2013) and event relations (White et al., 2001) are found that can provide useful information for generating multi-document summaries.

While many new models have been proposed for single-document summarization in the deep learning era, multi-document summarization has been relatively neglected, partially due to the paucity of large-scale datasets which are essential for training a neural network model. High-quality multi-document summarization datasets (i.e., document clusters paired with multiple reference summaries written by humans) have been produced for the Document Understanding and Text Analysis Conferences (DUC[1] and TAC[2]), but are relatively small (in the range of a few hundred examples) for training neural models. In an attempt to drive research further, Liu et al. (2018) tap into the potential of Wikipedia and propose a methodology for creating a large-scale dataset (WikiSum) for multi-document summarization with hundreds of thousands of instances. Wikipedia articles, specifically lead sections, are viewed as summaries of various topics indicated by their title, e.g.,"Florence" or "Natural Language Processing". Documents cited in the Wikipedia articles or web pages returned by Google (using the section titles as queries) are seen as the source cluster which the lead section purports to summarize.

[1]http://duc.nist.gov/
[2]https://tac.nist.gov/

Aside from the difficulties in obtaining training data, a major obstacle to the application of end-to-end models to multi-document summarization is the sheer size and number of source documents which can be very large. As a result, it is practically infeasible (given memory limitations of current hardware) to directly train a encoder-decoder model as used in previous sections to encode all input texts into vectors and subsequently generates a summary from them. Liu et al. (2018) propose a two-stage architecture, where an *extractive* model first selects a subset of salient passages, and subsequently an *abstractive* model generates the summary while conditioning on the extracted subset. The selected passages are concatenated into a flat sequence and the Transformer (Vaswani et al., 2017) is used to decode the summary.

Although the model of Liu et al. (2018) takes an important first step towards abstractive multi-document summarization, it still considers multiple input documents as a concatenated flat sequence. However, in multi-document summarization, each source input is first composed from tokens to a document and then from multiple documents to a complete input. This hierarchical structure of the input has been agnostic by Liu et al. (2018). Also, the relations that might exist among document are not modeled. For example, different web pages might repeat the same content, include additional content, present contradictory information, or discuss the same fact in a different light (Radev, 2000). The realization that cross-document links (Zhang et al., 2002) are important in isolating salient information, eliminating redundancy, and creating overall coherent summaries, has led to the widespread adoption of graph-based models for multi-document summarization (Erkan and Radev, 2004; Christensen et al., 2013; Wan, 2008; Parveen and Strube, 2014). Graphs conveniently capture the relationships between textual units within a document collection and can be easily constructed under the assumption that text spans represent graph nodes and edges are semantic links between them.

In this chapter, we develop a neural summarization model which can effectively process multiple input documents and distill abstractive summaries. Our model augments the vanilla Transformer architecture with the ability to encode multiple documents in a hierarchical manner. We represent cross-document relationships via an attention mechanism which allows to share information across multiple documents as opposed to simply concatenating text spans and feeding them as a flat sequence to the model. In this way, the model automatically *learns* richer structural dependencies among textual units, thus incorporating well-established insights from earlier work (Erkan and Radev, 2004; Guinaudeau and Strube, 2013). Advantageously, the proposed architecture can easily benefit from information external to the model, i.e., by replacing inter-document attention with a graph-matrix computed based on the basis of lexical similarity (Erkan and Radev, 2004) or discourse relations (Christensen et al., 2013).

We evaluate our model on the WikiSum dataset and show experimentally that the proposed architecture brings substantial improvements over several strong baselines. We also find that the addition of a simple ranking module which scores documents based on their usefulness for the target summary can greatly boost the performance of a multi-document summarization system.

6.2 Related Work

Most previous multi-document summarization methods are extractive operating over graph-based representations of sentences or passages. Approaches vary depending on how edge weights are computed (e.g., based on cosine similarity with tf-idf weights for words (Erkan and Radev, 2004) or on discourse relations (Christensen et al., 2013)), and the specific algorithm adopted for ranking text units for inclusion in the final summary. Several variants of the PageRank algorithm have been adopted in the literature (Erkan and Radev, 2004) in order to compute the importance or salience of a passage recursively based on the entire graph. More recently, Yasunaga et al. (2017) propose a neural version of this framework, where salience is estimated using features extracted from sentence embeddings and graph convolutional networks (Kipf and Welling, 2017b) applied over the relation graph representing cross-document links.

Abstractive approaches have met with limited success. A few systems generate summaries based on sentence fusion, a technique which identifies fragments conveying common information across documents and combines these into sentences (Barzilay and McKeown, 2005; Filippova and Strube, 2008; Bing et al., 2015). Although neural abstractive models have achieved promising results on single-document summarization (See et al., 2017; Paulus et al., 2018; Gehrmann et al., 2018; Celikyilmaz et al., 2018), the extension of sequence-to-sequence architectures to multi-document summarization is less straightforward. Apart from the lack of sufficient training data, neural models also face the computational challenge of processing multiple source documents. Previous solutions include model transfer (Zhang et al., 2018b; Lebanoff and Liu, 2018), where a sequence-to-sequence model is pretrained on single-document summarization data and fine-tuned on DUC (multi-document) benchmarks. Lebanoff et al. (2018)

combines a neural encoder-decoder model for single-document summarization, and an extractive system for identifying important sentences from multiple input documents. The extractive system modifies the attention distribution of the encoder-decoder model for the final generation of the summary. Unsupervised models were also developed. Ma et al. (2016) propose to train a document representation model by reconstructing the source document from summary sentences. They apply this document representation model to multiple documents and obtain a set of candidate summary sentences and filter a subset by beam search. Chu and Liu (2019) propose MeanSum, an unsupervised abstractive summarization model composed of an auto-encoder module and a language modeling module. It is applied to generate summaries of multiple reviews. Fabbri et al. (2019) propose a new dataset for multi-document summarization of news articles. They also develop a new pointer-generator network by introducing an additional sentence-level attention mechanism combined with the Maximal Marginal Relevance (Carbonell and Goldstein, 1998) method to determine salient sentences of the source documents.

Liu et al. (2018) propose a methodology for constructing large-scale summarization datasets and a two-stage model which first extracts salient information from source documents and then uses a decoder-only architecture (that can attend to very long sequences) to generate the summary. We follow their setup in viewing multi-document summarization as a supervised machine learning problem and for this purpose assume access to large, labeled datasets (i.e., source documents-summary pairs). In contrast to their approach, we use a learning-based ranker and our abstractive model can hierarchically encode the input documents, with the ability to learn latent relations across documents and additionally incorporate information encoded in well-known graph representations.

6.3 Model Description

We follow Liu et al. (2018) in treating the generation of lead Wikipedia sections as a multi-document summarization task. The input to a hypothetical system is the title of a Wikipedia article and a collection of source documents, while the output is the Wikipedia article's first section. Source documents are webpages cited in the *References* section of the Wikipedia article and the top 10 search results returned by Google (with the title of the article as the query). Since source documents could be relatively long, they are split into multiple paragraphs by line-breaks. More formally, given ti-

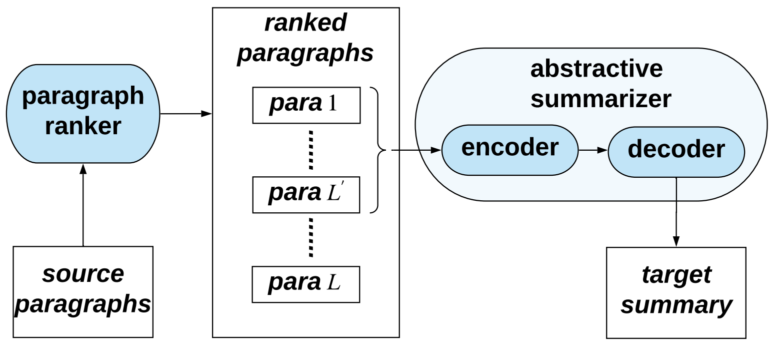


Figure 6.1: Pipeline of our multi-document summarization system. L source paragraphs are first ranked and the L'-best ones serve as input to an encoder-decoder model which generates the target summary.

tle T, and L input paragraphs $\{P_1, \cdots, P_L\}$ (retrieved from Wikipedia citations and a search engine), the task is to generate the lead section D of the Wikipedia article.

Our summarization system is illustrated in Figure 6.1. Since the input paragraphs are numerous and possibly lengthy, instead of directly applying an abstractive system, we first rank them and summarize the L'-best ones. Our summarizer follows the encoder-decoder architecture as described in Section 2.2.2. In this chapter, we focus exclusively on the encoder part of the model, our decoder follows the Transformer architecture introduced in Section 2.2.2; it generates a summary token by token while attending to the source input.

6.3.1 Paragraph Ranking

Unlike Liu et al. (2018) who rank paragraphs based on their similarity with the title (using tf-idf-based cosine similarity), we adopt a learning-based approach. A logistic regression model is applied to each paragraph to calculate a score indicating whether it should be selected for summarization. We use two recurrent neural networks with Long-Short Term Memory units (LSTM; Hochreiter and Schmidhuber 1997) to represent title T and source paragraph P:

$$\{\mathbf{u}_{t1}, \cdots, \mathbf{u}_{tm}\} = \text{lstm}_t(\{\mathbf{w}_{t1}, \cdots, \mathbf{w}_{tm}\}) \tag{6.1}$$

$$\{\mathbf{u}_{p1}, \cdots, \mathbf{u}_{pn}\} = \text{lstm}_p(\{\mathbf{w}_{p1}, \cdots, \mathbf{w}_{pn}\}) \tag{6.2}$$

where w_{ti}, w_{pj} are word embeddings for tokens in T and P, and u_{ti}, u_{pj} are the updated vectors for each token after applying the LSTMs.

