博士論文(要約)

Linguistic Knowledge-Enhanced Neural Sentence Compression

(言語知識を活用したニューラル文圧縮)

by

Zhao Yang
趙 陽

A Doctor Thesis

Submitted to
the Graduate School of the University of Tokyo
on December 6, 2019
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Information Science and Technology
in Computer Science

Thesis Supervisor: Akiko Aizawa 相澤彰子
Professor of Computer Science

ABSTRACT

Sentence compression aims to shorten sentences to form a single compressed sentence (compression) that remains readable and preserves important information of the original sentences. With the amount of text data increasing exponentially, there is an urgent need for sentence compression systems automatically and efficiently condensing the significant amount of information such as news content into a short compression. However, yielding both grammatical and informative compression remains challenging in the area of sentence compression. Despite the progress over the past few years, three important issues have not yet or rarely been explored by previous studies. (i) how to effectively integrate different word-level linguistic features to improve the linguistic and content quality of the compression; (ii) how to define the sentence-level readability of the compression and optimize it explicitly; and (iii) how to yield readable and abstractive compression while maintaining the informative coverage.

To tackle the challenging issues mentioned above, we herein propose the linguistic knowledge-enhanced sentence compressor aiming to effectively and efficiently produce grammatical and informative compression. Concretely, we introduce three essential ideas. The first one is to introduce a gating mechanism capable of selectively exploiting word-level linguistic features. The second idea takes a step forward by considering sentence-level features with a specific focus on improving the readability of the compression. While the first two ideas focus on single sentence compression, the third idea is concerned with multiple sentence compression. A coarse-to-fine rewriter is proposed to polish the disfluent compression into a fluent one. We will detail each part below.

Firstly, the gating mechanism (thus a gated neural network) is introduced to selectively exploit linguistic features for single sentence compression. The motivation is that in some cases, part of speech of a word determines word deletion, while in other cases, dependency relations such as nominal subject (nsubj) dominate that deletion process. The introduced gated neural network is capable of effectively and dynamically determining which linguistic feature should be dominant, given different cases. Experimental results show that the proposed gating mechanism leads to better compression upon both automatic metrics and human evaluation, compared to previous competitive compression systems. Also, compression yielded by the proposed method share more grammatical relations in common with the ground-truth compression than the baseline method, indicating that important grammatical relations, such as subject or object of a sentence, are more likely to be kept in the compression by the proposed method.

Despite the promise brought by selectively exploiting word-level features, readability itself, as an essential evaluation aspect, is a sentence-level feature. To bridge the gap between evaluation and optimization objective, we introduce the second idea that explicitly takes the readability of compression into account via a reinforcement learning framework. A linguistic knowledge-enhanced language model is constructed as a grammar checker. Subsequently, a series of trial-and-error word deletion operations are conducted on the source sentence via this grammar checker in search of the best compression. The empirical study shows that the proposed model can effectively generate more readable compression, comparable or superior to several strong baselines.

In order to build a more practical sentence compressor, our third idea extends to multiple sentence compression, dealing with longer input text, e.t., multiple sentences. The motivation behind is that deleting unnecessary words or sentences and then concatenating the rest will result in incoherent and disfluent compression. To address this problem, we introduce a coarse-to-fine rewriter for multiple sentence compression. Firstly, several sentences were converted into a word graph whose output is coarse-grained compression. Then, the back-translation technique is used to polish coarse-grained compression into a more fluent one. Experimental results show that the proposed method produces more readable and context-aware compression meanwhile introducing a considerable amount of novel n-grams.

To conclude, this thesis proposes incorporating linguistic knowledge into sentence compression via deep neural networks. The resulting sentence compression model will benefit many applications such as displaying compressed text content in a screen with limited size, e.g., mobile phone, shortening a lengthy product title for the E-commerce platform, and compressing financial news clusters into a summary. We believe that this

thesis takes a further step towards the challenging issues of readability and informativeness in the research of sentence compression, yielding a more robust and accurate sentence compressor for the real-world applications.

Acknowledgements

First and foremost, I am genuinely thankful to have Professor Akiko Aizawa as my supervisor. She has supported me enormously throughout my Ph.D. life, most importantly guiding me on how to approach scientific problems and how to explore it scientifically. I have benefited tremendously from her advice and suggestions.

I also want to thank our lab member, Hajime Senuma, who provides suggestions to my research; Zhiyuan Luo for assisting me in the research work; Vitou Phy for his valuable suggestions and discussion; Andre Greiner-Petter for his valuable suggestions; and my friend, Xiaoyu Shen, who spend time discussing research ideas. Also, I took a lot of inspiration from the discussion with other lab members in our weekly lab meeting. I appreciate all of them very much.

Furthermore, I want to appreciate Mr. Hiroshi Kanayama from IBM Research Tokyo. He gave me a lot of suggestions on my research during my internship in IBM Research Japan.

I want to thank the secretory Noriko Katsu from our laboratory in the National Institute of Informatics. She has guided and assisted me with plenty of things in my lab.

Finally, I want to appreciate the tremendous support of my family. My father is always asking me to share with him the interesting things happening in the university. My mother is always encouraging me when I feel frustrated. Both of my parents are full of curiosity. Without their encouragement and financial support, my Ph.D. life could not go such smoothly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Owing to the information explosion and the amount of text data increasing exponentially, people are receiving an enormous amount of information on a daily basis. News websites such as Bloomberg[1] and social media websites such as Twitter[2] are currently the primary source of knowledge acquisition and sharing. Nevertheless, the increasing amount of textual information from multiple sources makes people constantly suffer from information overload. The term information overload (also known as information anxiety) refers to the difficulty in understanding an issue and effectively making decisions when one has too much information about the issue [79]; once information overload occurs, the decision quality is more likely to reduce [72]. Inevitably, information overload poses as a challenging issue on how to digest information efficiently and effectively.

Fortunately, the urgent need for text compaction systems has pushed forward the development of automatic text summarization techniques over the past few decades. The very first text summarization system made its way into research in the early 1950s. It was aimed at generating the abstract of technical papers and magazine articles. Statistical information derived from word frequency and distribution was used by the machine to compute a relative measure of significance, first for individual words and then for sentences. Sentences scoring highest in significance were then extracted and printed out as the "auto-abstract" [47]. Later on, [50] described text summarization as follows:

> Text summarization is a process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks).

The definition implies that the goal of text summarization is three-fold, namely, (i) to select the salient content, (ii) to shorten the length, simultaneously, (iii) and to satisfy the user's information needs. These three points are universally applicable to all text summarization research area, including sentence compression.

This thesis centers on sentence compression. It is therefore worthwhile to first clarify the position of sentence compression research in the field of text summarization. Sentence compression is a particular type of text summarization. Depending on textual granularity of the output, text summarization can be categorized as follows:

[1] $https://en.wikipedia.org/wiki/Bloomberg_News$.
[2] $https://en.wikipedia.org/wiki/Twitter$.

1. Document-level Summarization (multiple documents summarization and single document summarization) with multiple sentences as output.

2. Sentence-level Summarization (multiple sentence and single-sentence compression) with a single sentence as output.

3. word-level Summarization (keywords generation) with words and phrases as output.

All of them can be either extraction-based or abstraction-based, depending on whether they are choosing preexisting words, or rewording in the summary. Extractive summarization is dedicated to selecting the salient documents, sentences, or words from the sources without introducing new words in the summary. In contrast, abstractive summarization refers to a method that employs complicated deep learning to produce the summary using rewording. The sentence compression research mostly refers to extraction-based sentence-level summarization, which only deletes words in sources to form a summary. It is thus also called *deletion-based sentence compression* or *extractive sentence compression*. As a matter of fact, sentence compression also includes abstractive sentence compression[3]. For sentence-level summarization, extractive sentence compression is relatively simple yet effective in practice; as a result, most of the sentence summarization works, to date, are extraction-based. Compared to extractive summarization, a fully abstractive summarization requires not only deep semantic understanding of the text, but also natural language generation technique to produce a concise summary. It is technically harder but worth exploration.

Sentence Compression
Over the past 20 years, deletion-based sentence summarization received considerable attention. [37] is one of the earliest works in sentence compression. The focus of their work is to delete phrases, by their definition, including a word, a prepositional phrase, or a clause in the source sentence without changing the order of the resting words. An example sentence and one plausible compression are shown below:

- Sentence: *A man suffered a serious head injury after an early morning car crash today, according to the report.*
- Compression: *A man suffered a ~~serious~~ head injury after ~~an early morning~~ car crash ~~today, according to the report~~.*

There are surely other plausible compressions, for example, one of them could be *A man suffered a serious head injury ~~after an early morning car crash today~~, according to the report.* It is worth noting that these compressions may depend on user's query. Such kind of an on-demand sentence compression is called query-based sentence compression, which goes beyond the scope of this thesis. This thesis herein is concerned with generic sentence compression that aims to produce an informative compression for an average user without specifying a query.

In 2005, [7] introduced a new sentence compression task, called Multiple Sentence Compression (MSC). It differed from other tasks in that the input text was no longer a single sentence but multiple sentences, although the output compression remained a single sentence. Considering three input sentences about the same event "Lonesome George passed away".

[3]Several previous studies have also given considerable attention to abstractive sentence compression such as headline generation. However, it has relatively different benchmarks and paradigms.

- Sentence$_1$: *Tributes from former US presidents have poured in for Republican Senator John McCain, who has died aged 81.*
- Sentence$_2$: *Senator John McCain, an American Original, Died at Age 81.*
- Sentence$_3$: *US Senator John McCain died aged 81 after battle with brain cancer.*
- Compression: *Senator John McCain died aged 81.*

A plausible compression could be *Senator John McCain died aged 81.* Compared to single-sentence compression, multi-sentence compression has a similar evaluation but different research paradigm. We will detail the related works of both the single-sentence compression and the multiple sentence compression in Chapter 2.

From a practical perspective, sentence compression researches, either single-sentence compression or multi-sentence compression, are beneficial to numerous real-world applications. For example,

1. Compact text to be displayed on small screens [22], as shown in Figure 1.1.

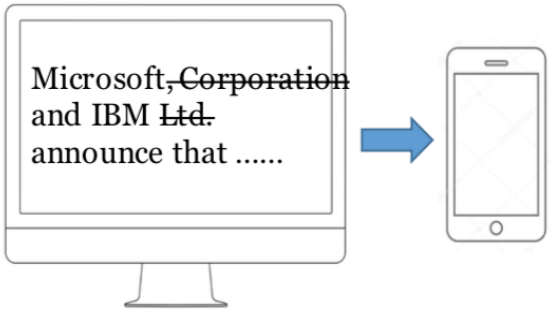


Figure 1.1: Display of compressed content.

2. Compress subtitle for high-rate speech, as shown in Figure 1.2.

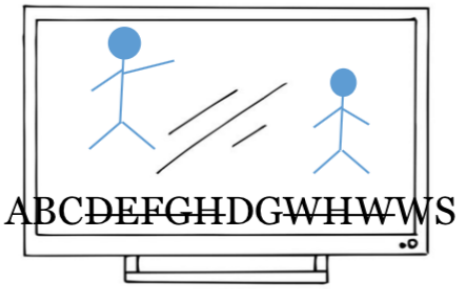


Figure 1.2: Subtitle compression for high-rate speech.

3. Cluster sentences in document(s) and compress similar sentences into a single concise summary. It serves as a pipeline step of document(s) summarization, as shown in Figure 1.3.

Figure 1.3: Compress sentence cluster in document(s) summarization.

Over the past five years, advances have been made in deep learning techniques. Sentence compression has also greatly gained from this progress; in particular, it evolved from the early rule-based approach to the recent entirely data-driven approach. Despite the promises that data-driven approaches hold for sentence compression, this research area is still facing several challenging issues, which we will elaborate in the next section of this chapter.

To summarize, this thesis is dedicated to sentence-level sentence compression, ranging from single-sentence compression to multi-sentence compression. Furthermore, we believe that a hybrid approach that augments the neural models with linguistic knowledge provides the best solution to the current challenges.

1.2 Challenges and Research Question

In this section, we first detail the challenges currently facing the sentence compression research. Then, we raise four research questions in an attempt to address these challenges, meanwhile resolving the shortcomings of previous studies.

1.2.1 Challenges

Sentence compression aims to generate a short version of the source text, which remains readable and preserves the informative content of the source. By definition, there are three critical (evaluation) aspects for this task:

1. Readability: compressed sentence should be readable.
2. Informativeness: compressed sentence should retain the important or informative content.
3. Compression rate[4]: how much content (tokens) should be removed. Compression rate is defined as summary length over source length, in line with previous study.

Among these aspects, a considerable amount of research work puts effort in improving the readability because informativeness and compression rate do not make sense if the compressed sentence is not even readable. To improve the readability and informativeness while maintaining a specific (or reasonable) compression rate, previous studies of single-sentence compression preferred two research lines.