A max-pooling operation is then used over title vectors to obtain a fixed-length vector $\hat{\mathbf{u}}_t$ to represent features of the title:

$$\hat{\mathbf{u}}_t = \text{maxpool}(\{\mathbf{u}_{t1}, \cdots, \mathbf{u}_{tm}\}) \tag{6.3}$$

We concatenate $\hat{\mathbf{u}}_t$ with vector $\mathbf{u}_{pi}$ of each token in the paragraph and apply a non-linear transformation to extract features for matching the title and the paragraph. A second max-pooling operation yields the final paragraph vector $\hat{\mathbf{p}}$:

$$\mathbf{p}_i = \tanh(\mathbf{W}_1([\mathbf{u}_{pi}; \hat{\mathbf{u}}_t])) \tag{6.4}$$

$$\hat{\mathbf{p}} = \text{maxpool}(\{\mathbf{p}_1, \cdots, \mathbf{p}_n\}) \tag{6.5}$$

where $\mathbf{p}_i$ is the matched vector with information from both the title and the i-th token in the paragraph; tanh is the hyperbolic tangent activation function.

Finally, to estimate whether a paragraph should be selected, we use a linear transformation and a sigmoid function over paragraph vector $\hat{\mathbf{p}}$:

$$s = \sigma(\mathbf{W}_2(\hat{\mathbf{p}})) \tag{6.6}$$

where s is the score indicating whether paragraph P should be used for summarization.

All input paragraphs $\{P_1, \cdots, P_L\}$ receive scores $\{\mathbf{s}_1, \cdots, \mathbf{s}_L\}$. The model is trained by minimizing the cross entropy loss between $\mathbf{s}_i$ and ground-truth scores $\mathbf{y}_i$ denoting the relatedness of a paragraph to the gold standard summary. We adopt ROUGE-2 recall (of paragraph P_i against gold target text D) as $\mathbf{y}_i$. In testing, input paragraphs are ranked based on the model predicted scores and an ordering $\{R_1, \cdots, R_L\}$ is generated. The first L' paragraphs $\{R_1, \cdots, R_{L'}\}$ are selected as input to the second abstractive stage.

6.3.2 Paragraph Encoding

Instead of treating the selected paragraphs as a very long sequence, we develop a hierarchical model based on the Transformer architecture (Vaswani et al., 2017) to capture inter-paragraph relations. The model is composed of several *local* and *global* Transformer layers which can be stacked freely. Let x_{ij} denote the j-th token in the i-th ranked paragraph R_i; the model takes vectors $\mathbf{x}_{ij}$ (for all tokens) as input. For the l-th Transformer layer, the input will be $\mathbf{h}_{ij}^{l-1}$, and the output is written as $\mathbf{h}_{ij}^{l}$. More details on the vanilla Transformer model can be found in Section 2.1.2.

6.3.2.1 Embeddings

Input tokens are first represented by word embeddings. Let $\mathbf{w}_{ij} \in \mathbb{R}^d$ denote the word embeddings assigned to x_{ij}. Since the Transformer is a non-recurrent model, we also assign a special positional embedding $\mathbf{pe}_{ij}$ to x_{ij}, to indicate the position of the token within the input.

To calculate positional embeddings, we follow Vaswani et al. (2017) and use sine and cosine functions of different frequencies. The embedding $\mathbf{e}_p$ for the p-th element in a sequence is:

$$\mathbf{e}_p[i] = \sin(p/10000^{2i/d}) \tag{6.7}$$

$$\mathbf{e}_p[2i+1] = \cos(p/10000^{2i/d}) \tag{6.8}$$

where $\mathbf{e}_p[i]$ indicates the i-th dimension of the embedding vector.

In multi-document summarization, token x_{ij} has two positions that need to be considered, namely i (the rank of the paragraph) and j (the position of the token within the paragraph). Positional embedding $\mathbf{pe}_{ij} \in \mathbb{R}^d$ represents both positions (via concatenation) and is added to word embedding $\mathbf{w}_{ij}$ to obtain the final input vector $\mathbf{x}^0_{ij}$ of the Transformer model:

$$\mathbf{pe}_{ij} = [\mathbf{e}_i; \mathbf{e}_j] \tag{6.9}$$

$$\mathbf{x}_{ij} = \mathbf{w}_{ij} + \mathbf{pe}_{ij} \tag{6.10}$$

6.3.2.2 Local Transformer Layer

A local Transformer layer is used to encode contextual information for tokens within each paragraph. The local Transformer layer is the same as the vanilla Transformer layer (Vaswani et al., 2017), and composed of two sub-layers (as described in Section 2.1.2.). For the i-th ranked paragraph:

$$\tilde{H}^l_i = \text{LayerNorm}(H^{l-1}_i + \text{MHAtt}(H^{l-1}_i)) \tag{6.11}$$

$$H^l_i = \text{LayerNorm}(\tilde{H}^l_i + \text{FFN}(\tilde{H}^l_i)) \tag{6.12}$$

where $H^l_i = [\boldsymbol{h}^l_{i1}, \cdots, \boldsymbol{h}^l_{in}]$; LayerNorm is layer normalization; MHAtt is the multi-head attention mechanism; and FFN is a two-layer feed-forward network with ReLU as hidden activation function.

6.3.2.3 Global Transformer Layer

A global Transformer layer is used to exchange information across multiple paragraphs. As shown in Figure 6.2, we first apply a multi-head pooling operation to each paragraph. Different heads will encode paragraphs with different attention weights. Then, for each head, an inter-paragraph attention mechanism is applied, where each paragraph can collect information from other paragraphs by self-attention, generating a context vector to capture contextual information from the whole input. Finally, context vectors are concatenated, linearly transformed, added to the vector of each token, and fed to a feed-forward layer, updating the representation of each token with global information.

Multi-head Pooling To obtain fixed-length paragraph representations, we apply a weighted-pooling operation; instead of using only one representation for each paragraph, we introduce a multi-head pooling mechanism, where for each paragraph, weight distributions over tokens are calculated, allowing the model to flexibly encode paragraphs in different representation subspaces by attending to different words.

Let $\mathbf{h}_{ij}^{l-1} \in \mathbb{R}^d$ denote the output vector of the last Transformer layer for token x_{ij}, which is used as input for the current layer. For each paragraph R_i, for $z \in \{1, \cdots, n_{head}\}$, we first transform the input vectors into attention scores $\mathbf{a}_{ij}^z$ and value vectors $\mathbf{b}_{ij}^z$. Then, for each head, we calculate a probability distribution $\hat{\mathbf{a}}_{ij}^z$ over tokens within the paragraph based on attention scores:

$$\mathbf{a}_{ij}^z = \mathbf{W}_a^z \mathbf{h}_{ij}^{l-1} \tag{6.13}$$

$$\mathbf{b}_{ij}^z = \mathbf{W}_b^z \mathbf{h}_{ij}^{l-1} \tag{6.14}$$

$$\hat{\mathbf{a}}_{ij}^z = \exp(\mathbf{a}_{ij}^z) / \sum_{j=1}^{n} \exp(\mathbf{a}_{ij}^z) \tag{6.15}$$

where $\mathbf{W}_a^z \in \mathbb{R}^{1*d}$ and $\mathbf{W}_b^z \in \mathbb{R}^{d_{head}*d}$ are weights. $d_{head} = d/n_{head}$ is the dimension of each head. n is the number of tokens in R_i.

We next apply a weighted summation with another linear transformation and layer normalization to obtain vector $\mathbf{head}_i^z$ for the paragraph:

$$\mathbf{head}_i^z = \text{LayerNorm}(\mathbf{W}_c^z \sum_{j=1}^{n} \mathbf{a}_{ij}^z \mathbf{b}_{ij}^z) \tag{6.16}$$

where $\mathbf{W}_c^z \in \mathbb{R}^{d_{head}*d_{head}}$ is the weight.

The model can flexibly incorporate multiple heads, with each paragraph having multiple attention distributions, thereby focusing on different views of the input. This operation is shown in the bottom part of Figure 6.2.

Inter-paragraph Attention We model the dependencies across multiple paragraphs with an inter-paragraph attention mechanism. Similar to self-attention, inter-paragraph attention allows for each paragraph to attend to other paragraphs by calculating an attention distribution:

$$\mathbf{q}_i^z = \mathbf{W}_q^z \mathbf{head}_i^z \tag{6.17}$$

$$\mathbf{k}_i^z = \mathbf{W}_k^z \mathbf{head}_i^z \tag{6.18}$$

$$\mathbf{v}_i^z = \mathbf{W}_v^z \mathbf{head}_i^z \tag{6.19}$$

$$\mathbf{context}_i^z = \sum_{i'=1}^{m} \frac{\exp(\mathbf{q}_i^z \mathbf{k}_{i'}^{z\,T})}{\sum_{o=1}^{m} \exp(\mathbf{q}_i^z \mathbf{k}_{\mathbf{o}}^{z\,T})} \mathbf{v}_{i'}^z \tag{6.20}$$

where $\mathbf{q}_i^z, \mathbf{k}_i^z, \mathbf{v}_i^z \in \mathbb{R}^{d_{head} * d_{head}}$ are query, key, and value vectors that are linearly transformed from $\mathbf{head}_i^z$ as described in Section 2.1.2; $\mathbf{context}_i^z \in \mathbb{R}^{d_{head}}$ represents the context vector generated by a self-attention operation over all paragraphs. m is the number of input paragraphs.