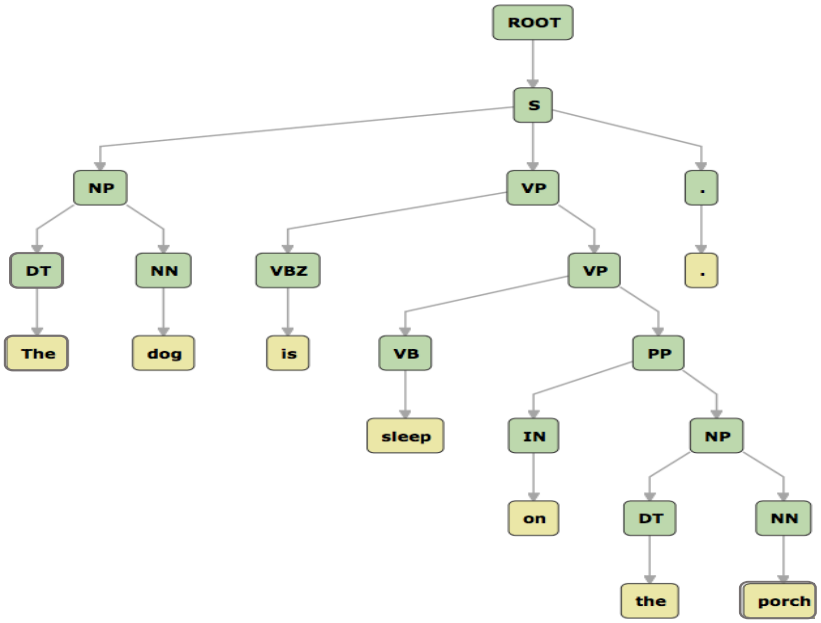


Figure 1.4: Constituency tree of the example sentence, *The dog is sleep on the porch.* Items in yellow are tokens while tokens in green are constituency components.

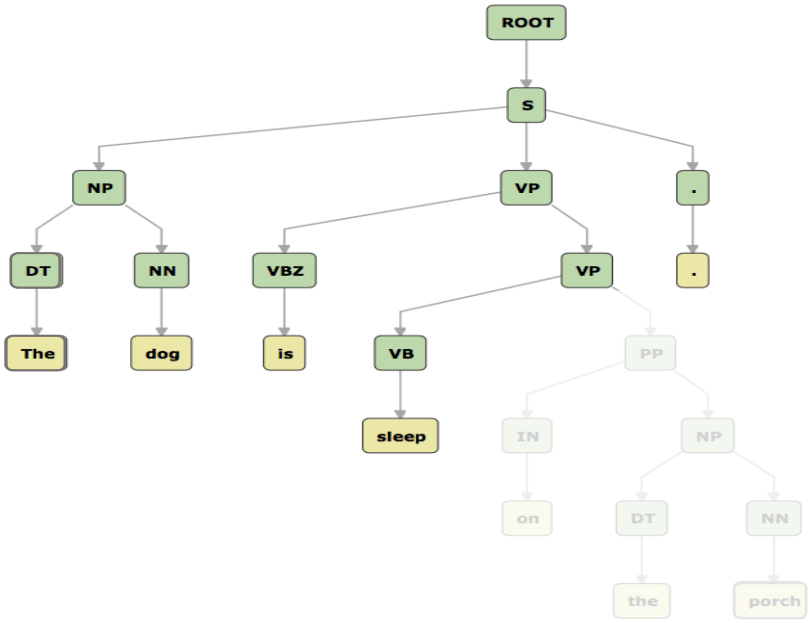


Figure 1.5: Abridged constituency tree of the example sentence, *The dog is sleep on the porch.* Items in yellow are tokens while tokens in green are constituency components. The prepositional phrase (PP) has been removed.

The first research line focuses on the rule-based approach that relies indirectly or directly on syntactic information [19, 31, 8, 9]. For example, a sentence shown

[4]Generic sentence compression does not have a clear constraint regarding the compression rate.

in Figure 1.4, *the dog is sleeping on the porch.*, one of the deletion rules, such as removing all the prepositional phrases (PPs) can be hand-crafted. The resulting compression, as shown in Figure 1.5, would be *the dog is sleeping.* Meanwhile, other works employ a parallel corpus to learn which constituency component should be deleted. However, the parallel corpus is too small (hundreds of parallel sentence compression pairs) to learn the production of a good quality compression.

More importantly, the challenge comes from that no matter how sophisticated the rule (or heuristics) is designed, there is always an out-of-the-box case where these rules fail. For example, the rule, either by handcrafted or learned, is to remove all the prepositional phrases in the text, the out-of-the-box case could be: *At the heart of the problem is a complex physical model proposed by Professor Paul.* as shown in Figure 1.6. When removing the prepositional phrases (PPs) in this case, the resulting compression, *is a complex physical model proposed.*, as shown in Figure 1.7, is no longer a grammatical sentence.

In short, the challenging issue in the first research line is that it is impossible to design deletion rules capable of being universally applied to all the sentence compression cases.

The second research line attempts getting rid of hard linguistic rules and approaches the problem from a data-driven perspective. The task is formulated as a sequence labeling problem. Given a sentence, $x_1, x_2, ..., x_n$ where x_i refers to the token in the sentence, learn a function $f(x; \theta)$ that converts the word sequence into zeros and ones sequence, $y_1, y_2, ..., y_n$ where $y_i \in \{0, 1\}$ and θ are learnable parameters of a machine learning model. 1 refers to keep the word, while 0 refers to delete the word in the sentence. θ is a learnable parameter. In the early days, widely used parallel datasets such as Ziff-Davis corpus [40] or Clarke-News corpus [17] composed of 1,000+ instances, which makes learning θ in $f(x; \theta)$ much difficult to yield promising results. There has not been much progress in this regard until 2013, [29] when a large-scale parallel corpus was introduced and [28], for the first time, recurrent neural network i.e., Long short-term memory network was applied, to sentence compression. Furthermore, in 2009, [51] employed the unigram accuracy as an automatic evaluation of improvements. Let us assume that the predicted $\hat{Y} = \{1, 1, 0, 0, 1\}$, and the ground-truth $Y = \{1, 0, 0, 0, 1\}$. This metric computes how many percent of labels are correctly predicted. In this case, the accuracy is 0.8.

Since then, the recurrent neural network-based approach has been the de facto standard for sentence compression. In 2016, [38] incorporated the eye-movement information into a recurrent neural network-based model under the multi-task learning framework to boost the token-level accuracy. Despite the success, as pointed out by [51], this token-level accuracy evaluation is not able to properly capture the readability of the compression because readability is sentence-level property while the objective function for e.g., maximum likelihood estimation of these neural models, is optimized only for word-level prediction. In other words, the second challenging issue comes from the gap between the "token-level" optimization objective and "sentence-level" evaluation.

Furthermore, although previous works such as [28] and [3] incorporated the word-level linguistic features including part-of-speech tag and dependency relation label, little is known about whether integrating structure-level feature in a neural model would be beneficial to the prediction. Moreover, there is a lack of comprehensive investigation on how these linguistic features contribute to the compression quality.

The readability issue (or called grammaticality in a few works) poses the

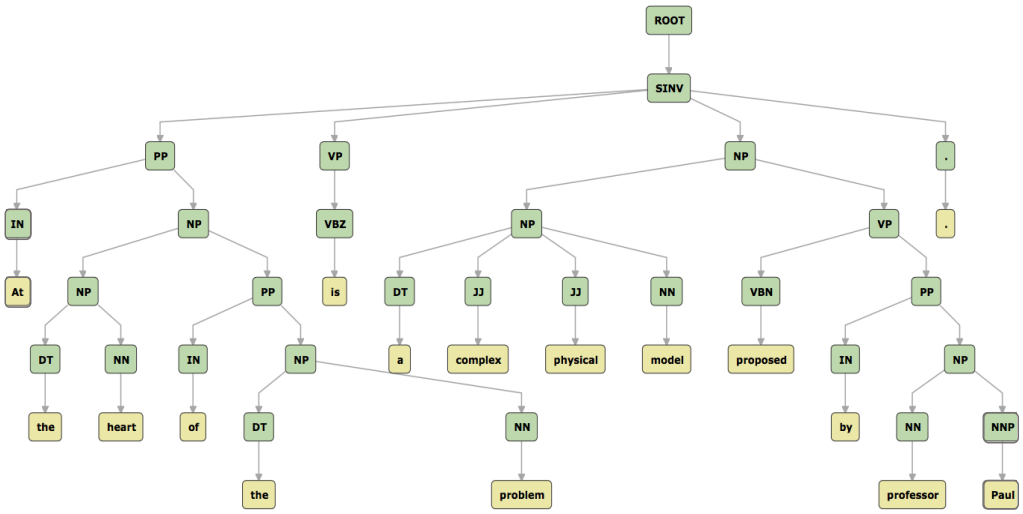


Figure 1.6: Constituency tree of the example sentence, At the heart of the problem is a complex physical model proposed by professor Paul. Items in yellow are tokens, while items in green are constituency components.

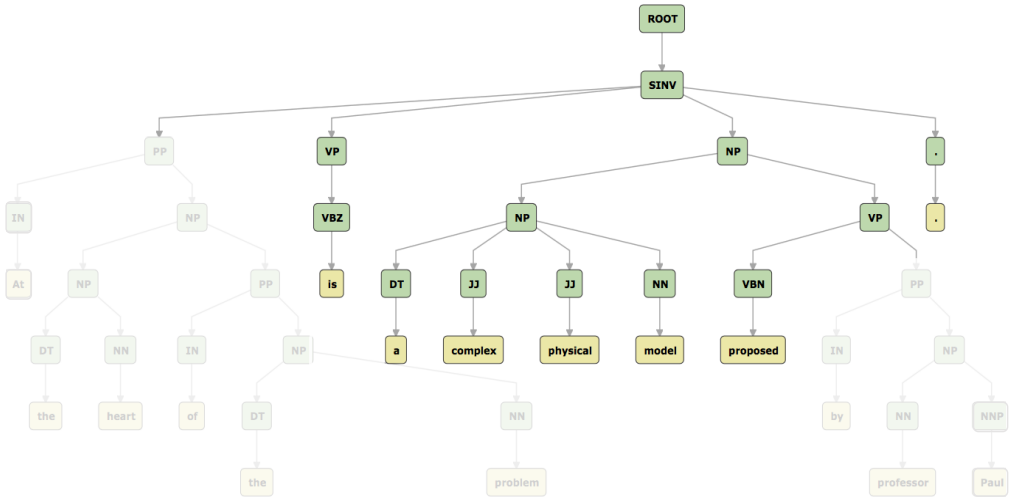


Figure 1.7: Abridged constituency tree of the example sentence, *At the heart of the problem is a complex physical model proposed by professor Paul.* Items in yellow are tokens while tokens in green are constituency components. It removes the prepositional phrase (PP).

same or even greater challenge to the multiple sentence compression task where the compression output is much longer than that of single-sentence compression task. Statistics on the benchmark datasets show that the length of single-sentence compression is around 19 tokens on average, while that of multiple sentence compression is around 37 tokens on average. The increase of output length augments the possibility of making grammatical mistakes for sentence compression model.

The research line of multiple sentence compression is starting from [7]. Later in 2010, [55] introduced a high-quality parallel compression corpus with 300 instances, which has become the benchmark in multiple sentence compression (English language) and largest dataset; The same year, Filippova presented a simple

way to produce compression from word graph [27]. Due to the simplicity and effectiveness, it was inspiring a lot of subsequent works developing its variants such as [10, 6, 59].

However, despite the rapid progress made by the past research, the challenges remain two-fold: first, as identified by [27, 59], the previous method gets higher informativeness scores at the cost of a drop in readability. Therefore, overcoming the conflict between informativeness and readability is one of the biggest challenges. Secondly, according to our knowledge, there is a lack of large-scale parallel corpus to investigate whether the neural network-based approach could provide promise for multi-sentence compression task.

For both the single-sentence compression and the multi-sentence compression tasks, informativeness is more challenging to tackle. There is a trade-off between readability and informativeness to produce a good compression. Based on these observations, we shall raise several research questions in the next section in an attempt to address these challenges.

1.2.2 Research Questions

The goal of this thesis is to improve the readability for (both single and multiple) sentence compression output and explore how to effectively model the informativeness. From our perspective, we hypothesize that a hybrid approach that augments the neural network-based model with linguistic knowledge could provide a solution for the above-mentioned challenges. In particular, we are interested in answering the following research questions:

1. Whether sentence compression neural models could be enhanced by incorporating word-level and sentence-level linguistic knowledge? And, how will they contribute to a better performance?
2. What would be the relationship among readability, informativeness, and compression rate in order to ensure the good quality of the compression?
3. How can we model and optimize both readability and informativeness in a more explicit way?
4. Given the lack of training data, can we overcome the conflict between readability and informativeness for multiple sentence compression using deep learning techniques?

We believe these questions could help us gain a better understanding of the current challenge and indicate the possible solutions.

1.3 Contributions

This section elaborates our approach and contribution to the sentence compression research. We have three main chapters, each exploring some of the technical challenges we discussed earlier, while simultaneously, providing answers to our proposed research questions.

Throughout these three chapters, linguistic knowledge plays an essential role. In chapter 3, we present a neural model augmented with a gating mechanism capable of selectively exploiting three types of word-level linguistic knowledge, namely, part-of-speech tags, dependency relation, and named entity tags. We

also investigate the contribution of each linguistic feature to the overall performance. Further, Chapter 4.1 considers the structure-level (sentence-level) linguistic knowledge and presents a novel self-attention model augmented with a syntactic tree structure bias. Empirical results on six downstream benchmark datasets demonstrated the effectiveness of integration of both the word-level and the sentence-level linguistic knowledge.

As there has been no previous study modeling informativeness in the context of sentence compression, Chapter 4.2 takes the first step in search of appropriate automatic evaluation metrics for informativeness; Ten candidate quantities were compared to investigate which quantity could correlate the best with human judgment on informativeness. Chapter 4.3 presents a novel knowledge-based evaluator for explicitly optimizing both the readability of the compression.

To overcome the conflict between readability and informativeness of multiple sentence compression, Chapter 5 introduces external knowledge, i.e., WordNet and Paraphrase Database (PPDB), together with a coarse-to-fine rewriter to polish the coarse-grained compression into a fine-grained one using the external knowledge. We introduce a significant amount of novel n-grams in the compression, while simultaneously, improving its readability. Furthermore, a large-scale multi-sentence compression corpus with more than 140k instances is constructed to alleviate the lack of training dataset for neural network-based models.