Feed-forward Networks We next update token representations with contextual information. As show in the top part of Figure 6.2, we first fuse information from all heads by concatenating all context vectors and applying a linear transformation with weight $\mathbf{W}_c \in \mathbb{R}^{d*d}$:

$$\mathbf{c}_i = \mathbf{W}_c[\mathbf{context}_i^1; \cdots; \mathbf{context}_i^{n_{head}}] \tag{6.21}$$

We then add $\mathbf{h}_i$ to each input token vector $\mathbf{h}_{ij}^{l-1}$, and feed it to a two-layer feed-forward network with ReLU as the activation function and a highway layer normalization on top:

$$\mathbf{g}_{ij} = \mathbf{W}_{o2}\text{ReLU}(\mathbf{W}_{o1}(\mathbf{h}_{ij}^{l-1} + \mathbf{c}_i)) \tag{6.22}$$

$$\mathbf{h}_{ij}^{l} = \text{LayerNorm}(\mathbf{g}_{ij} + \mathbf{h}_{ij}^{l-1}) \tag{6.23}$$

where $\mathbf{W}_{o1} \in \mathbb{R}^{d_{ff}*d}$ and $\mathbf{W}_{o2} \in \mathbb{R}^{d*d_{ff}}$ are the weights; d_{ff} is the hidden size of the feed-forward. This way, each token within paragraph R_i can collect information from other paragraphs in a hierarchical and efficient manner.

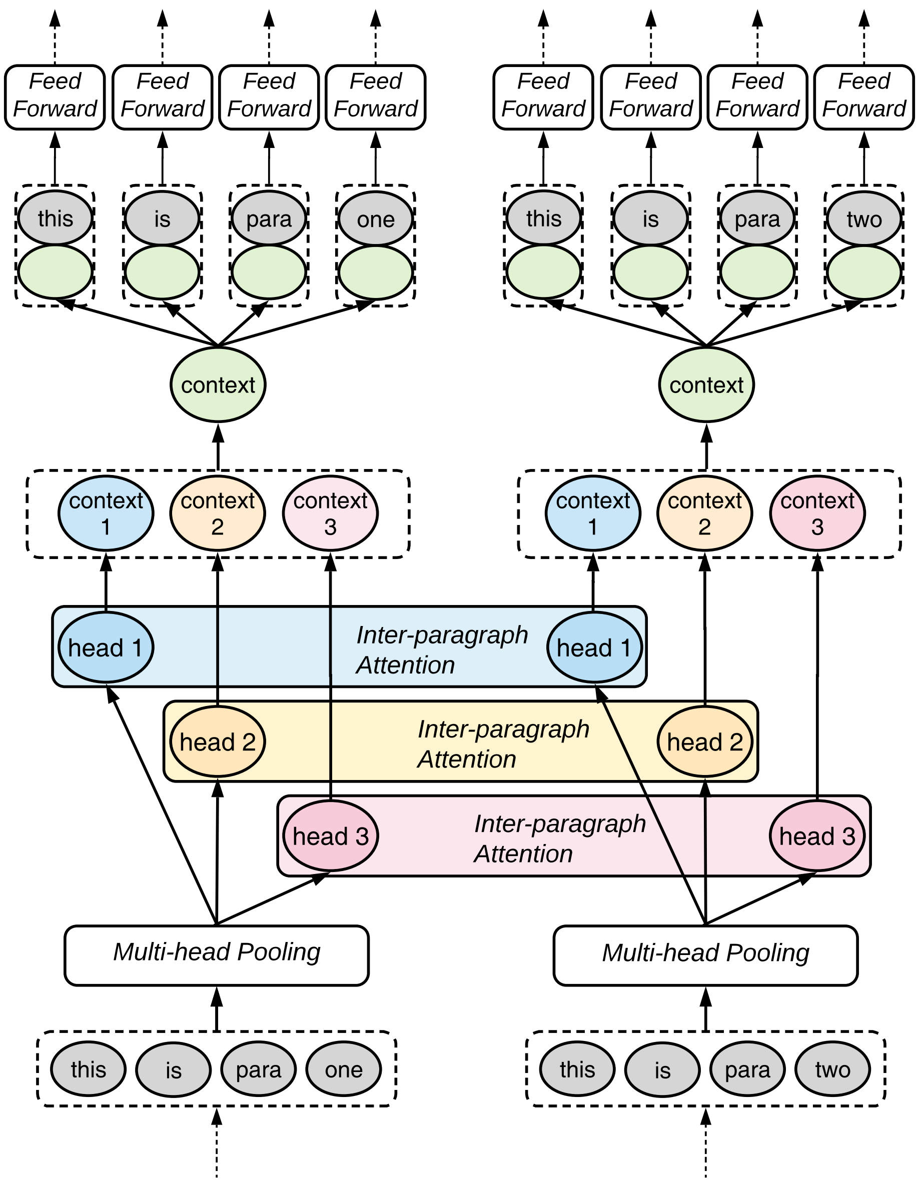


Figure 6.2: A global Transformer layer. Different colors indicate different heads in multi-head pooling and inter-paragraph attention. The grey circles at the bottom indicate the input vectors of this Transformer layer. Then a multi-head pooling operation is applied over these vectors to generate representations of the paragraph. Inter-paragraph attention is used to exchange information across different paragraphs, generating a context vector for each paragraph. At the top part of the model, these context vectors are duplicated and concatenated to input vectors of each token. They are fed into a feed-forward layer to generate the output vectors of this global Transformer layer.

6.3.2.4 Graph-informed Attention

The inter-paragraph attention mechanism can be viewed as learning a latent graph representation (self-attention weights) of the input paragraphs. Although previous work has shown that similar latent representations are beneficial for down-stream NLP tasks (Liu and Lapata, 2018; Kim et al., 2017; Williams et al., 2018; Niculae et al., 2018; Fernandes et al., 2019), much work in multi-document summarization has taken advantage of explicit graph representations, each focusing on different facets of the summarization task (e.g., capturing redundant information or representing passages referring to the same event or entity). One advantage of the hierarchical Transformer is that we can easily incorporate graphs external to the model, to generate better summaries.

We experimented with two well-established graph representations which we discuss briefly below. However, there is nothing inherent in our model that restricts us to these, any graph modeling relationships across paragraphs could have been used instead.

Our first graph aims to capture lexical relations; graph nodes correspond to paragraphs and edge weights are cosine similarities based on tf-idf representations of the paragraphs. Formally, we first represent each paragraph P_i as a bag of words. Then, we calculate the tf-idf value v_{ik} for each token x_{ik} in a paragraph:

$$v_{ik} = N_w(x_{ik}) log(\frac{N_d}{N_{dw}(x_{ik})}) \tag{6.24}$$

where $N_w(t)$ is the count of word t in the paragraph, N_d is the total number of paragraphs, $N_{dw}(t)$ is the total number of paragraphs containing the word.

As a result, we obtain a tf-idf vector for each paragraph. Then, for all paragraph pairs $< P_i, P_{i'} >$, we calculate the cosine similarity of their tf-idf vectors and use this as the weight for the edge connecting the pair in the graph. We filter edges with weights lower than 0.2.

Our second graph aims to capture discourse relations (Christensen et al., 2013); it builds an Approximate Discourse Graph[3] (ADG; Yasunaga et al. 2017) over paragraphs; edges between paragraphs are drawn by counting (a) co-occurring entities and (b) discourse markers (e.g., *however*, *nevertheless*) connecting two adjacent paragraphs.

[3]An Approximate Discourse Graph is a graph structure on documents proposed by Christensen et al. (2013) for generating coherent multi-document summaries. Unlike dicourse theories, ADG does not align edges to exact discourse relations, but it seeks to represent the pairs of sentences that have a relationship in an approximated manner by textual features from the discourse literature and other co-occurrent features.

For each paragraph P_i, we extract a set of entities in the paragraph using the Spacy[4] NER recognizer. We only use entities with type {`PERSON, NORP, FAC, ORG, GPE, LOC, EVENT, WORK_OF_ART, LAW`}. For each paragraph pair $< P_i, P_{i'} >$, we count $e_{ii'}$, the number of entities with exact match.

Meanwhile, we also use the following 36 explicit discourse markers (Christensen et al., 2013) to identify edges between two adjacent paragraphs in a source webpage:

> again, also, another, comparatively, furthermore, at the same time, however, immediately, indeed, instead, to be sure, likewise, meanwhile, moreover, nevertheless, nonetheless, notably, otherwise, regardless, similarly, unlike, in addition, even, in turn, in exchange, in this case, in any event, finally, later, as well, especially, as a result, example, in fact, then, the day before

That is, if two paragraphs $< P_i, P_{i'} >$ are adjacent in one source webpage and they are connected by one of the 36 discourse markers, $m_{ii'}$ will be 1, otherwise it will be 0. The final edge weight $D_{ii'}$ is the weighted sum of $e_{ii'}$ and $m_{ii'}$

$$D_{ii'} = 0.2 * e_{ii'} + m_{ii'} \tag{6.25}$$

We represent both similarity and discourse graphs them with a matrix G, where $G_{ii'}$ is the weight of edge connecting paragraphs i and i'. We can then inject this graph into our hierarchical Transformer by simply substituting one of its (learned) heads z' with G. Equation 6.20 for calculating the context vector for this head is modified as:

$$\mathbf{context}_i^{z'} = \sum_{i'=1}^{m} \frac{\mathbf{G}_{ii'}}{\sum_{o=1}^{m} \mathbf{G}_{io}} \mathbf{v}_{i'}^{z'} \tag{6.26}$$

where the difference against Equation 6.20 is that the attention matrix is replaced with the normalized graph matrix.

6.4 Experimental Setup

6.4.1 WikiSum Dataset

We used the scripts and urls provided in Liu et al. (2018) to crawl Wikipedia articles and source reference documents. We successfully crawled 78.9% of the original documents (some urls have become invalid and corresponding documents could not be retrieved). We further removed clone paragraphs (which are exact copies of some

[4]https://spacy.io/api/entityrecognizer

Methods	ROUGE-L Recall			
	$L' = 5$	$L' = 10$	$L' = 20$	$L' = 40$
Similarity	24.86	32.43	40.87	49.49
Ranking	39.38	46.74	53.84	60.42

Table 6.1: ROUGE-L recall against target summary for L'-best paragraphs obtained with tf-idf cosine similarity and our ranking model.

parts of the Wikipedia articles); these were paragraphs in the source documents whose bigram recall against the target summary was higher than 0.8. On average, each input has 525 paragraphs, and each paragraph has 70.1 tokens. The average length of the target summary is 139.4 tokens. We split the dataset with $1,579,360$ instances for training, $38,144$ for validation and $38,205$ for test.

For both ranking and summarization stages, we encode source paragraphs and target summaries using subword tokenization with SentencePiece (Kudo and Richardson, 2018). Our vocabulary consists of $32,000$ subwords and is shared for both source and target.

6.4.2 Paragraph Ranking

To train the regression model, we calculated ROUGE-2 recall (Lin, 2004) for each paragraph against the target summary and used this as the ground-truth score. The hidden size of the two LSTMs was set to 256, and dropout (with dropout probability of 0.2) was used before all linear layers. Adagrad (Duchi et al., 2011) with learning rate 0.15 is used for optimization. We compared our ranking model against the method proposed in Liu et al. (2018) who use the tf-idf cosine similarity between each paragraph and the article title to rank the input paragraphs. We take the first L' paragraphs from the ordered paragraph set produced by our ranker and the similarity-based method, respectively. We concatenate these paragraphs and calculate their ROUGE-L recall against the gold target text. The results are shown in Table 6.1. We can see that our ranker effectively extracts related paragraphs and produces more informative input for the downstream summarization task.

6.4.3 Training Configuration

In all abstractive models, we apply dropout (with probability of 0.1) before all linear layers; label smoothing (Szegedy et al., 2016) with smoothing factor 0.1 is also used. The training method is in traditional sequence-to-sequence manner with maximum likelihood estimation. The optimizer was Adam (Kingma and Ba, 2014) with learning rate of 2, $\beta_1 = 0.9$, and $\beta_2 = 0.998$; we also applied learning rate warmup over the first $8,000$ steps, and decay as in Vaswani et al. (2017). All Transformer-based models had 256 hidden units; the feed-forward hidden size was $1,024$ for all layers. All models were trained on 4 GPUs (NVIDIA TITAN Xp) for $500,000$ steps. We used gradient accumulation to keep training time for all models approximately consistent, and training the Hierarchical Transformer takes 5 days. We selected the 5 best checkpoints based on performance on the validation set and report averaged results on the test set. During decoding we used beam search (size 5), and applied length penalty (Wu et al., 2016) with $\alpha = 0.4$ (see section 5.4.2 for details on length penalty).

6.4.4 Comparison Systems

We compared the proposed hierarchical Transformer against several strong baselines:

LEAD is a simple baseline that concatenates the title and ranked paragraphs, and extracts the first *k* tokens; we set *k* to the length of the ground-truth target.

LexRank (Erkan and Radev, 2004) is a widely-used graph-based extractive summarizer; we build a graph with paragraphs as nodes and edges weighted by tf-idf cosine similarity; we run a PageRank-like algorithm on this graph to rank and select paragraphs until the length of the ground-truth summary is reached.

Flat Transformer (FT) is a baseline that applies a Transformer-based encoder-decoder model to a flat token sequence. The input paragraphs are ordered by their ranking scores and then concatenated with a special separate token. We used a 6-layer Transformer. The title and ranked paragraphs were concatenated and truncated to $600, 800$, and $1,200$ tokens.

T-DMCA is the best performing model of Liu et al. (2018) and a shorthand for Transformer Decoder with Memory Compressed Attention; they only used a Transformer decoder and compressed the key and value in self-attention with a convolutional layer. The model has 5 layers as in Liu et al. (2018). Its hidden size is

512 and its feed-forward hidden size is $2,048$. The title and ranked paragraphs were concatenated and truncated to 3,000 tokens.

Hierarchical Transformer (HT) is the model proposed in this paper. The model architecture is a 7-layer network (with 5 local-attention layers at the bottom and 2 global attention layers at the top). The model takes the title and $L' = 24$ paragraphs as input to produce a target summary, which leads to approximately $1,600$ input tokens per instance.

6.5 Results

6.5.1 Automatic Evaluation

We evaluated summarization quality using ROUGE F_1 (Lin, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

Table 6.2 summarizes our results. The first block in the table includes extractive systems (LEAD, LexRank), the second block includes several variants of Flat Transformer-based models (FT, T-DMCA), while the rest of the table presents the results of our Hierarchical Transformer (HT). As can be seen, abstractive models generally outperform extractive ones. The Flat Transformer, achieves best results when the input length is set to 800 tokens, while longer input (i.e., $1,200$ tokens) actually hurts performance. The Hierarchical Transformer with $1,600$ input tokens, outperforms FT, and even T-DMCA when the latter is presented with $3,000$ tokens. Adding an external graph also seems to help the summarization process. The similarity graph does not have an obvious influence on the results, while the discourse graph boosts ROUGE-L by 0.16. This could be explained by the fact that the inter-paragraph attention can already capture the similarity to some degree by multi-head self-attention.

We also found that the performance of the Hierarchical Transformer further improves when the model is presented with longer input at test time.[5] As shown in the last row of Table 6.2, when testing on $3,000$ input tokens, summarization quality improves across the board. This suggests that the model can potentially generate better summaries without increasing training time.

[5]This was not the case with the other Transformer models.

Model	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	38.22	16.85	26.89
LexRank	36.12	11.67	22.52
FT (600 tokens, no ranking)	35.46	20.26	30.65
FT (600 tokens)	40.46	25.26	34.65
FT (800 tokens)	40.56	25.35	34.73
FT (1,200 tokens)	39.55	24.63	33.99
T-DMCA (3000 tokens)	40.77	25.60	34.90
HT (1,600 tokens)	40.82	25.99	35.08
HT (1,600 tokens) + Similarity Graph	40.80	25.95	35.08
HT (1,600 tokens) + Discourse Graph	40.81	25.95	35.24
HT (train on 1,600 tokens/test on 3000 tokens)	**41.53**	**26.52**	**35.76**

Table 6.2: Test set results on the WikiSum dataset using ROUGE F_1. The bold cells indicate best results.

Model	R1	R2	RL
HT	40.82	25.99	35.08
HT w/o PP	40.21	24.54	34.71
HT w/o MP	39.90	24.34	34.61
HT w/o GT	39.01	22.97	33.76

Table 6.3: Hierarchical Transformer and versions thereof without (w/o) paragraph position (PP), multi-head pooling (MP), and global Transformer layer (GT).

Table 6.3 summarizes ablation studies aiming to assess the contribution of individual components. Our experiments confirm that encoding paragraph position in addition to token position within each paragraph is beneficial (see row w/o PP), as well as multi-head pooling (w/o MP is a model where the number of heads is set to 1), and the global Transformer layer (w/o GT is a model with only 5 local Transformer layers in the encoder).

6.5.2 Human Evaluation

In addition to automatic evaluation, we also assessed system performance by eliciting human judgments on 20 randomly selected test instances. Our first evaluation study

Model	QA	Rating
LEAD	31.59	-0.383
FT	35.69	0.000
T-DMCA	43.14	0.147
HT	**54.11**	**0.237**

Table 6.4: System scores based on questions answered by AMT participants and summary quality rating.

quantified the degree to which summarization models retain key information from the documents following a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018b). As described in Section 5, we created a set of questions based on the gold summary under the assumption that it contains the most important information from the input paragraphs. We then examined whether participants were able to answer these questions by reading system summaries alone without access to the gold summary. The more questions a system can answer, the better it is at summarization. We created 57 questions in total varying from two to four questions per gold summary. Examples of questions and their answers are given in Table 6.5. We adopted the same scoring mechanism used in Clarke and Lapata (2010), i.e., correct answers are marked with 1, partially correct ones with 0.5, and 0 otherwise. A system's score is the average of all question scores.

Our second evaluation study assessed the overall quality of the summaries by asking participants to rank them taking into account the following criteria: *Informativeness* (does the summary convey important facts about the topic in question?), *Fluency* (is the summary fluent and grammatical?), and *Succinctness* (does the summary avoid repetition?). We used Best-Worst Scaling (Louviere et al., 2015), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2017). Participants were presented with the gold summary and summaries generated from 3 out of 4 systems and were asked to decide which summary was the best and which one was the worst in relation to the gold standard, taking into account the criteria mentioned above. The rating of each system was computed as the percentage of times it was chosen as best minus the times it was selected as worst. Ratings range from -1 (worst) to 1 (best).