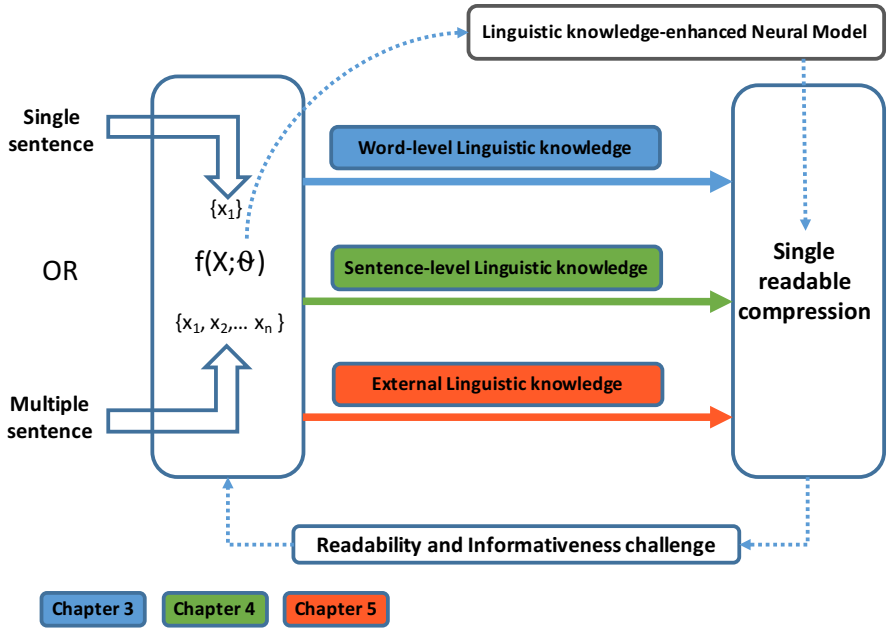


Figure 1.8: A graphical illustration of the technical overview of this thesis.

Figure 1.8 provides a high-level technical overview of our proposed approaches in this thesis. We will detail how we organized the entire thesis in the next section.

1.4 Thesis Organization

The remainder of the thesis is organized as follows. In chapter 2, we provide a background introduction and review related works. This chapter begins with an introduction of sentence summarization in a broader context of text summarization. We briefly clarify the position of sentence compression in field of text

summarization. A brief introduction of text summarization in Chapter 2.1, is followed by a detailed introduction of both single-sentence compression (Chapter 2.2) and multiple sentence compression (Chapter 2.3). The Background section ends with a summary.

Chapters 3, 4, and 5 discuss our proposed approaches. We hypothesize that integration of linguistic knowledge into neural models can address the challenges facing the sentence compression research. Chapter 3 presents a neural model augmented with a gating mechanism capable of exploiting three types of word-level linguistic knowledge, namely, part-of-speech tags, dependency relation labels, and named entity tags, as shown in Figure 1.9.

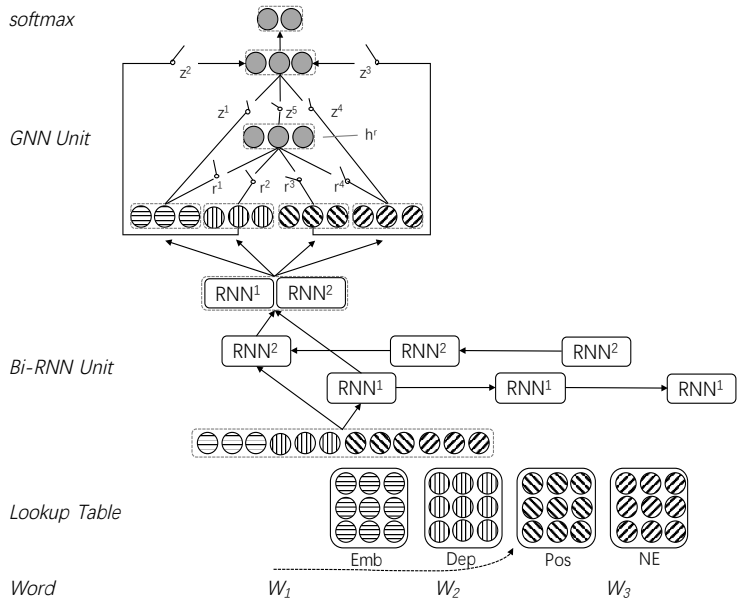


Figure 1.9: A graphical illustration of the gated neural network. Emb, Dep, and Pos respectively stand for word embedding, dependency, and part-of-speech lookup tables.

Chapter 4 describes how to incorporate the sentence-level linguistic knowledge. A structural bias was introduced in Chapter 4.1; we then investigate the quantity that could serve as a proxy of human informativeness assessment in Chapter 4.2, which is incorporated into the evaluator. Chapter 4.3 details the proposed evaluator as shown in Figure 1.10, which is capable of optimizing both readability and informativeness of the compression.

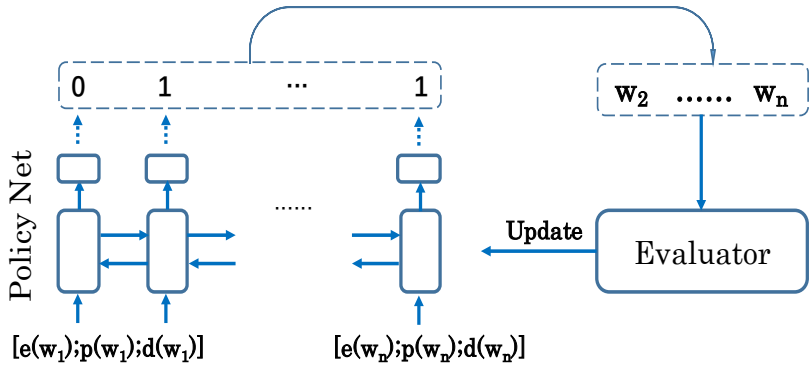


Figure 1.10: A graphical illustration of an evaluator-based framework.

Chapter 5 describes a coarse-to-fine rewriter to polish the coarse-grained compression into fine-grained one using the external knowledge, as shown in Figure 1.11.

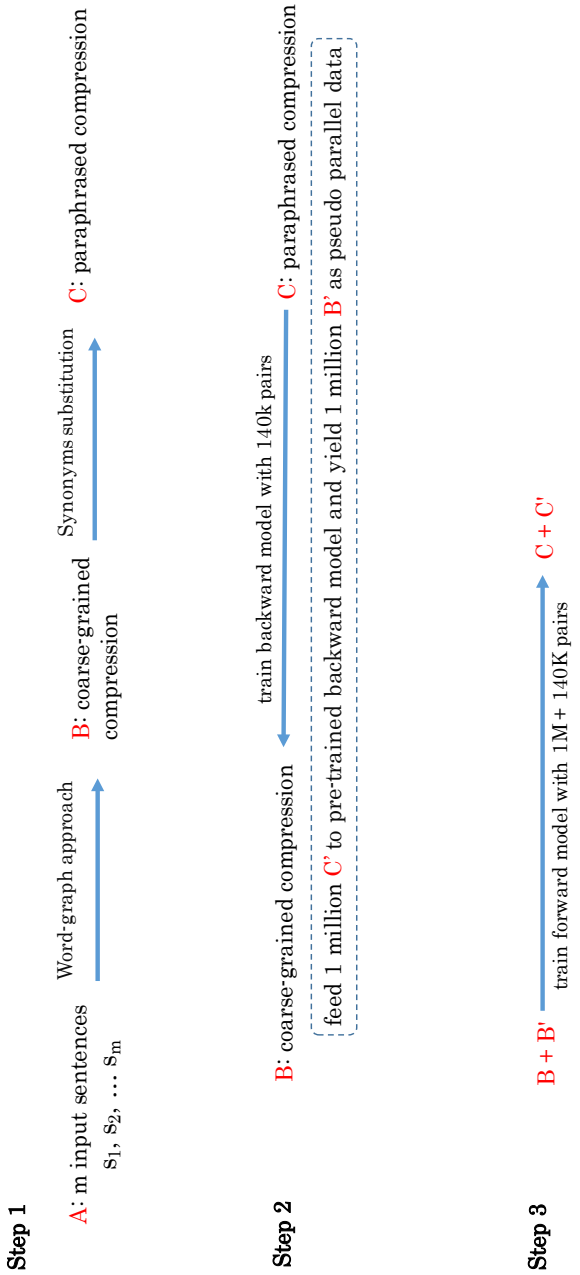


Figure 1.11: A graphic illustration for our rewriter model. A refers to multiple input sentences. B denotes a single compressed sentence using the word graph approach. C is the paraphrased sentence using external knowledge WordNet 3.0 and PPBD 2.0 . C' is a large-scale and in-domain monolingual corpus, while B' refers to the predicted compression using a pre-trained backward model given as C'. $B + B'$ and $C + C'$ are the mixing datasets from both *step*.1 and *step*.2, respectively.

Chapter 6 concludes this thesis. In Chapter 6.1, we discussed the research

questions raised in chapter 1 by summarizing our technical contributions, which are discussed throughout the thesis; In Chapter 6.2, we then discuss the limitations of the proposed approaches; Chapter 6.3 describes future directions in which the sentence compression would go.

Chapter 2

Background

In this chapter, we review the related works with respect to text compression. Section 2.1 gives a background overview in the broader context of text summarization. Section 2.2 details the related studies on single sentence compression, while the third section elaborates the past research into multiple sentence compression. It is followed by a chapter summary.

2.1 Text Summarization

Text summarization, together with natural language processing research, started in the 1950s [47]. The very first system is developed for producing abstract for technique and scientific articles. Later on, text summarization evolved a lot but most of them focus on extractive summarization, which goes in the following way:

In general, the extractive text summarization consists of three steps [1], as shown in Figure 2.1:

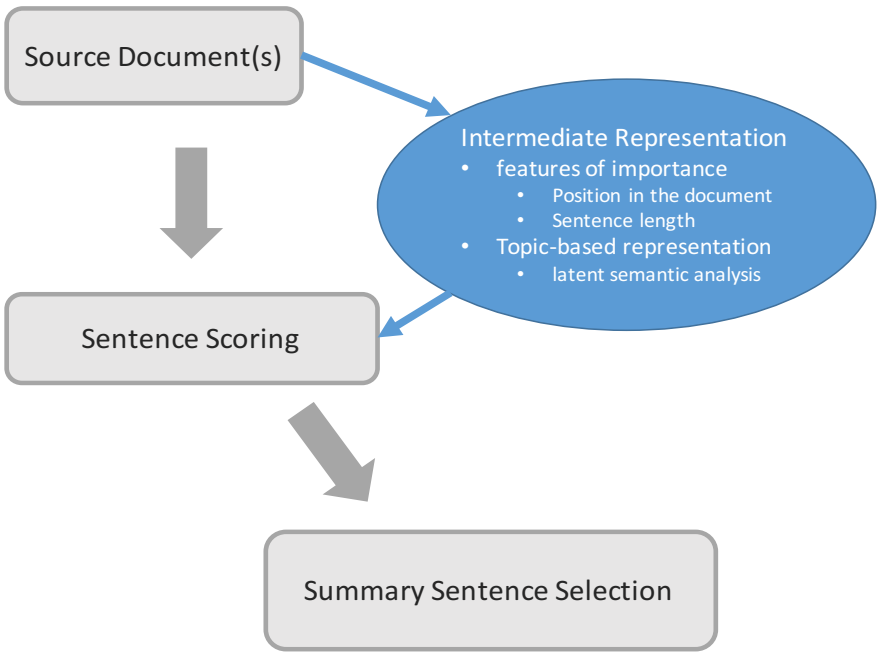


Figure 2.1: Workflow of extractive text summarization.

1. Convert the original text into intermediate representation, either being feature-based or topic-based.

2. Based on intermediate representation, scoring the sentence in the document accordingly.

3. Based on scoring, choose top-n sentences as the output or select the summary sentence under several constraints such as length of the summary, etc.

Because of the simplicity and effectiveness, extractive text summarization has received considerable attention since its very beginning. Compared to abstractive text summarization, there are two advantages:

1. The grammaticality (or readability) of summary can be guaranteed because sentences are directly selected from the source document(s).

2. Compared to abstractive summarization, which involves the semantic representation and natural language generation, extractive summary is more simple and usually adequate to satisfy the needs in practice.

Due to these reasons, there is a popularity of extraction-based approaches. However, it is not without disadvantage: the extracted sentences might not be coherent to each other, and salient information is usually spread across sentences, extracted summaries are usually longer than average, inevitably including unimportant fragments. To tackle this issue, sentence compression technique is introduced to shorten the lengthy sentence, as a pipeline of text summarization, deleting irrelevant or unimportant words in the extracted sentences. As shown in Figure 2.2 and Figure 2.3, sentences are extracted from text and either single sentence compression (Figure 2.2) and multiple sentence compression (Figure 2.3) technique are employed to produce the final concise summaries [37, 45, 80, 6].

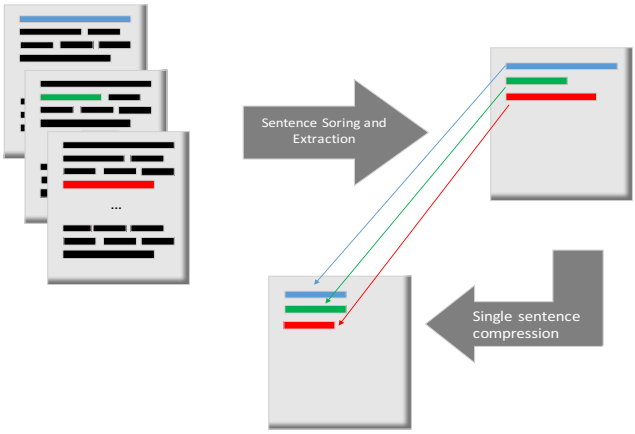


Figure 2.2: Single sentence compression for extractive text summarization.