Both evaluations were conducted on the Amazon Mechanical Turk platform with 5 responses per hit. Detailed instructions of human evaluation can be found in Appendix A. Participants evaluated summaries produced by the LEAD baseline, the Flat Transformer, T-DMCA, and our Hierarchical Transformer. All evaluated systems were variants that achieved the best performance in automatic evaluations. As shown in Table 6.4, on both evaluations, participants overwhelmingly prefer our model (HT). All pairwise comparisons among systems are statistically significant (using a one-way ANOVA with post-hoc Tukey HSD tests; $p < 0.01$).

6.5.3 Examples of System Output and Evaluation Questions

Table 6.5 shows examples of system output on the WikiSum dataset. Specifically, we show summaries produced from the LEAD baseline, the Flat Transformer (FT), T-DMCA (Liu et al., 2018), and our Hierarchical Transformer (HT). The table also contains examples of questions (and their answers) used in our QA-based evaluation study. We can observe that, unlike single-document summarization, here in the multi-document setting, the LEAD baseline performs badly, with much redundant information. The Flat Transformer produces less precise summaries with some incorrect facts. For example, in Table 6.5, Flat Transformer produces 'It was listed on the national register of historic places in 1983', which actually should be in 1993. The outputs of T-DMCA and Hierarchical Transformer are similar, but our model produces more informative summaries. As shown in Table 6.5, the summary of our model covers more about 'Pentagoet Archeological District' on the colonial period.

6.6 Summary

In this chapter we extended our study to the field of multi-document summarization by conceptualizing it as a machine learning problem. We proposed a new model which is able to encode multiple input documents hierarchically, learn latent relations across them, and additionally incorporate structural information from well-known graph representations. We have also demonstrated the importance of a learning-based approach for selecting which documents to summarize. Experimental results show that our model produces summaries which are both fluent and informative outperforming competitive systems by a wide margin.

	Pentagoet Archeological District
GOLD	The Pentagoet Archeological District is a National Historic Landmark District located at the southern edge of the Bagaduce Peninsula in Castine, Maine. It is the site of Fort Pentagoet, a 17th-century fortified trading post established by fur traders of French Acadia. From 1635 to 1654 this site was a center of trade with the local Abenaki, and marked the effective western border of Acadia with New England. From 1654 to 1670 the site was under English control, after which it was returned to France by the Treaty of Breda. The fort was destroyed in 1674 by Dutch raiders. The site was designated a National Historic Landmark in 1993. It is now a public park.
QA	What is the Pentagoet Archeological District? [a National Historic Landmark District] Where is it located? [Castine, Maine] What did the Abenaki Indians use the site for? [trading center]
LEAD	The Pentagoet Archeological District is a National Historic Landmark District located in Castine, Maine. This district forms part of the traditional homeland of the Abenaki Indians, in particular the Penobscot tribe. In the colonial period, Abenakis frequented the fortified trading post at this site, bartering moosehides, sealskins, beaver and other furs in exchange for European commodities. "Pentagoet Archeological district" is a National Historic Landmark District located at the southern edge of the Bagaduce Peninsula in Treaty Of Breda.
FT	the Pentagoet Archeological district is a National Historic Landmark District located at the southern edge of the Bagaduce Peninsula in Treaty Of Breda. It was listed on the national register of historic places in 1983.
T-DMCA	The Pentagoet Archeological District is a national historic landmark district located in Castine, Maine. This district forms part of the traditional homeland of the Abenaki Indians, in particular the Penobscot tribe. The district was listed on the national register of historic places in 1982.
HT	The Pentagoet Archeological district is a National Historic Landmark District located in Castine, Maine. This district forms part of the traditional homeland of the Abenaki Indians, in particular the Penobscot tribe. In the colonial period, Abenaki frequented the fortified trading post at this site, bartering moosehides, sealskins, beaver and other furs in exchange for European commodities.

Table 6.5: GOLD human authored summaries, questions based on them (answers shown in square brackets) and automatic summaries produced by the LEAD baseline, the Flat Transformer (FT), T-DMCA (Liu et al., 2018) and our Hierachical Transformer (HT).

Melanesian Whistler

GOLD	The Melanesian whistler or Vanuatu whistler (Pachycephala chlorura) is a species of passerine bird in the whistler family Pachycephalidae. It is found on the Loyalty Islands, Vanuatu, and Vanikoro in the far south-eastern Solomons.
QA	What is the Melanesian Whistler? [a species of passerine bird in the whistler family Pachycephalidae] Where is it found? [Loyalty Islands, Vanuatu, and Vanikoro in the far south-eastern Solomons]
LEAD	The Australian golden whistler (Pachycephala pectoralis) is a species of bird found in forest, woodland, mallee, mangrove and scrub in Australia (except the interior and most of the north) Most populations are resident, but some in south-eastern Australia migrate north during the winter.
FT	The Melanesian whistler (P. Caledonica) is a species of bird in the family Muscicapidae. It is endemic to Melanesia.
T-DMCA	The Australian golden whistler (Pachycephala chlorura) is a species of bird in the family Pachycephalidae, which is endemic to Fiji.
HT	The Melanesian whistler (Pachycephala chlorura) is a species of bird in the family Pachycephalidae, which is endemic to Fiji.

Table 6.5 Continued

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we focused on neural summarization and examined how to improve the current state of the art by focusing on three hypothesis: a) explicit modeling of document structure is beneficial for capturing important document content; b) beyond structural knowledge at the discourse-level, it is possible to incorporate general knowledge contained in a large corpus via model pretraining into summarization systems; c) despite being computationally challenging, neural models can be developed for multi-document summarization by introducing a trained document pre-ranking module and a hierarchical Transformer encoder.

For the first hypothesis, we proposed to incorporate document structure into neural networks for modeling documents in an end-to-end manner. Document modelling is a foundational module for various NLP tasks and document structure has been proven helpful to generating better document representations (Ji and Smith, 2017). Starting from this problem, in Chapter 3, we developed a structure-aware document encoder. Given an input document, the encoder encodes it with a self-attention based LSTM model. And to incorporate structural information, we constrain self-attention weights to non-projective dependency structures making use of the matrix-tree theorem (Kirchhoff, 1847). The proposed structured attention framework is flexible and powerful, and does not rely on an external parser. We performed experiments on four document classification datasets, and showed that the structured attention model achieves better results compared to previous models. In Chapter 4, we extended this framework to the task of extractive single-document summarization. We reframed summarization as a tree induction problem and used structured attention as both the attention weights and the summarization objective which serves as a regularizer when learning a summarization model. Experimental results showed that compared to previous systems our model achieves competitive performance across different news summarization datasets, confirming that the proposed structure-aware representations of documents can improve the performance of a summarization system.

Apart from structural information, knowledge in large-scale unannotated natural language corpora can also be useful for a summarization system. With the increasing usage of pretrained language models in NLP tasks, the expectation of building a pretrained summarization system is put forward. In Chapter 5, we proposed a framework for using pretrained language models as encoders for both extractive and abstractive summarization. For extractive summarization, we modified the input to the pretrained encoder to obtain sentence representations. For abstractive summarization, we separated the optimizer for the encoder and the decoder to achieve more stable training. The proposed model can be considered as a minimum-requirement system for the summarization task, and can be easily combined with other modules. Experimental results showed that our system can achieve new state-of-the-art results across multiple datasets under both extractive and abstractive settings. We found that the knowledge in pretrained language models can bring improvement for summarization systems. Also we showed it is important to choose appropriate optimization methods when using a neural model composed of both pretrained and non-pretrained modules.

Although the study of using neural networks for text summarization has made progress in recent years, most work has focused on single-document summarization. Multi-document summarization, another interesting and important subtask, has attracted much less attention. In Chapter 6, we extended our work to the field of multi-document summarization. We analyzed the challenges of directly applying existing models into this task. And we proposed several solutions against these challenges. We developed a new method for ranking paragraphs before applying a neural abstractive model. We augmented the vanilla Transformer model with the ability to process multiple input documents. In the experiments, our Hierarchical Transformer model, has shown superiority over previous systems on a large-scale multi-document summarization dataset. We found the ranking of input documents is important for multi-document summarization. Meanwhile, adding hierarchical neural network layers into the Transformer model can produce better representations of multiple input documents, and improve the performance of a multi-document summarization system.

7.2 Future Work

There are several remaining challenges in the field of text summarization. Avenues for future research are many and varied. We discuss some promising directions as follows.

Noisy Data The collection of document-summary pairs to form large-scale summarization datasets has been a key factor that contributed to the recent wave of neural summarization systems. However, some of these datasets have been found to contain a certain degree of noise due to the underlying collection process (Kryscinski et al., 2019). More specifically, for some instances, the reference summary cannot be aligned to the source document, since producing the summaries requires world knowledge or external information that cannot be found in the source document. For example, in the XSum dataset we experimented with in previous chapters, the reference summary is collected by extracting the first sentence of the source article. And for some reference summaries in the dataset, the entities is the summaries are never mentioned in the rest part of the article. Meanwhile, the problem of noisy data is particularly more serious in those datasets that are collected in a loosely-aligned manner. In the WikiSum dataset where the reference summaries are the lead sections of Wikipedia articles, a large part of the reference summaries cannot be found from the input documents. How to learn from noisy data could be a future direction. Potential future work include Neural Bootstrapping (Bengio and LeCun, 2015) and Knowledge Distillation (Hinton et al., 2015). Neural Bootstrapping (Bengio and LeCun, 2015) attempts to use a convex combination of training labels and the predictions of current model to generate the training targets, which could avoid directly modeling the noise distribution. Knowledge Distillation (Hinton et al., 2015; Kim and Rush, 2016; Li et al., 2017) applies a teacher-student training process, where the teacher model is trained directly on the noisy labels and the student model is trained on the combination of the teacher predictions and the noisy labels. Since the teacher predictions could implicitly denoise the training data, the student model can achieve better generalization.