In 2000s, single and multiple sentence compression start to become independent tasks with wide applications not only in serving as a pipeline in summarization but also a critical natural language processing technique for text compaction to meet the real world needs. For example, compress content to be displayed on very small screens like mobile phones [22], subtitle compression for high-rate speech [74] and multiple news sentence fusion [7]. In the following section, we

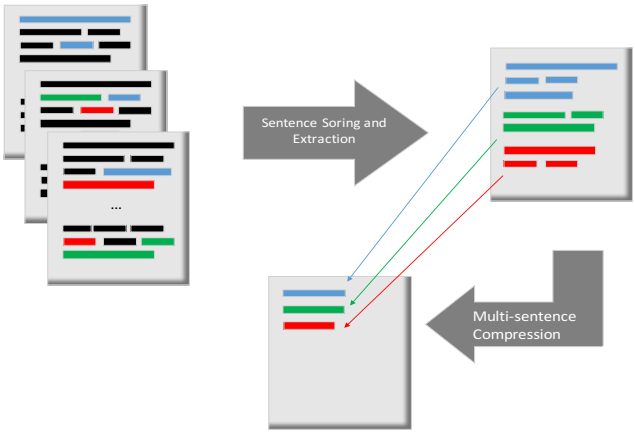


Figure 2.3: Multiple sentence compression for extractive text summarization.

will review several important works in the development of sentence compression techniques. We also elaborate the problem settings and terminology in beginning of each section.

2.2 Single Sentence Compression

Sentence compression aims to produce a single shorter and readable sentence (compression) while preserving the important content. For example, given the following sentence, there are several possible candidate outputs:

- Sentence: *In this article, we present the concept of sentence fusion which, given a set of similar sentences, produces a new sentence containing the information common to most sentences in the set.*
- Compression(1): In this article, we present the concept of sentence fusion.
- Compression(2): We present the concept which produces a new sentence containing the information common.
- Compression(3): We present the concept.

There are two research lines regarding single sentence compression technique, (i) rule-based approach and (ii) data-driven approaches. The rule-based approaches uses linguistic knowledge like syntactic information as signals [37, 53, 19, 9] or generate compression directly by pruning constituency or dependency trees [39, 8, 29]. The later has been given much attention and we will illustrate tree pruning-based approaches. As this thesis focus on neural sentence compression, we kindly refer readers to above-mentioned work for details of tree-based approach. We herein briefly introduce the core idea of constituency tree-based and dependency tree-based approaches with several examples.

Constituency Tree Pruning Approach

Figure 2.4 shows the constituency tree of an example sentence, *In this article, we present the concept of sentence fusion which, given a set of similar*

sentences, produces a new sentence containing the information common to most sentences in the set. To produce compression, we handcraft pruning rules. For example, if removing all the Subordinate Clauses (SBARs), we yield an abridged constituency tree, as shown in Figure 2.5. Then, tree traversal algorithm (e.g., postorder traversal) was applied to linearize the abridged tree into compression (1), *In this article, we present the concept of sentence fusion.*. Likewise, if we handcraft pruning rules to delete all the prepositional phrases (PPs), abridges tree in Figure 2.6 will result in compression(2); if we remove both Subordinate Clauses (SBARs) and prepositional phrases (PPs) in the tree, a very short reduced sentence, compression(3) can be yielded, as shown in Figure 2.7.

The focus of the approach is to identify the proper set of constituents to be removed so that the resting components can still form a short yet readable and informative sentence. These rules can be either handcrafted [33] or learned by model [53, 77]. It is worth noting that most of the constituency tree-based approaches require training data to learn which constituents can be removed from a sentence.

Dependency Tree Pruning Approach

By comparison, another tree pruning approach can be fully unsupervised. It is based on dependency tree. Unlike constituency tree, each node in dependency tree is a token, and the edge is the directed dependency relation between two nodes (the curved arrow of the edge is from the children nodes to the parent node). Previous works such as [30] elaborate how to obtain grammatical compression in an unsupervised fashion. One of the important rules that guarantee grammaticality is that if the parent node is removed, all its children nodes should be removed as well. With this grammaticality guarantee, three possible compressions are yielded by pruning the dependency tree (Figure 2.8) of the same input sentence. Compression(4) (Figure 2.9) is yielded by removing all *amod* dependency relations; Compression(5) (Figure 2.10) is yielded by removing all *prepositional phrase* dependency relations; Compression(6) (Figure 2.11) is yielded by removing both *amod* and *prepositional phrase* dependency relations.

- Sentence: *In this article, we present the concept of sentence fusion which, given a set of similar sentences, produces a new sentence containing the information common to most sentences in the set.*
- Compression(4): In this article, we present the concept of sentence fusion which, given a set of sentences, produces a sentence containing the information in the set.
- Compression(5): we present the concept which, produces a new sentence containing the information common.
- Compression(6): we present the concept which, produces a sentence containing the information.

To summarize, the tree-based approaches can be either unsupervised by handcrafting the pruning rules [30, 19] or supervised by learning [53, 77]. However, they are facing the challenging issue that rules may not cover all cases. For example, as for the sentence, *At the heart of the problem is a complex physical model proposed by professor Paul*, removing prepositional phrase will yield ungrammatical compression.

Figure 2.4: Constituency tree of original sentence, ***In this article, we present the concept of sentence fusion which, given a set of similar sentences, produces a new sentence containing the information common to most sentences in the set.***

Figure 2.5: Abridged constituency tree corresponds to the compression(1), *In this article, we present the concept of sentence fusion.*

Figure 2.6: Abridged constituency tree corresponds to the compression(2), *We present the concept which produces a new sentence containing the information common.*

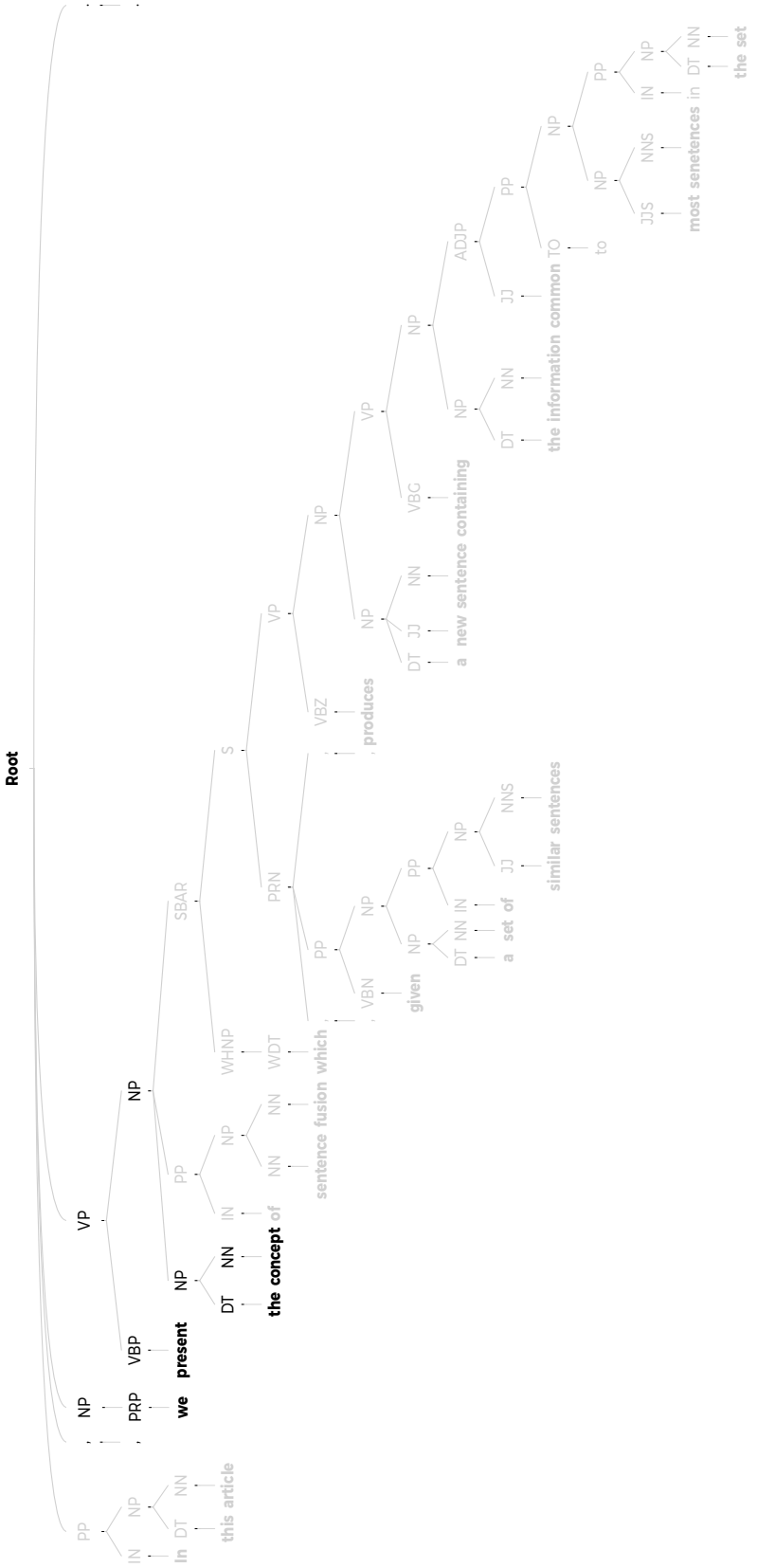


Figure 2.7: Abridged constituency tree corresponds to the compression(3), *We present the concept.*

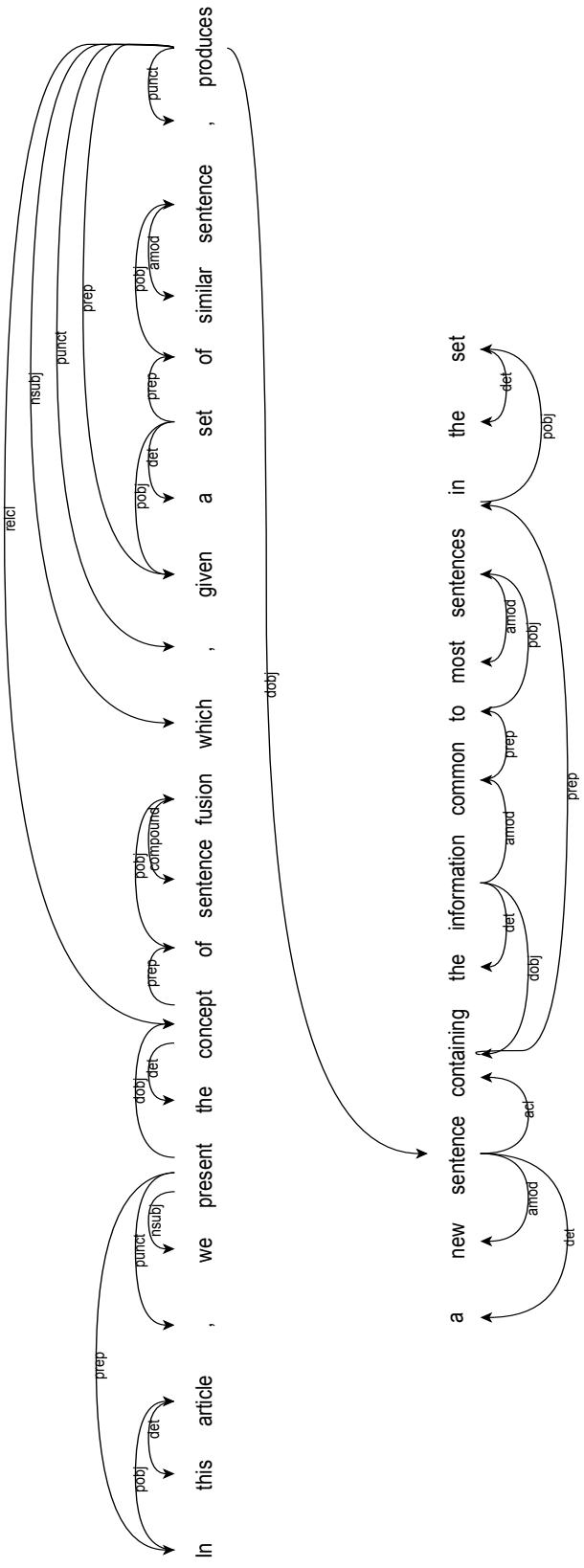


Figure 2.8: Dependency tree of the original sentence, ***In this article, we present the concept of sentence fusion which, given a set of similar sentences, produces a new sentence containing the information common to most sentences in the set.***