Fact-Faithful Language Generation Fact-faithfulness should be an important requirement for automatic summarization systems. It means that the produced summaries should respect the facts in the source documents and not generate untrue information or hallucinations. However, with the neural encoder-decoder architecture, it is difficult to control the faithfulness (Kryscinski et al., 2019), and this has become a major obstacle that prevents from deploying neural abstractive summarization systems in practice. For future research, one direction should be to evaluate the fact-faithfulness of generated summaries, instead of completely relying on ngram-based metrics like ROUGE. For example, by using external NLP tools like Information Retrieval and Semantic Parsing, the generated summaries and the reference summaries could be matched and compared on semantic-level. Another direction is to reduce hallucinations in summarization systems. Very recently, Tian et al. (2019) present a confidence oriented decoder which predicts an auxiliary confidence score at each decoding step. This confidence score can be used as a calibration of the final decoding probability for a more faithful generation. While this work is experimented on data-to-text generation task, summarization can also take advantages of modeling decoding confidence. Zhu et al. (2020) build a knowledge graph of the input document and integrate this graph during summary generation via an attention mechanism over the knowledge graph nodes. Their model is able to preserve more facts during summarization as it incorporates factual information via the knowledge graph.

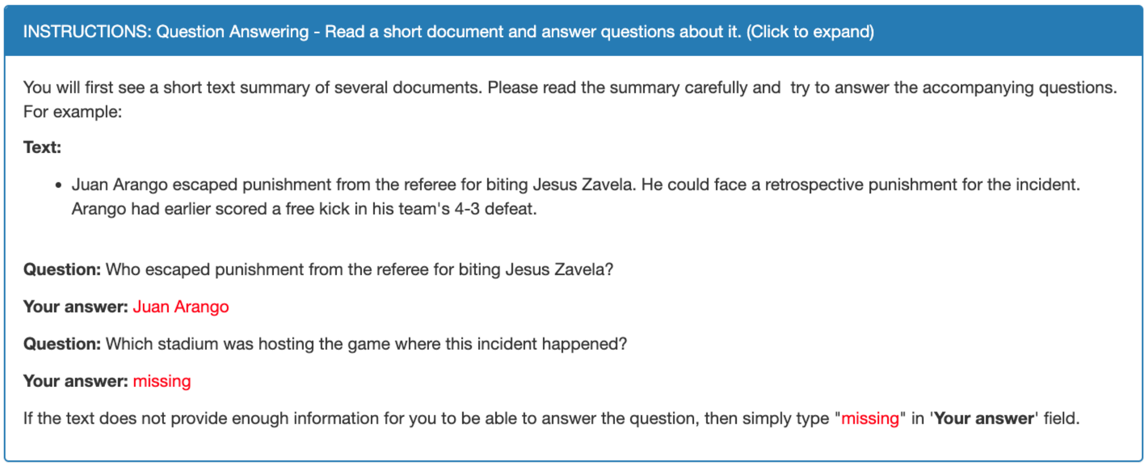
Long Document Summarization Due to the limitation of available training corpora, most work on text summarization has focused on news summarization, where the length of the input documents is usually less than 1,000 tokens. However, summarization could also be useful for long input texts, like Wikipedia articles, patent files (Sharma et al., 2019) or book chapters. How to adapt existing neural summarization models to process long input documents is a promising future research direction. Existing models face several challenges when processing long documents, including computational limitation and the dependency of tokens. These dependencies can be very distant in long documents, and it is difficult for current systems to capture these dependencies (Dai et al., 2019). Very recently, there has been work focusing on augmenting recurrent neural networks or Transformer models with the ability to model long documents. Zhang et al. (2018a) employ an average attention mechanism to replace self-attention in the Transformer model. With a cumulative average operation over history representations, the model accelerates the speed of Transformer models when modeling long documents. Dai et al. (2019) augment the Transformer model with a segment-level recurrence mechanism and a novel positional encoding scheme, achieving faster and better performance on long text language modeling. Sukhbaatar et al. (2019) present a novel self-attention mechanism with an adaptive span, where at each time step, the model can dynamically decide the attention span. The model is found to have the capability to catch longer dependencies than vanilla Transformer models. These models can be adopted as encoders for the summarization task for a better encoding of long input documents.

Appendix A

Instruction for Human Evaluation

A.1 Question Answering Human Evaluation

Question answering human evaluation is used in Chapter 4, Chapter 5 and Chapter 6 to evaluate a summarization system. Figure A.1 shows the instructions we give to the evaluation participants on the Amazon Mechanical Turk platform. Each participant is asked to first read a system generated summary of a article carefully and the answer several questions based this summary. The questions are complied based on the reference summaries of this article.


INSTRUCTIONS: Question Answering - Read a short document and answer questions about it. (Click to expand)

You will first see a short text summary of several documents. Please read the summary carefully and try to answer the accompanying questions. For example:

Text:

- Juan Arango escaped punishment from the referee for biting Jesus Zavela. He could face a retrospective punishment for the incident. Arango had earlier scored a free kick in his team's 4-3 defeat.

Question: Who escaped punishment from the referee for biting Jesus Zavela?

Your answer: Juan Arango

Question: Which stadium was hosting the game where this incident happened?

Your answer: missing

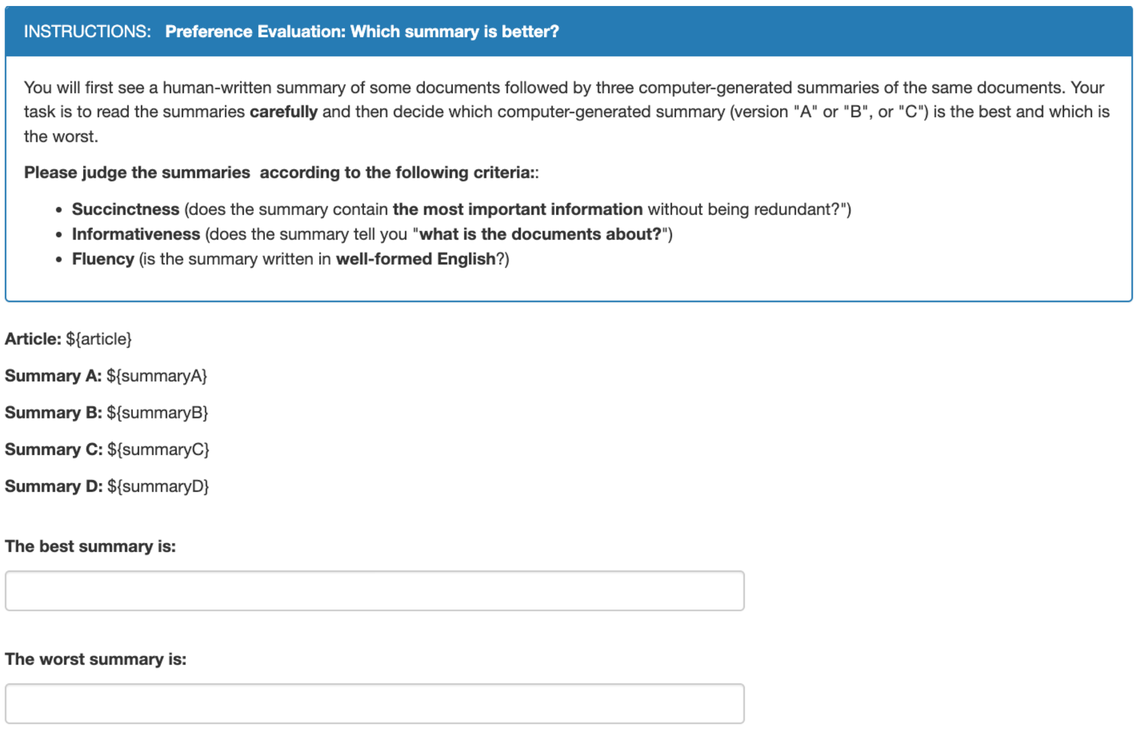
If the text does not provide enough information for you to be able to answer the question, then simply type "missing" in '**Your answer**' field.

Figure A.1: Instructions for question answering human evaluation of summarization systems on the webpage of Amazon Mechanical Turk platform.

A.2 Quality Ranking Human Evaluation

Quality ranking human evaluation is used in Chapter 4, Chapter 5 and Chapter 6 to evaluate a summarization system. Figure A.2 shows the instructions we give to the evaluation participants on the Amazon Mechanical Turk platform. Each participant is asked to first read a article carefully and then read several summaries generated by different models. The participant is asked to select the best and the worst summary. To help participants judge the quality, we provide three criteria:

1. Succinctness: does the summary contain the most important information without being redundant?
2. Informativeness: does the summary tell you “what is the documents about”?
3. Fluency: is the summary written in well-formed English?



INSTRUCTIONS: **Preference Evaluation: Which summary is better?**

You will first see a human-written summary of some documents followed by three computer-generated summaries of the same documents. Your task is to read the summaries **carefully** and then decide which computer-generated summary (version "A" or "B", or "C") is the best and which is the worst.

Please judge the summaries according to the following criteria::

- **Succinctness** (does the summary contain **the most important information** without being redundant?")
- **Informativeness** (does the summary tell you "**what is the documents about?**")
- **Fluency** (is the summary written in **well-formed English**?)

Article: ${article}

Summary A: ${summaryA}

Summary B: ${summaryB}

Summary C: ${summaryC}

Summary D: ${summaryD}

The best summary is:

The worst summary is:

Figure A.2: Instructions for quality ranking human evaluation of summarization systems on the webpage of Amazon Mechanical Turk platform.

Bibliography

Abuobieda, A., Salim, N., Albaham, A. T., Osman, A. H., and Kumar, Y. J. (2012). Text summarization features selection method using pseudo genetic-based model. In *2012 International Conference on Information Retrieval & Knowledge Management*, pages 193–197.

Aone, C., Okurowski, M. E., Gorlinsky, J., and Larsen, B. (1997). A scalable summarization system using robust nlp. *Intelligent Scalable Text Summarization*.

Azzam, S., Humphreys, K., and Gaizauskas, R. (1999). Using coreference chains for text summarization. In *Proceedings of the Workshop on Coreference and its Applications*, pages 77–84.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.

Baker, J. K. (1979). Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Barrera, A. and Verma, R. (2012). Combining syntax and semantics for automatic extractive single-document summarization. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 366–377. Springer.

Barzilay, R. and Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Barzilay, R. and McKeown, K. R. (2005). Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–327.

Baxendale, P. B. (1958). Machine-made index for technical literaturean experiment. *IBM Journal of research and development*, 2(4):354–361.

Belanger, D., Yang, B., and McCallum, A. (2017). End-to-end learning for structured prediction energy networks. *ICML*.

Bengio, Y. and LeCun, Y. (2015). Training deep neural networks on noisy labels with bootstrapping. In *Proceedings of the 3rd International Conference on Learning Representations, Workshop Track Proceedings*, San Diego, California.

Bhatia, P., Ji, Y., and Eisenstein, J. (2015). Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218, Lisbon, Portugal.

Bing, L., Li, P., Liao, Y., Lam, W., Guo, W., and Passonneau, R. (2015). Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1587–1597, Beijing, China.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Bowman, S. R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C. D., and Potts, C. (2016). A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany.

Brychcın, T. and Habernal, I. (2013). Unsupervised improving of sentiment analysis using global target context. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 122–128.

Carbonell, J. G. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, Melbourne, Australia.

Carenini, G., Ng, R. T., and Zhou, X. (2008). Summarizing emails with conversational cohesion and subjectivity. In *Proceedings of ACL-08: HLT*, pages 353–361, Columbus, Ohio.

Carlson, L., Marcu, D., and Okurowski, M. E. (2001). Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana.

Chen, Q., Zhu, X., Ling, Z., Wei, S., and Jiang, H. (2016). Distraction-based neural networks for modeling documents. In *Proceedings of the IJCAI Conference*, pages 2754–2760.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas.

Cheng, J. and Lapata, M. (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)*, pages 484–494, Berlin, Germany.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). 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)*, pages 1724–1734, Doha, Qatar.

Christensen, J., Mausam, Soderland, S., and Etzioni, O. (2013). Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1173, Atlanta, Georgia. Association for Computational Linguistics.

Chu, E. and Liu, P. J. (2019). Meansum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1223–1232, Long Beach, California, USA.

Chu, Y.-J. and Liu, T.-H. (1965). On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Clarke, J. and Lapata, M. (2010). Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Corro, C. and Titov, I. (2019). Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *ICLR*.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy.

Daniluk, M., Rocktäschel, T., Welbl, J., and Riedel, S. (2017). Frustratingly short attention spans in neural language modeling. *Proceedings of the ICLR Conference*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Diao, Q., Qiu, M., Wu, C.-Y., Smola, A. J., Jiang, J., and Wang, C. (2014). Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the ACM SIGKDD Conference*, pages 193–202.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H. (2019). Unified language model pre-training for natural language understanding and generation. In *Proceedings of the 2019 Annual Conference on Neural Information Processing Systems*, pages 13042–13054, Vancouver, BC, Canada.

Dong, Y., Shen, Y., Crawford, E., van Hoof, H., and Cheung, J. C. K. (2018). BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Durrett, G., Berg-Kirkpatrick, T., and Klein, D. (2016). Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Edunov, S., Baevski, A., and Auli, M. (2019). Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Fabbri, A., Li, I., She, T., Li, S., and Radev, D. (2019). Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy.

Fattah, M. A. and Ren, F. (2009). Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Computer Speech & Language*, 23(1):126–144.

Feldman, R. and Sanger, J. (2007). *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press.

Feng, V. W. and Hirst, G. (2012a). Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 60–68, Jeju Island, Korea.

Feng, V. W. and Hirst, G. (2012b). Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 60–68, Jeju Island, Korea.

Feng, V. W. and Hirst, G. (2014). A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 511–521.

Fernandes, P., Allamanis, M., and Brockschmidt, M. (2019). Structured neural summarization. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, Louisiana.

Ferreira, R., de Souza Cabral, L., Lins, R. D., e Silva, G. P., Freitas, F., Cavalcanti, G. D., Lima, R., Simske, S. J., and Favaro, L. (2013). Assessing sentence scoring techniques for extractive text summarization. *Expert systems with applications*, 40(14):5755–5764.

Filippova, K. and Strube, M. (2008). Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 177–185, Honolulu, Hawaii.

Gehrmann, S., Deng, Y., and Rush, A. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium.

Gerani, S., Mehdad, Y., Carenini, G., Ng, R. T., and Nejat, B. (2014). Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602–1613, Doha, Qatar.

Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. (2000). Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

Graves, A., Mohamed, A.-R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE ICASSP Conference*, pages 6645–6649.

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

Guinaudeau, C. and Strube, M. (2013). Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 93–103, Sofia, Bulgaria.

Hayashi, K., Hirao, T., and Nagata, M. (2016). Empirical comparison of dependency conversions for rst discourse trees. In *Proceedings of the Annual Meeting of SIGDIAL*, page 128.

He, R., Lee, W. S., Ng, H. T., and Dahlmeier, D. (2018). Exploiting document knowledge for aspect-level sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–585, Melbourne, Australia.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hirao, T., Yoshida, Y., Nishino, M., Yasuda, N., and Nagata, M. (2013a). Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, Washington, USA.

Hirao, T., Yoshida, Y., Nishino, M., Yasuda, N., and Nagata, M. (2013b). Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, Washington, USA.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ji, Y. and Eisenstein, J. (2014). Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland.

Ji, Y. and Smith, N. A. (2017). Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 996–1005, Vancouver, Canada.

Jiang, J., Zhang, M., Li, C., Bendersky, M., Golbandi, N., and Najork, M. (2019). Semantic text matching for long-form documents. In *Proceedings of the 2019 World Wide Web Conference*, pages 795–806, San Francisco, CA, USA.

Jones, K. S. et al. (1999). Automatic summarizing: factors and directions. *Advances in automatic text summarization*, pages 1–12.

Joty, S., Carenini, G., and Ng, R. T. (2015). Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Kågebäck, M., Mogren, O., Tahmasebi, N., and Dubhashi, D. (2014). Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39.

Kim, Y., Denton, C., Hoang, L., and Rush, A. M. (2017). Structured attention networks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2017a). Semi-supervised classification with graph convolutional networks. *ICLR*.

Kipf, T. N. and Welling, M. (2017b). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.

Kirchhoff, G. (1847). Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508.

Kiritchenko, S. and Mohammad, S. (2017). Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 465–470, Vancouver, Canada.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada.

Koo, T., Globerson, A., Carreras, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic.

Kryscinski, W., Keskar, N. S., McCann, B., Xiong, C., and Socher, R. (2019). Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Kupiec, J., Pedersen, J., and Chen, F. (1999). A trainable document summarizer. *Advances in Automatic Summarization*, pages 55–60.

Lebanoff, L. and Liu, F. (2018). Automatic detection of vague words and sentences in privacy policies. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3508–3517, Brussels, Belgium.

Lebanoff, L., Song, K., Dernoncourt, F., Kim, D. S., Kim, S., Chang, W., and Liu, F. (2019). Scoring sentence singletons and pairs for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, Florence, Italy.

Lebanoff, L., Song, K., and Liu, F. (2018). Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141, Brussels, Belgium.

Lee, A., Prasad, R., Joshi, A., Dinesh, N., and Webber, B. (2006). Complexity of dependencies in discourse: Are dependencies in discourse more complex than in syntax. In *Proceedings of the International Workshop on Treebanks and Linguistic Theories*, page 12.

Lee, J., Mansimov, E., and Cho, K. (2018). Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium.

Li, J., Li, R., and Hovy, E. (2014a). Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069, Doha, Qatar.

Li, S., Wang, L., Cao, Z., and Li, W. (2014b). Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35, Baltimore, Maryland.

Li, W., Xiao, X., Lyu, Y., and Wang, Y. (2018). Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium.

Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, L. (2017). Learning from noisy labels with distillation. In *Proceedings of the 2017 IEEE International Conference on Computer Vision*, pages 1928–1936, Venice, Italy.

Lin, C.-Y. (1999). Training a selection function for extraction. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 55–62. ACM.

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

Lin, C.-Y. and Hovy, E. (1997). Identifying topics by position. In *Fifth Conference on Applied Natural Language Processing*, pages 283–290.