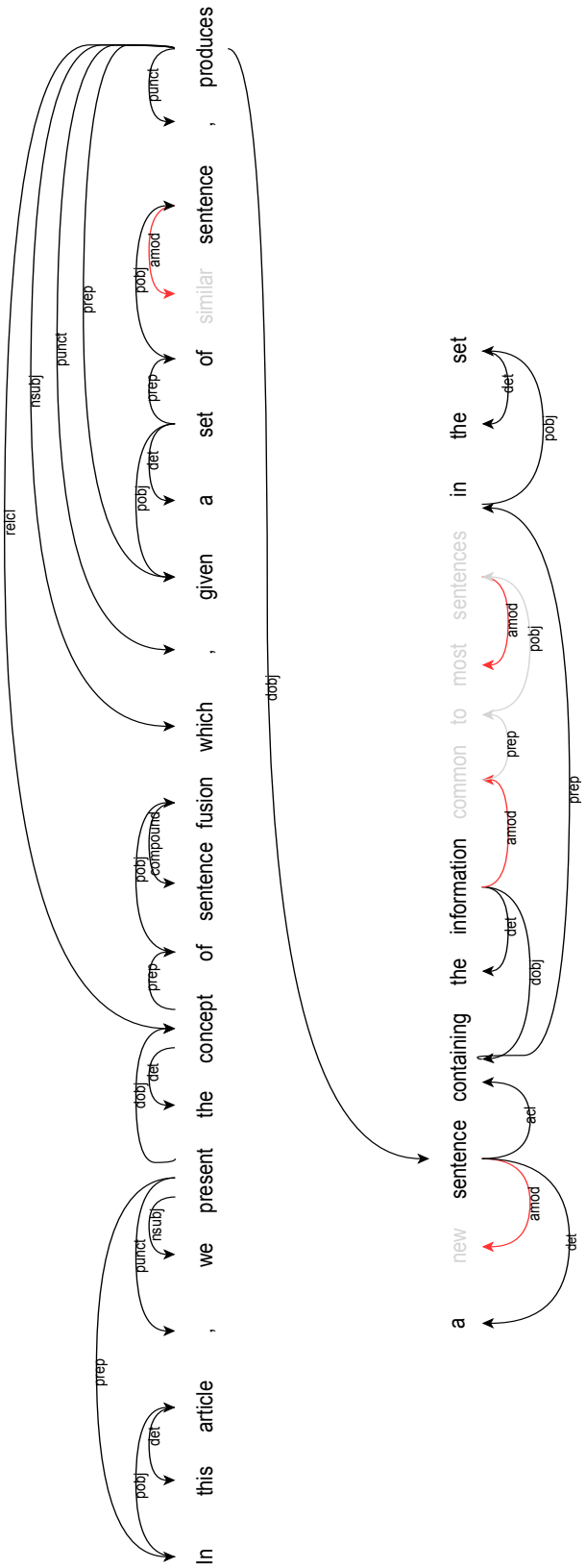


Figure 2.9: Abridged constituency tree corresponds to the compression(4), *In this article, we present the concept of sentence fusion which, given a set of sentences, produces a sentence containing the information in the set.*

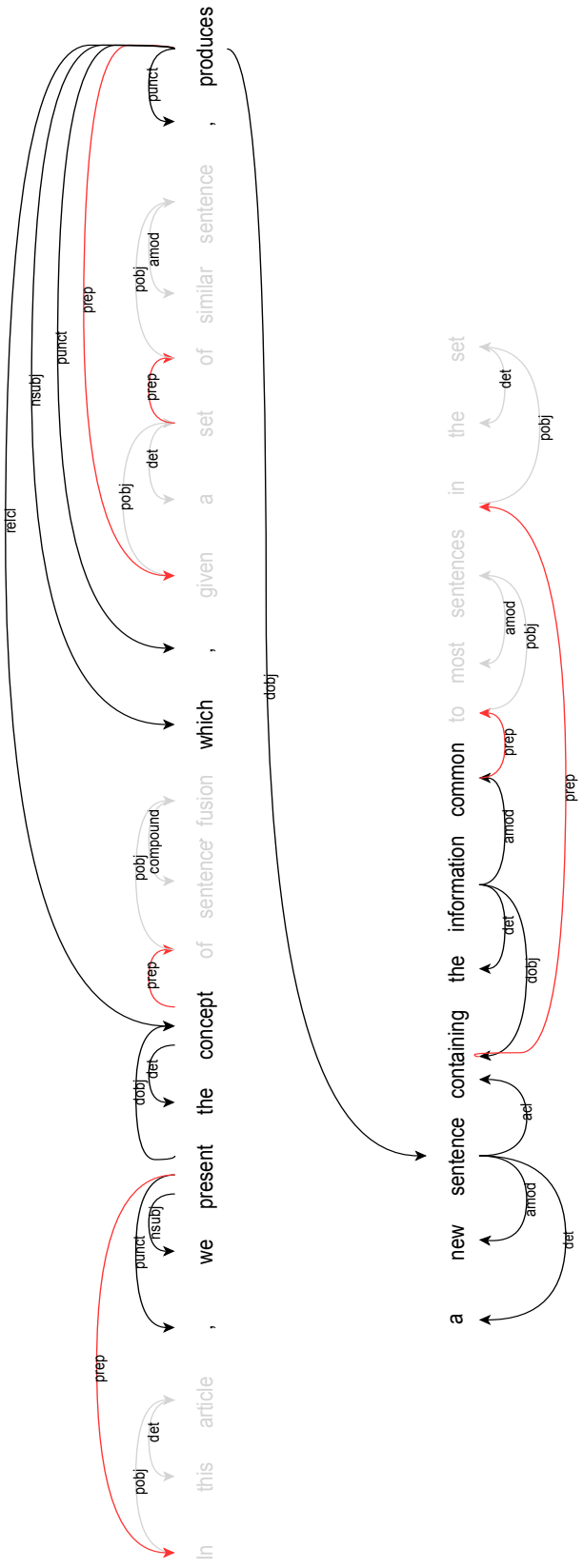


Figure 2.10: Abridged constituency tree corresponds to the compression(5), *we present the concept which, produces a new sentence containing the information common.*

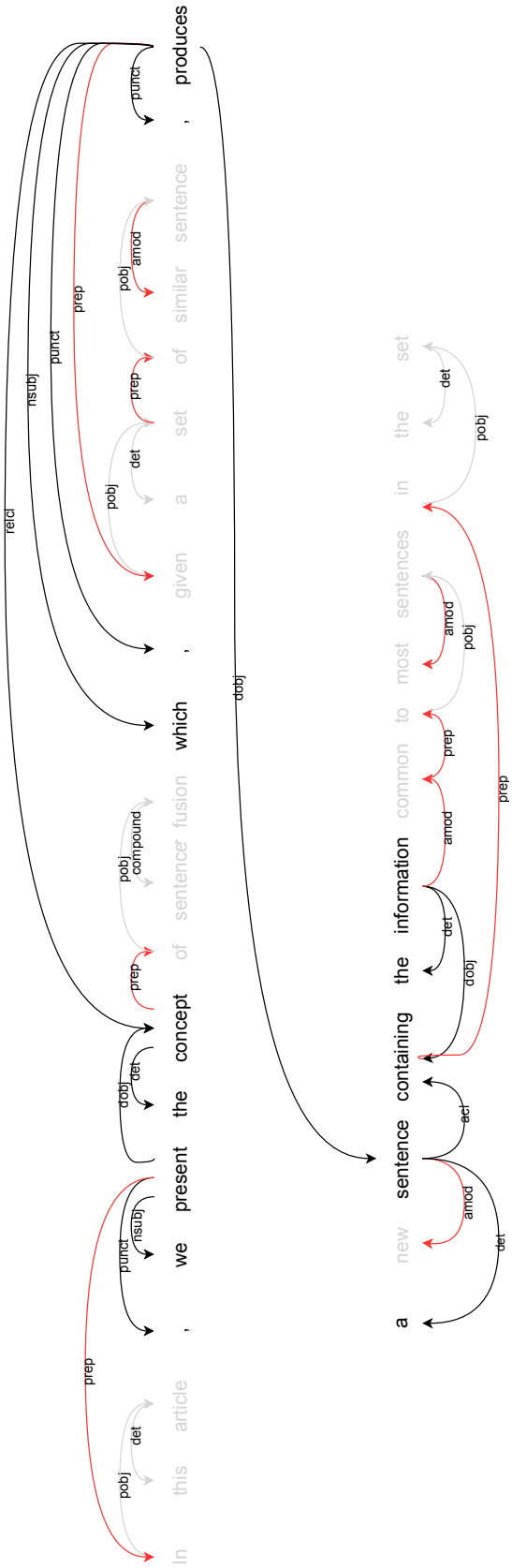


Figure 2.11: Abridged constituency tree corresponds to the compression(6), *we present the concept which, produces a sentence containing the information.*

Data-Driven Neural Approach

From 2013, the focus is shifting from rule-based approach to data-driven approach [29]. In 2015, Filippova et al. [28], for the first time, applied recurrent neural network for sentence compression, yielding encouraging results. In 2016, [3] represented a neural network architecture for deletion-based sentence compression using over 2.3 million instances using the method in [29]. They represented word with word embedding, dependency labels, as well as part of speech tags, sliding a window left-to-right to make decisions based on the local context. However, their large-scale deletion-based sentence compression datasets is not publicly available. In the same year, Klerke et al. [38] leveraged eye-movement information as external knowledge in a multi-task learning fashion to achieve comparable performance.

Although an RNN (e.g., Long short-term memory networks) can implicitly model syntactic information, it still produces ungrammatical sentence. We argue this is because that optimization objective of an RNN is the likelihood function which is based on individual words instead of readability of the whole compressed sentence. Therefore, a gap exists between optimization objective and evaluation.

To tackle these issues, this thesis takes the sentence compression problem as a sequence labeling task, and present a linguistic knowledge-enhanced neural sentence compression model. Different from the previous works such as [3], our proposed approaches have the following advantages: firstly, it is able to selectively make use of linguistic features at both word-level and sentence-level. Secondly, our approach bridges the gap between optimization objective and evaluation.

2.3 Multiple Sentence Compression

Multiple sentence compression is a text-to-text generation technique which, given a group of related sentences, produces a shortened sentence covering all or a portion of the relevant information from the inputs. [7] presented a very early work in the field of multiple-sentence compression. They introduced a text-to-text generation technique of expressing content common to most of the input sentences in a single sentence. Later on, [31] viewed multiple sentence compression as an integer linear programming problem, and showed promising results for German. However, this method relies on linguistic rules and a dependency parser, which is not available for all languages.

In 2010, [27] proposed word graph approach that only requires a POS tagger. It is widely used and updated by the following research works. A simple illustration are as follows: considering three related sentences about the same event (e.t., John mccain passed away).

1. Tributes from former US presidents have poured in for Republican Senator John McCain, who has died aged 81.
2. Senator John McCain, an American Original, Died at Age 81.
3. US Senator John McCain died aged 81 after battle with brain cancer.

As shown in Figure 2.12, a directed word graph is constructed from a group of related sentences in which nodes represent unique words (punctuation is excluded), and edges express the adjacent relation of words. To begin with, the START and the END symbols (in red color) are added in the word graph. Then,

each word in the sentences is mapped onto a node in the graph in order until the last word (from the first word in sentence (1) to the last word in sentence (3)). The key point of mapping is that if there already exists a word node with same lowercased word form and the same part of speech, just mapped to the existing node and do not create any new node. Using part of speech information could reduce the chances of merging verbs with nouns (e.g., visit) and generating ungrammatical sequences, as suggested by [27]. When the creation of the graph is completed, the next step is to define edge weights.

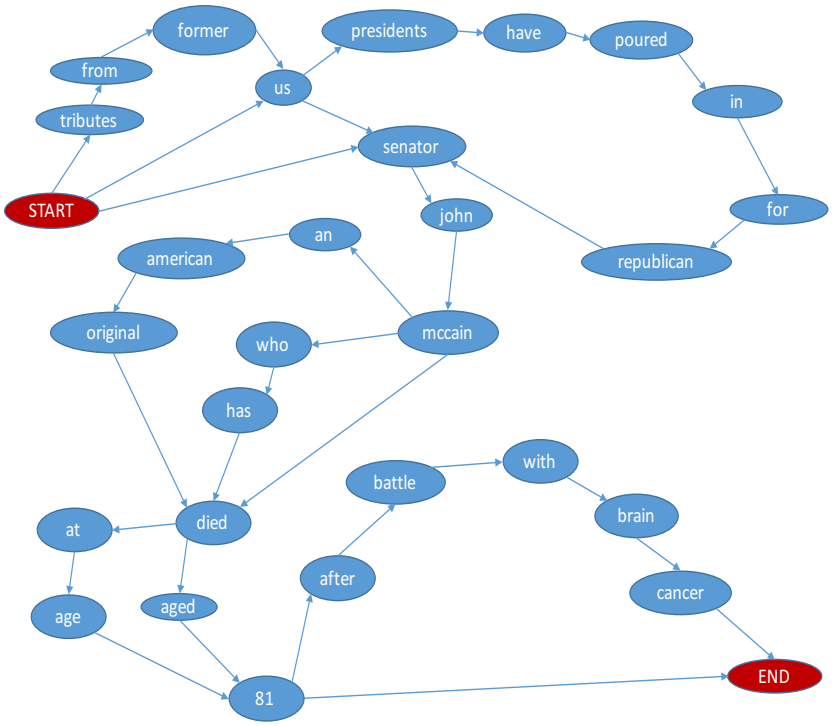


Figure 2.12: Graphical illustration of word graph approach. Nodes are words in sentence, while edges indicates a direct succession relation of words. The graph starts from node START to node END. Each edge corresponds to a pre-defined weight which is omitted for clarity.

Edge weights are calculated using the pre-defined weighting function. The purpose of this function is two-fold: (i) to generate a grammatical compression, it favors strong links, i.e., connections between words which appear significantly often in this order; (ii) to produce an informative compression, it promotes paths in the graph that are passing salient nodes.