Lin, X., Joty, S., Jwalapuram, P., and Bari, M. S. (2019). A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy.

Lin, Z., Ng, H. T., and Kan, M.-Y. (2011). Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 997–1006, Portland, Oregon, USA.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating Wikipedia by summarizing long sequences. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada.

Liu, X., Webster, J. J., and Kit, C. (2009). An extractive text summarizer based on significant words. In *International Conference on Computer Processing of Oriental Languages*, pages 168–178. Springer.

Liu, Y. and Lapata, M. (2017). Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1298, Copenhagen, Denmark.

Liu, Y. and Lapata, M. (2018). Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Liu, Y., Titov, I., and Lapata, M. (2019). Single document summarization as tree induction. In *Proceedings of the 2019 Conference of the North American Chapter* *of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1745–1755, Minneapolis, Minnesota.

Louviere, J. J., Flynn, T. N., and Marley, A. A. J. (2015). *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Ma, S., Deng, Z.-H., and Yang, Y. (2016). An unsupervised multi-document summarization framework based on neural document model. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1514–1523, Osaka, Japan.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Marcheggiani, D. and Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1507–1516, Copenhagen, Denmark.

Marcu, D. (1999). Discourse trees are good indicators of importance in text. *Advances in automatic text summarization*, 293:123–136.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Marino, J., Yue, Y., and Mandt, S. (2018). Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3403–3412, Stockholm, Sweden.

Maruf, S. and Haffari, G. (2018). Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284, Melbourne, Australia.

McDonald, R. and Satta, G. (2007). On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 121–132.

Melčuk, I. A. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.

Mesgar, M. and Strube, M. (2015). Graph-based coherence modeling for assessing readability. In *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, pages 309–318.

Meyer, T. and Webber, B. (2013). Implicitation of discourse connectives in (machine) translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 19–26.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the NIPS Conference*, pages 3111–3119.

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas.

Mim, F. S., Inoue, N., Reisert, P., Ouchi, H., and Inui, K. (2019). Unsupervised learning of discourse-aware text representation for essay scoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 378–385, Florence, Italy.

Morey, M., Muller, P., and Asher, N. (2017). How much progress have we made on RST discourse parsing? a replication study of recent results on the RST-DT. In

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1319–1324, Copenhagen, Denmark.

Nallapati, R., Zhai, F., and Zhou, B. (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*, page 30753081.

Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Narayan, S., Cohen, S. B., and Lapata, M. (2018a). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium.

Narayan, S., Cohen, S. B., and Lapata, M. (2018b). Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana.

Nenkova, A. (2005). Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, page 14361441.

Nenkova, A. and McKeown, K. (2012). A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer.

Nenkova, A. and Vanderwende, L. (2005). The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101.

Niculae, V., Martins, A. F. T., and Cardie, C. (2018). Towards dynamic computation graphs via sparse latent structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 905–911, Brussels, Belgium.

Osborne, M. (2002). Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 1–8, Phildadelphia, Pennsylvania, USA.

Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas.

Parveen, D., Ramsl, H.-M., and Strube, M. (2015). Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954, Lisbon, Portugal.

Parveen, D. and Strube, M. (2014). Multi-document summarization using bipartite graphs. In *Proceedings of TextGraphs-9: the workshop on Graph-based Methods for Natural Language Processing*, pages 15–24, Doha, Qatar.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, BC, Canada.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A. K., and Webber, B. L. (2008). The Penn discourse TreeBank 2.0. In *LREC*.

Putzky, P. and Welling, M. (2017). Recurrent inference machines for solving inverse problems. *CoRR*, abs/1706.04008.

Radev, D. (2000). A common theory of information fusion from multiple text sources step one: Cross-document structure. In *1st SIGdial Workshop on Discourse and Dialogue*, pages 74–83, Hong Kong, China.

Radev, D., Winkel, A., and Topper, M. (2002a). Multi document centroid-based text summarization. In *Proceedings of the ACL-02 Demonstration Session*, page 112113, Philadelphia, PA.

Radev, D. R., Hovy, E., and McKeown, K. (2002b). Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408.

Radev, D. R., Jing, H., Styś, M., and Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. In *CoRR, abs/1704.01444, 2017*.

Rothe, S., Narayan, S., and Severyn, A. (2019). Leveraging pre-trained checkpoints for sequence generation tasks. *arXiv preprint arXiv:1907.12461*.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Sandhaus, E. (2008). The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

See, A., Liu, P. J., and Manning, C. D. (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)*, pages 1073–1083, Vancouver, Canada.

Sharma, E., Li, C., and Wang, L. (2019). BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Florence, Italy.

Shi, J., Liang, C., Hou, L., Li, J., Liu, Z., and Zhang, H. (2019). Deepchannel: Salience estimation by contrastive learning for extractive document summarization. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, pages 6999–7006, Honolulu, Hawaii, USA.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA.

Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. (2019). Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335, Florence, Italy.

Svore, K., Vanderwende, L., and Burges, C. (2007). Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 448–457, Prague, Czech Republic.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 2015 Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566, Beijing, China.

Tang, D., Qin, B., and Liu, T. (2015a). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal.

Tang, D., Qin, B., and Liu, T. (2015b). Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1014–1023, Beijing, China.

Tenney, I., Das, D., and Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy.

Tesniére, L. (1959). *Éléments de Syntaxe Structurale*. Editions Klincksieck.

Thomas, M., Pang, B., and Lee, L. (2006). Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335, Sydney, Australia.

Tian, R., Narayan, S., Sellam, T., and Parikh, A. P. (2019). Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Tutte, W. T. (1984). Graph theory.

van den Bosch, A., Bogers, T., and De Kunder, M. (2016). Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, Long Beach, CA, USA.

Verberne, S., Boves, L., Oostdijk, N., and Coppen, P.-A. (2007). Discourse-based answering of why-questions. *Traitement Automatique des Language, Discours et Document: Traitements Automatics*, 47(2):21–41.

Wan, X. (2008). An exploration of document impact on graph-based multi-document summarization. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Honolulu, Hawaii.

Wan, X. and Xiao, J. (2008). Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, page 855860.

Wan, X. and Yang, J. (2008). Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, Singapore.

Wang, D., Zhu, S., Li, T., and Gong, Y. (2009). Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300, Suntec, Singapore.

White, M., Korelsky, T., Cardie, C., Ng, V., Pierce, D., and Wagstaff, K. (2001). Multidocument summarization via information extraction. In *Proceedings of the First International Conference on Human Language Technology Research*.

Williams, A., Drozdov, A., and Bowman, S. R. (2018). Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Wolf, F. and Gibson, E. (2006). *Coherence in Natural Language: Data Structures and Applications*. The MIT Press.

Woodsend, K. and Lapata, M. (2010). Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, Uppsala, Sweden.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.

Xiao, W. and Carenini, G. (2019). Extractive summarization of long documents by combining global and local context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3011–3021, Hong Kong, China.

Xie, P. and Xing, E. P. (2013). Integrating document clustering and topic modeling. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 694–703.

Xu, H., Liu, B., Shu, L., and Yu, P. (2019a). BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, Minneapolis, Minnesota.

Xu, J. and Durrett, G. (2019). Neural extractive text summarization with syntactic compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 3290–3301, Hong Kong, China.

Xu, J., Gan, Z., Cheng, Y., and Liu, J. (2019b). Discourse-aware neural extractive model for text summarization. *arXiv preprint arXiv:1910.14142*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., and Lin, J. (2019). End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019*

Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 72–77, Minneapolis, Minnesota.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California.

Yasunaga, M., Zhang, R., Meelu, K., Pareek, A., Srinivasan, K., and Radev, D. (2017). Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada.

Yin, W. and Pei, Y. (2015). Optimizing sentence modeling and selection for document summarization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1383–1389, Buenos Aires, Argentina.

Yogatama, D. and Smith, N. A. (2014). Linguistic structured sparsity in text categorization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–796, Baltimore, Maryland.

Yoshida, Y., Suzuki, J., Hirao, T., and Nagata, M. (2014). Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839, Doha, Qatar.

You, Y., Jia, W., Liu, T., and Yang, W. (2019). Improving abstractive document summarization with salient information modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2132–2141, Florence, Italy.

Yu, N., Zhang, M., and Fu, G. (2018). Transition-based neural RST parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 559–570, Santa Fe, New Mexico, USA.

Zhang, B., Xiong, D., and Su, J. (2018a). Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, Australia.

Zhang, J., Tan, J., and Wan, X. (2018b). Adapting neural single-document summarization model for abstractive multi-document summarization: A pilot study. In *Proceedings of the International Conference on Natural Language Generation*.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2019a). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*.

Zhang, X. and Lapata, M. (2017). Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark.

Zhang, X., Lapata, M., Wei, F., and Zhou, M. (2018c). Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium.

Zhang, X., Wei, F., and Zhou, M. (2019b). HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy.

Zhang, Z., Blair-Goldensohn, S., and Radev, D. R. (2002). Towards cst-enhanced summarization. In *Eighteenth National Conference on Artificial Intelligence*, page 439445, USA.

Zhao, K. and Huang, L. (2017). Joint syntacto-discourse parsing and the syntacto-discourse treebank. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2117–2123, Copenhagen, Denmark.

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., and Zhao, T. (2018). Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia.

Zhu, C., Hinthorn, W., Xu, R., Zeng, Q., Zeng, M., Huang, X., and Jiang, M. (2020). Boosting factual correctness of abstractive summarization with knowledge graph. *arXiv preprint arXiv:2003.08612*.