With respect to (i) strong links, the assumption behind is that the redundancy is the guarantee of grammaticality. Natually, the (i) is related to frequency. To be more specific, compression path going through edges between words that are frequently associated to each other is favored. Furthermore, longer paths between words are weak signals of word association. The weight of an edge between two nodes i and j is thus reduced for every possible path between them but reduced proportionally to its length, as defined by [27]:

$$\mathrm{w}(\mathrm{e}_{\mathrm{i,j}}) = \frac{\mathrm{frequency(i)} + \mathrm{frequency(j)}}{\sum_{\mathrm{s}\in\mathrm{S}} \mathrm{diff(s,i,j)}^{-1}}, \tag{2.1}$$

$$\mathrm{diff(s,i,j)} = \mathrm{pos(s,i)} - \mathrm{pos(s,j)}, \quad \text{if } \mathrm{pos(s,i)} < \mathrm{pos(s,j)}, \tag{2.2}$$

$$\mathrm{diff(s, i, j)} = 0, \quad \text{otherwise.} \tag{2.3}$$

where $diff(s, i, j)$ refers to the distance between the offset positions ($pos(s, i)$) of words i and j in sentence s.

With respect to (ii) salient words, since the function above only consider the association between two nodes is. It assigns equal weights to edges connecting words encountered in a single sentence and words encountered next to each other in every sentence. To produce the compression that containing most significant words, they force the path to go through most frequent nodes by redefining the edge weight formula:

$$\mathrm{w'(e_{i,j})} = \frac{\mathrm{w(e_{i,j})}}{\mathrm{frequency(i)} \times \mathrm{frequency(j)}}. \tag{2.4}$$

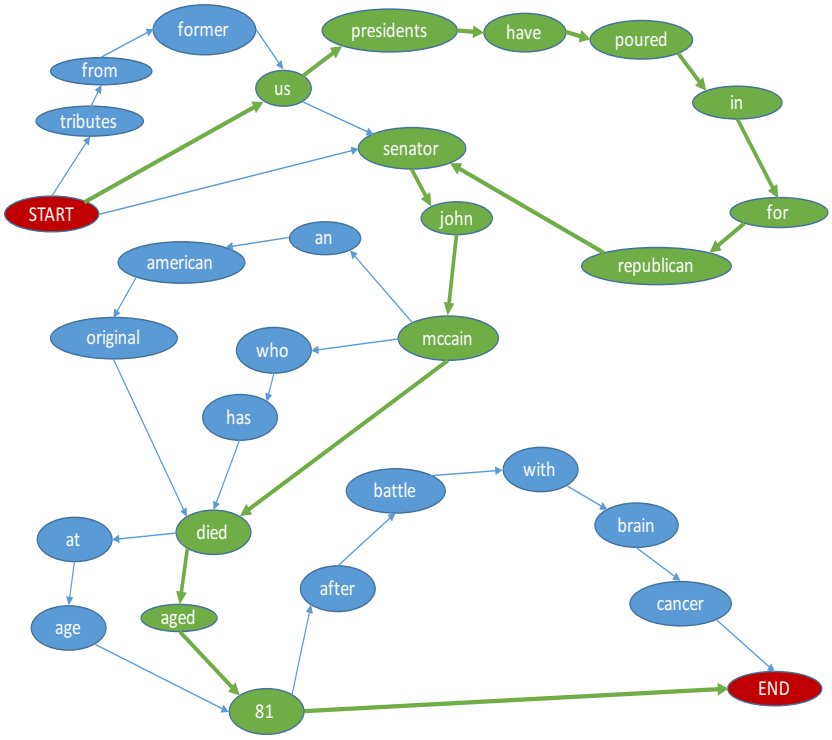


Figure 2.13: Graphical illustration of compression candidate (a) whose path is in green color, starting from node START to node END.

The weight function is used to find K shortest paths from start to end nodes in the graph, as our purpose is to compress sentences. Figure 2.12 2.18 shows six possible compression candidates generated by the word graph approach with different lengths. Especially, compression candidate (e) in Figure 2.17 and compression candidate (f) in Figure 2.18 are respectively the shortest and longest possible compression produced by the graph. We can observe that the compression is the trade-off between grammaticality and information coverage. The longer the compression is, the more information it covers.

In spite of its simplicity, [27] reported that the original word graph approach missed 48%–60% important information, which motivated later works, such as that of [10], to identify the keyphrase using an unsupervised approach [75] to cover as much important information (informativeness) as possible. However,

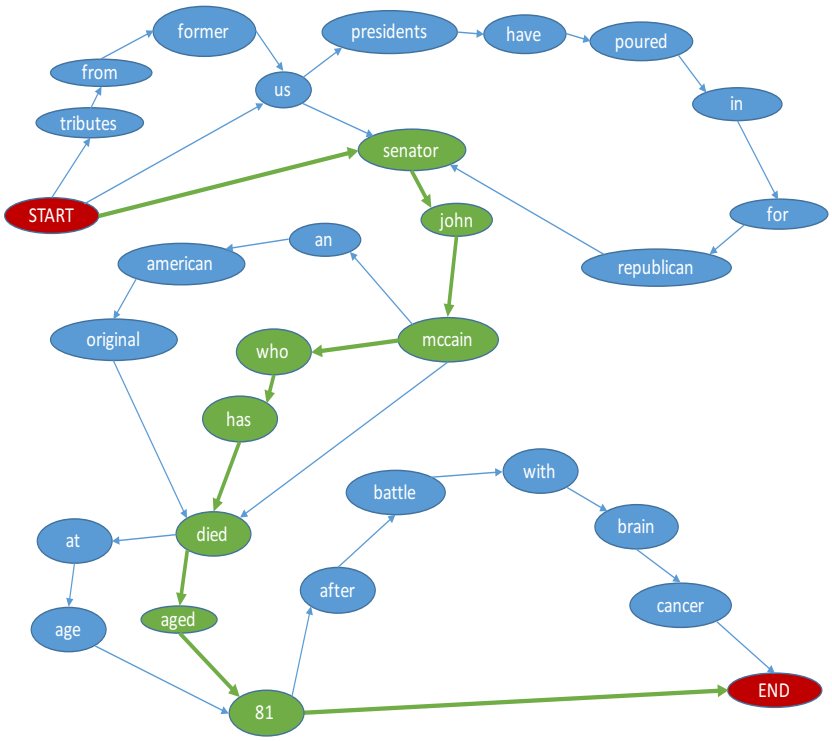


Figure 2.14: Graphical illustration of compression candidate (b) whose path is in green color, starting from node START to node END.

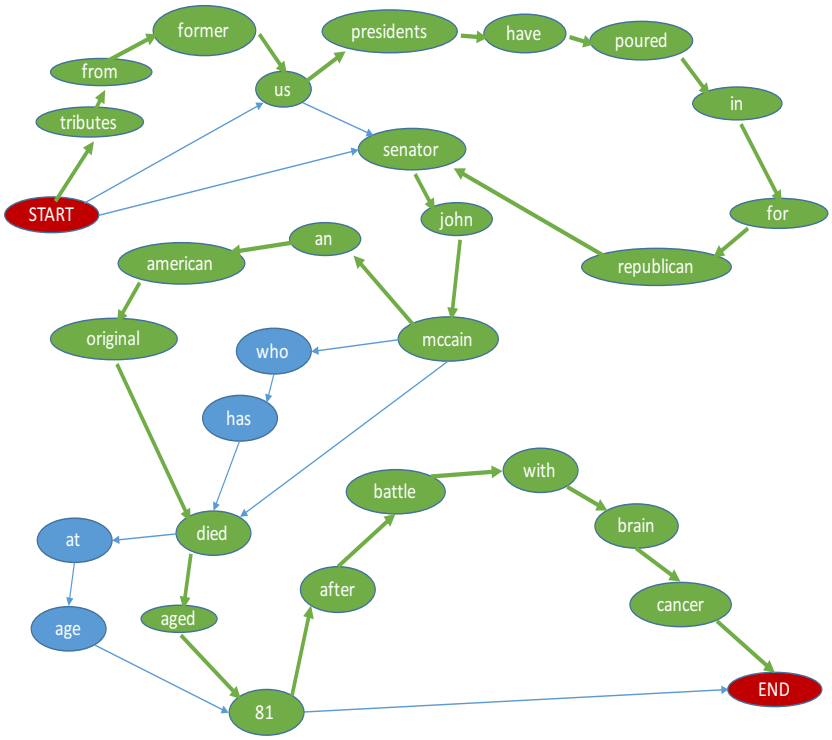


Figure 2.15: Graphical illustration of compression candidate (c) whose path is in green color, starting from node START to node END.

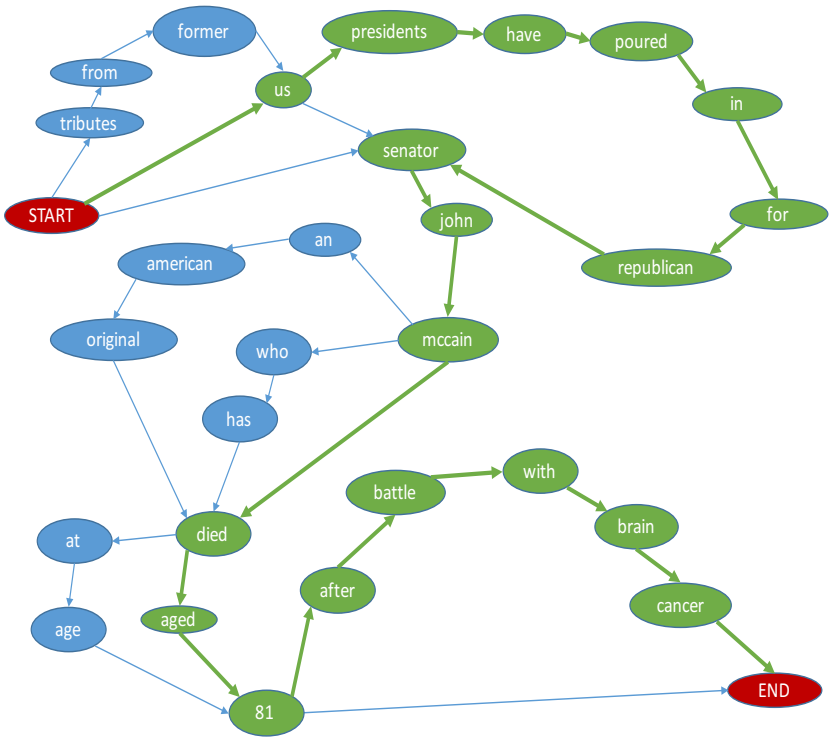


Figure 2.16: Graphical illustration of compression candidate (d) whose path is in green color, starting from node START to node END.

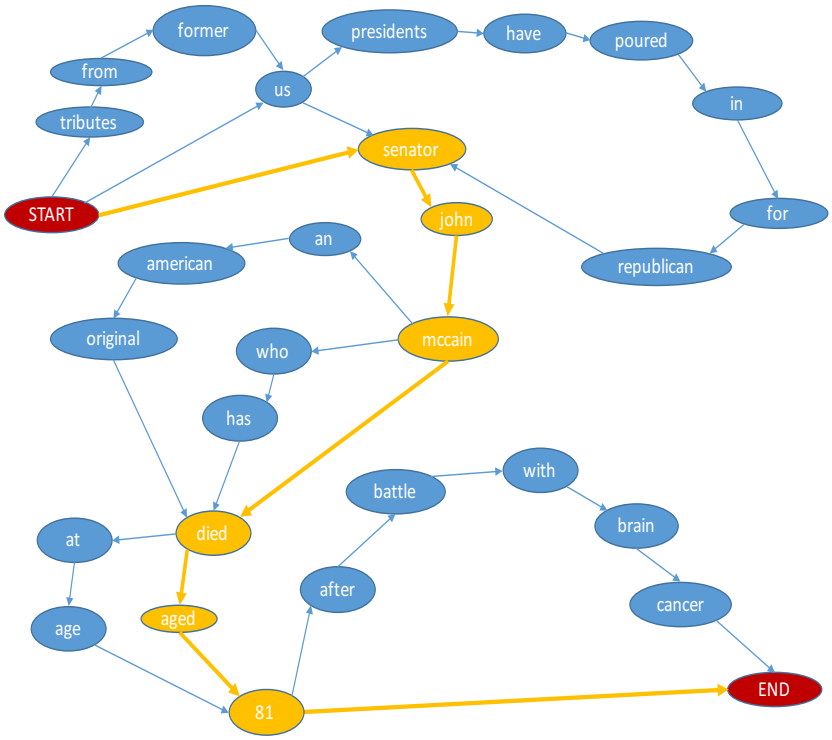


Figure 2.17: Graphical illustration of the shorest compression candidate (e) whose path is in orange color, starting from node START to node END.

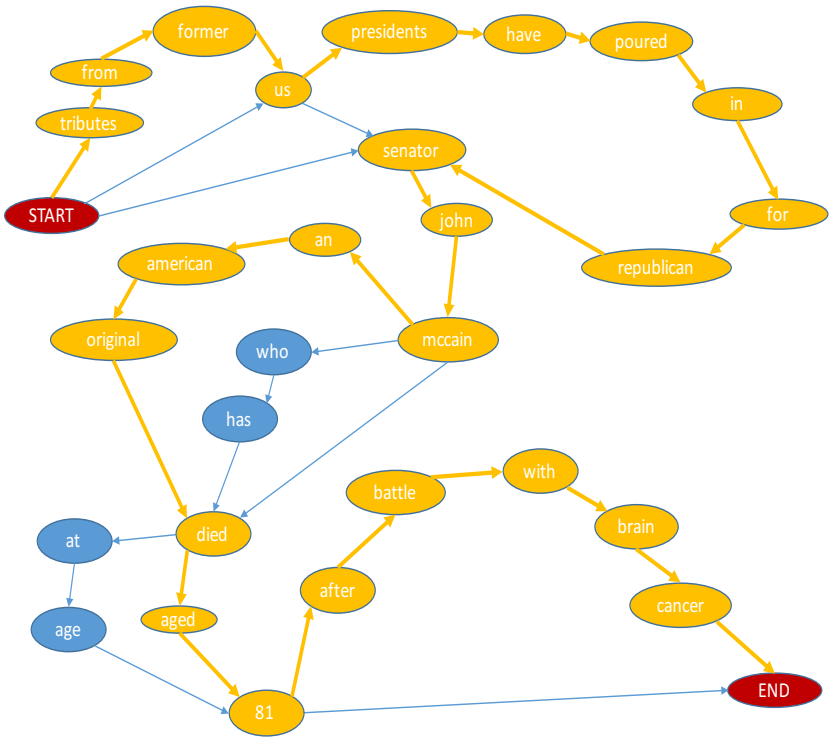


Figure 2.18: Graphical illustration of the longest compression candidate (f) whose path is in orange color, starting from node START to node END.

the cost for higher informativeness scores is a drop in readability. To simultaneously improve readability and informativeness, [6] further defined the linguistic quality (readability) and informativeness function as the constraints through an integer linear programming framework. Despite the promise brought about by the abovementioned works, the conflict between readability and informativeness still exists.

A few recent works started to consider paraphrasing at the lexical level for multi-sentence compression. With the word graph framework, [70] proposed a mapping strategy by considering synonymy between words to reduce the number of nodes in a word graph. [59] further used the Paraphrase Database (PPDB)[1] to substitute words with their paraphrasing counterparts. [60] considered word embedding to overcome the word graph drawback of two sentences possibly talking about the same topic without using any overlap words. However, 25% of the generated compression remained ungrammatical [59]. Compared to these works, our work is different for two aspects:

- First, we not only substitute the words using external knowledge but also present a rewriter to ease the conflict between the readability and informativeness.
- Second, a large-scale dataset was collected to alleviate the lack of parallel corpus, making the generation-based neural model readily applied.

[1] http://paraphrase.org/

2.4 Summary

In this chapter, we describe the research background and review representative related works of both single sentence compression and multiple sentence compression. Several challenging issues are recognized and still far from be fully solved. We state the differences of our approaches from theirs in Chapter 2.2 as well as Chapter 2.3, and will detail our approaches in next three chapters.

Chapter 3

Integration of Word-Level Linguistic Knowledge for Single Sentence Compression

Sentence compression aims to shorten a sentence into a compression while remaining grammatical and preserving the underlying meaning of the original sentence. Previous works have recognized that linguistic features such as parts-of-speech tags and dependency labels are helpful to compression generation. In this work, we introduce a gating mechanism and propose a gated neural network that selectively exploits linguistic knowledge for deletion-based sentence compression. An extensive experiment was conducted on four downstream datasets, showing that the proposed gated neural network method leads to better compression upon both automatic metrics and human evaluation, compared to previous competitive compression methods. We also observed that the generated compression by the proposed gated neural network share more grammatical relations in common with the ground-truth compression than the baseline method, indicating that important grammatical relations, such as subject or object of a sentence, are more likely to be kept in the compression by the proposed method. Furthermore, visualization analysis is conducted to explore the selective use of linguistic features, suggesting that the gate mechanism could condition the predicted compression on different linguistic features.

3.1 Introduction

Text compression and summarization aims to simplify a text while retaining its underlying meaning. It benefits lots of real-world applications such as compressing text to be displayed on small screens like a cellphone [22], compressing online reviews [2], summarizing scientific papers [46], and so forth. Among others, sentence compression is an important task where the grammar and sentence structure are compressed to generate more concise sentences, which can ideally contribute to improving automatic summarization and statistical machine translation [8]. In recent years, most of the sentence compression systems have been deletion-based; the compressions consist of subset tokens of the original sequence [37, 39, 53, 19, 8]. For example, as in Figure 3.1, if an input sentence is, *A man suffered a serious head injury after an early morning car crash today.*, an appropriate compression would be *A man suffered a serious head injury after an car crash.* In such a way, compression is generated by keeping and dropping tokens from the original input.

To avoid introducing grammatical errors in the compressions, previous work has often relied on syntactic information or syntactic features as signals [37, 53, 19, 62, 9], or on directly pruned dependency or constituency trees [39, 8, 29] to generate compressions. Recently, much attention has focused on neural-networks-

based approaches that do not require labor-intensive feature designs. Filippova et al. [28] applied long short-term memory networks (LSTMs) to sentence compression for the first time; Klerke et al. [38] further leveraged eye-movement information in multi-tasking using LSTMs and achieved encouraging results. However, linguistic knowledge could also play an important role in supervising the neural model in order to learn better feature representation, such as [28] and [3] for the problems in particular with small training data, such as [38]. Crucial to this is to the design of an effective way of integrating linguistic knowledge in neural model. To this end, we propose linguistic knowledge-enhanced gated neural networks capable of selectively utilizing linguistic features for prediction. Here, we consider three types of linguistic knowledge: dependency relations, part-of-speech (POS) tags, and named entity tags. These will be detailed later in Chapter 3.3. In short, our contributions are two-fold:

- Present a neural model augmented with a gating mechanism for sentence compression and achieve competitive results on four downstream datasets upon several evaluation metrics, compared to the comparison methods.
- Incorporate three types of linguistic knowledge into gated neural models, with the capability of selectively exploiting linguistic features by the gating mechanism. We also investigate the contribution of each linguistic feature make to the whole performance.
- Visualization analysis is also conducted for the exploration of gating mechanism, further validating the effectiveness of the proposed method.

3.2 Background

There are two research lines regarding deletion-based sentence compression. The first uses linguistic knowledge like syntactic information or syntactic features as signals [37, 53, 19, 9]. These linguistic features function as an indicator for compression. By contrast, [39, 8, 29] generate compressions directly by pruning dependency or constituency trees. Among others, [43] incorporates linguistic knowledge into a compression model, which is similar to ours. Instead of dropping or keeping words on the word level, they focus on operating on nodes in syntactic trees, yielding better sentence quality.

Another line of research has focused on neural network methods due to the advances in computational power and data amount. Such methods rely on labeled data to automatically extract features, yielding promising results. In 2015, Filippova et al. [28] applied LSTM to sentence compression in a sequence-to-sequence learning fashion for the first time. They constructed over 2 million sentence-compression pairs, yielding encouraging results. In 2016, [3] represented a neural network architecture for deletion-based sentence compression using over 2.3 million instances as [28] did. They represented word with word embedding, dependency labels, as well as part of speech tags, and sliding a window left-to-right to make decisions based on the local context. However, neither of their large deletion-based sentence compression datasets are publicly available. On the other hand, in the same year, Klerke et al. [38] leveraged eye-movement information as external knowledge in a multi-task learning fashion to achieve comparable performance. Their work implies that external knowledge usually aids sentence compression in the case of small training datasets. We follow this approach and take it as a sequence labeling task. While the previous works such as [3] and [14] also exploit dependency labels and part-of-speech tags, our work is different

from them in two ways. Firstly, we represent a neural network architecture that is able to selectively make use of different feature sources. Secondly, we further investigate the gating mechanism through visualization analysis.

3.3 Methodology

3.3.1 Linguistic Knowledge

In this work [85], three kinds of linguistic features are considered, dependency labels, part-of-speech (POS) tags, and named entity tags.

Part of speech tags

Parts-of-speech (also known as POS, word classes, or syntactic categories) are useful because knowing whether a word is a noun or an adverb contributes to making a better decisions on whether to delete or keeping a certain word in the input sentence. For example, in the sentence, "He was quite surprised by the birthday gift his parents prepared specifically for him .", the removal of adverbs "quite" and "specifically" will not affect the syntax and semantics of the original sentence if they are being removed. We expect that adding such a lexical feature can help the model to remove unnecessary tokens. Here, we firstly parse[1] the input sentences, yielding POS tags for each word, Figure 3.1 shows an example and the corresponding POS tags.

Dependency relations

Dependency parsing involves finding links between heads (also called governors) and modifiers (or dependents) with one word being the root of the sentence. Each link can be annotated with a grammatical function (dependency relations), such as nsubj short for nominal subject, or dobj short for direct object. Figure 3.1 gives an example dependency parsing tree. Each node (word) in a dependency tree will have a unique parent node except for the root node for which we assign the ROOT label. We take the modified relations labels between target word and its parent as the dependency relation labels. As shown in example sentence, "A man suffered a serious head injury after an early morning car crash today. ", the dependency labels tuple, (nsubj, ROOT, dobj) corresponds to a short sentence, (man, suffered, injury), which preserves the main structural information of the input sentence. We expect that dependency relations could enable our model to make use of structural information. The Parsey McParseface parser is used here as well.

Named entity tags

A named entity is a real-world object, such as persons, locations, organizations, etc., that can be denoted with a proper name. Here, we used three kinds of named entity types: persons, locations, and organizations. Figure 3.2 shows an example sentence, "The Columbus Blue Jackets have hired prominent player agent Bill Zito as assistant general manager. ", with two named entities, "Columbus Blue Jackets" and "Bill Zito," which are recognized by the Stanford Named Entity Recognizer[2] used in this work.

[1] We use Parsey McParseface, one of the state-of-the-art English parsers released by Google `https://github.com/tensorflow/models/tree/master/syntaxnet`

[2] https://nlp.stanford.edu/software/CRF-NER.shtml

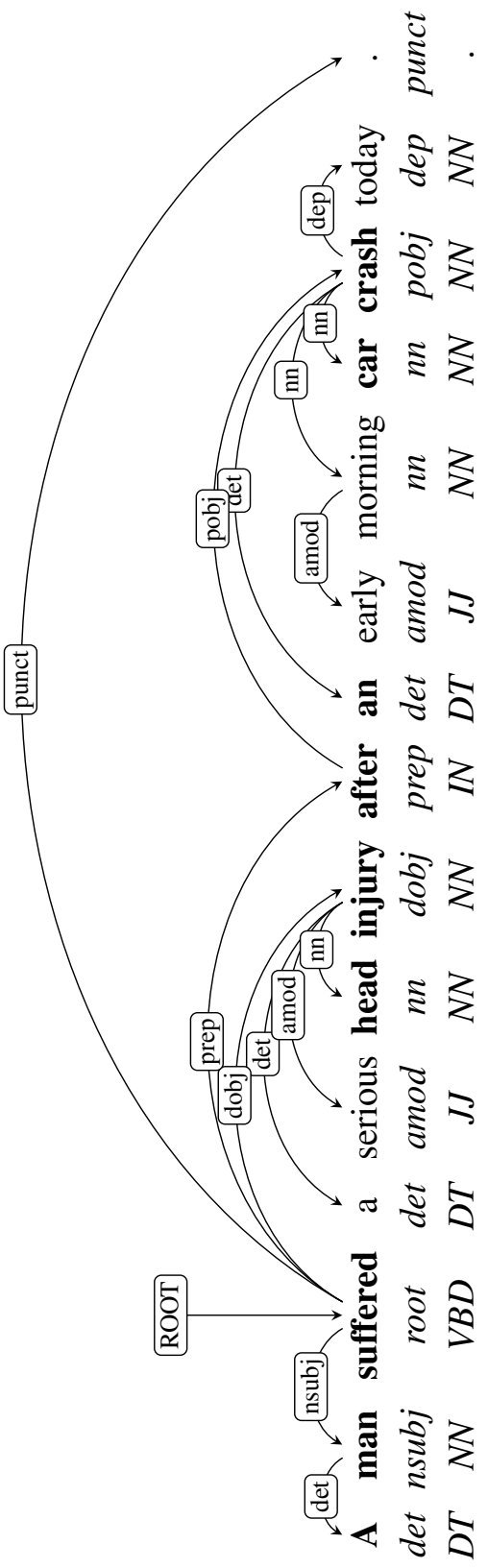


Figure 3.1: Dependency parsing result of the example sentence from GOOGLE dataset, which lies in the first row. Bold words refer to the compression, the second row consists of dependency labels for each word, while the third row consists of part-of-speech tags for each word.

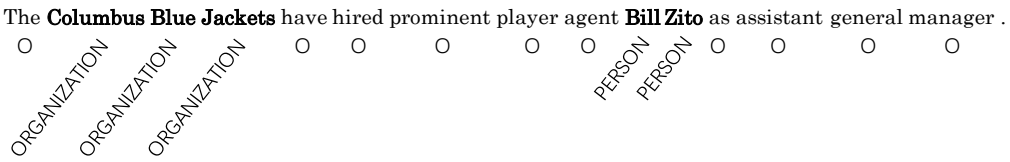


Figure 3.2: Named entity tags of the example sentence from GOOGLE dataset. Bold words refer to the named entities. The first named entity is an organization noted as "ORGANIZATION", while the second named entity is a person name noted as "PERSON".

Our motivation is that a named entity tagger should be able to tag continuous token sequences (or chunks) and, to some extent, help the model consider this continuous token sequence as a whole. Take "Columbus Blue Jackets" as an example. If the model removes the words "Columbus" and "Jackets," and only keeps the word "Blue," then it would be difficult for people to understand the meaning of the sentence. We expect that such cases will be prevented by incorporating the named entity tag feature.

3.3.2 Lookup Table

We build three lookup tables for part of speech tags, dependency relations labels, and named entity tags respectively. Each of these labels and tags corresponds to a low-dimensional real vector through their own lookup tables, as shown in Figure 3.3. We randomly initialized these lookup tables. Furthermore, similar to [28], we pre-trained word vector embedding using the skip-gram model[3] [56]. Therefore, we finally have embedding, dependency, POS, and named entity lookup tables. By virtue of these lookup tables, each token corresponds to four feature representations whose concatenation is taken as the input of neural networks.

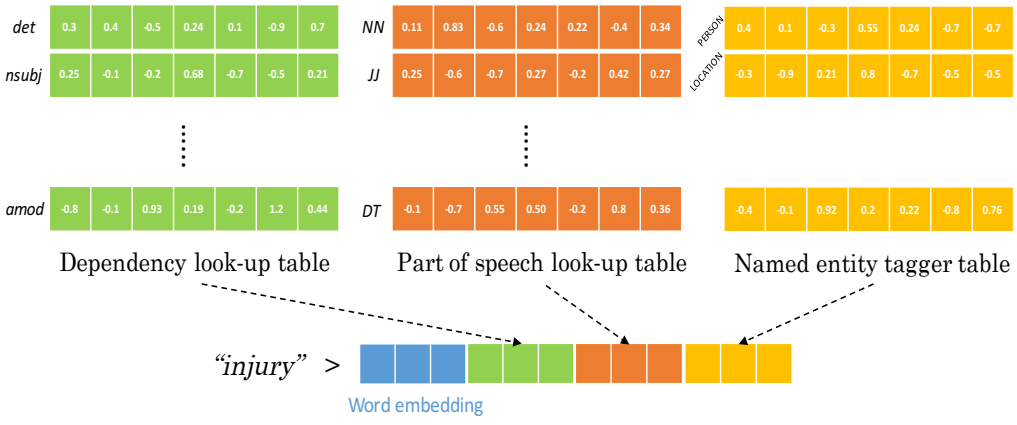


Figure 3.3: Visualization of gates.

[3] https://code.google.com/p/word2vec/

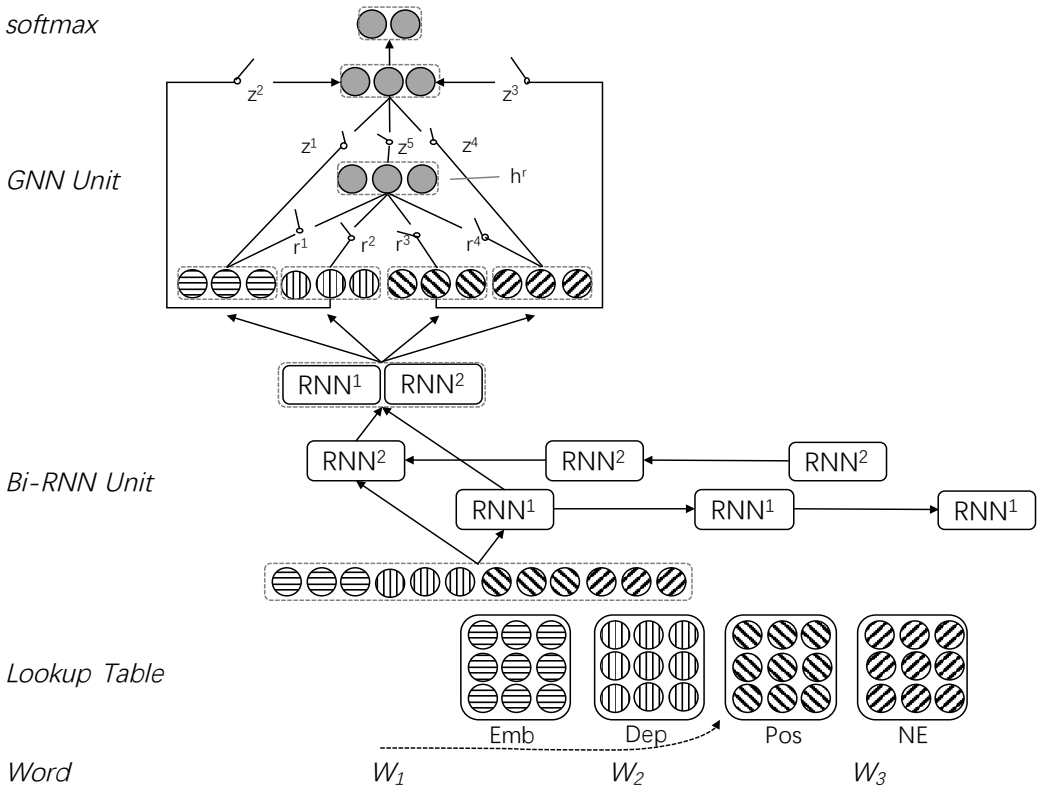


Figure 3.4: Graphical illustration of gated neural network. Emb, Dep, and Pos respectively stand for word embedding, dependency, and part-of-speech lookup tables.

3.3.3 Linguistic Knowledge Enhanced Recurrent Neural Network (LK-RNN)

The recurrent neural network (RNN) [24] is able to model sequences with arbitrary length and bi-directional RNN is even more powerful, capable of capturing both forward and backward information. Formally, the hidden layer $\mathbf{h}_i^1$ in the forward direction and the hidden layer $\mathbf{h}_i^2$ in the backward direction are

$$\mathbf{h}_i^1 = \tanh(W_1\mathbf{x}_i + U_1\mathbf{h}_{i-1}^1 + b_1), \tag{3.1}$$

$$\mathbf{h}_i^2 = \tanh(W_2\mathbf{x}_i + U_2\mathbf{h}_{i+1}^2 + b_2), \tag{3.2}$$

Where W_1, U_1, b_1 and W_2, U_2, b_2 are model parameters. As shown in Figure 3.4, we obtain $\mathbf{h}_i^{12} = [\mathbf{h}_i^1; \mathbf{h}_i^2]$ for the i-th word in input sentences. Here, [ ] means the concatenation operation of vectors. We take the concatenation of feature representations as input to a two layer bi-directional RNN (two bi-RNN units) followed by a fully connected layer. The output layer is a Softmax classifier that predicts label 1 if a word is to be retained in the compression or label 0 if a word is to be dropped.

3.3.4 Linguistic Knowledge Enhanced Gated Neural Network (LK-GNN)

On the top of LK-RNN, we add a gated neural network (GNN) unit taking the hidden states of the bi-directional RNN as the inputs as shown in Figure 3.4.

The added GNN is capable of fusing each feature to yield a combination in the first place.

$$\mathbf{r}_1 = \text{sigmoid}(W_3[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{emb}}] + b_3), \tag{3.3}$$

$$\mathbf{r}_2 = \text{sigmoid}(W_4[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{dep}}] + b_4), \tag{3.4}$$

$$\mathbf{r}_3 = \text{sigmoid}(W_5[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{pos}}] + b_5), \tag{3.5}$$

$$\mathbf{r}_4 = \text{sigmoid}(W_6[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{ne}}] + b_6), \tag{3.6}$$

Here, the contextual information from the bi-directional RNN, $\mathbf{h}_i^{12}$, can be seen as a conditioning signal for gates. Similar to candidate activation in a GRU unit [15], the combination of three feature inputs, $\mathbf{h}_i^r$, is yielded through element-wise multiplication $\odot$.

$$\mathbf{h}_i^r = \mathbf{r}_1 \odot \mathbf{x}_i^{\text{emb}} + \mathbf{r}_2 \odot \mathbf{x}_i^{\text{dep}} + \mathbf{r}_3 \odot \mathbf{x}_i^{\text{pos}} + \mathbf{r}_4 \odot \mathbf{x}_i^{\text{ne}}. \tag{3.7}$$

Then, update gates $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$, $\mathbf{z}_4$ and $\mathbf{z}_5$ are used to yield linear interpolation between three feature inputs and their combination, $\mathbf{h}_i^z$.

$$\mathbf{z}_1 = \exp(W_7[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{emb}}] + b_7), \tag{3.8}$$

$$\mathbf{z}_2 = \exp(W_8[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{dep}}] + b_8), \tag{3.9}$$

$$\mathbf{z}_3 = \exp(W_9[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{pos}}] + b_9), \tag{3.10}$$

$$\mathbf{z}_4 = \exp(W_{10}[\mathbf{h}_i^{12}; \mathbf{x}_i^{\text{ne}}] + b_{10}), \tag{3.11}$$

$$\mathbf{z}_5 = \exp(W_{11}[\mathbf{h}_i^{12}; \mathbf{h}_i^r] + b_{11}), \tag{3.12}$$

Each $\mathbf{z}_i$ is normalized by their sum $\sum_i \mathbf{z}_i$. Finally,

$$\mathbf{h}_{\text{i}}^{\text{z}} = \mathbf{z}_1 \odot \mathbf{x}_{\text{emb}} + \mathbf{z}_2 \odot \mathbf{x}_{\text{dep}} + \mathbf{z}_3 \odot \mathbf{x}_{\text{pos}} + \mathbf{z}_4 \odot \mathbf{x}_{\text{ne}} + \mathbf{z}_5 \odot \mathbf{h}_{\text{i}}^{\text{r}}. \tag{3.13}$$

Here, $\mathbf{h}_i^z$ is fed into the Softmax layer to make a binary prediction (REMOVE or KEEP). $\mathbf{x}^{\text{emb}}$, $\mathbf{x}^{\text{dep}}$, $\mathbf{x}^{\text{pos}}$, $\mathbf{x}^{\text{ne}}$, W_i, b_i (i ranges from 1 to 10), U_1, U_2 and W_{11}, b_{11} (for the Softmax layer) are parameters to be learned during training. Models are trained to minimize a cross-entropy loss function, with a L_2 regularization term.

$$\text{L}(\theta) = -\sum_{\text{w}\in\text{S}} \sum_{\text{i}=1}^{\text{C}} \text{P}_i^{'} \log \text{P}_i + \frac{\lambda}{2} \parallel \theta \parallel^2, \tag{3.14}$$

where θ is the set of model parameters, w refers to a word, S refers to a sentence, C is the number of classes (here, C=2) and P_i is the predicted probability. $P_i^{'}$ has a 1-of-C coding scheme where C equals the number of classes, and the dimension corresponding to the ground truth is 1, with all others being 0. λ is the regularization hyper-parameter.

3.4 Experiment

3.4.1 Datasets

To evaluate the proposed models, we used three popular downstream datasets in sentence compression research.

GOOGLE dataset

In 2015, Filippova et al. [28] created a large-scale parallel sentence compression dataset. For some reason, only a small part (subset), 10,000 out of the total number of entries is publicly available, which we refer to as the GOOGLE dataset in this work. The compression rate of this subset is 0.43[4] (Note that unlike deletion rate, the compression rate refers to the length of compression divided by the input sentence length).

Clarke-News dataset

The Clarke-News dataset [17] consists of two corpora that belong to the same genre (news) but to different domains (written and spoken). The spoken corpus consists of 1,370 sentences from the English Broadcast News corpus, with manually-generated compression. This corpus contains broadcast news stories from a variety of networks (CNN, ABC, CSPAN and NPR) which have been manually transcribed with compression rate 0.73. The spoken corpus has three annotators, which lead to three datasets with same input sentence but different compression, namely, spoken1, spoken2, and spoken3. The second corpus (written) consists of news articles gathered from the BNC and the American News Text corpus. It contains 1,629 (sentence, human compression) pairs[5] and the corresponding compression rate is 0.71.

Ziff-Davis dataset

The Ziff-Davis corpus [40] originates from a collection of news articles on computer products and contains 1,087 sentence compression pairs with 0.56 compression rate. This could be the earliest sentence compression parallel corpus. For comparison purposes, we used the same testing sets as in previous studies [28, 38]. Finally, the partitions of the training, development, and testing sets are respectively 8,000/1,000/1,000 for the GOOGLE dataset, 882/78/410 for the spoken dataset, 1,104/63/462 for written dataset, and 1,023/32/32 for the Ziff-Davis dataset.

3.4.2 Comparison methods

Here, we detail several strong comparison methods in sentence compression research.

Conditional Random Fields

Conditional random fields (CRFs) [41] are a framework for building probabilistic models to segment and label sequence data. Different from other methods like

[4]To avoid a misunderstanding, it is worth noting that in our Natural Language & Information Systems work [83] , we reported compression rate in a micro-averaging fashion, while in this work we used macro-averaging method as consistent with previous studies.

[5]We found that the total number is a bit different when we processed this corpus.

the hidden Markov model, CRFs can relax strong independence assumptions [61], making them more suitable for context-dependent problems such as POS tagging. Here, we considered four types of features as input: unigrams, POS tags, dependency relation labels, and named entity tags. The task is to predict label 1 or label 0. CRF model implementation is based on the CRFsuite[6].

Integer linear programming

Integer linear programming (ILP) has attracted much attention in the natural language processing community. ILP techniques have been applied to several tasks, including machine translation [34], syntactic parsing [65], and multi-document summarization [54]. In 2008, Clarke and Lapata employed ILP for sentence compression. The motivation is that ILP is capable of capturing the global properties or long-range dependencies of a problem, thus being able to perform decisions based on evidence beyond the local scope (i.e., beyond adjacent words or POS). For example, compression length is a global feature rather than a local one. Only by considering all tokens as a whole in the input sentence can we control the length property of compression. It is noteworthy that this method relies heavily on syntactic structures in an unsupervised fashion, depending on the choice of compression rate and (pretrained) language model[7]. A trigram language model is used as it takes the context information into account, helping to generate more grammatical sentences. More specifically, similar to [19], the tri-gram language model constrain is as follow, with encouraging the reasonable trigram collocation.

$$\max \mathrm{Y} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}} \cdot \mathrm{P}(\mathrm{w_i}|\mathrm{BOS}) + \sum_{\mathrm{i}=1}^{\mathrm{n}-2} \sum_{\mathrm{j}=\mathrm{i}+1}^{\mathrm{n}-1} \sum_{\mathrm{k}=\mathrm{j}+1}^{\mathrm{n}} \gamma_{\mathrm{ijk}} \cdot \mathrm{P}(\mathrm{w_k}|\mathrm{w_i}, \mathrm{w_j}) + \sum_{\mathrm{i}=0}^{\mathrm{n}-1} \sum_{\mathrm{j}=\mathrm{i}+1}^{\mathrm{n}} \beta_{\mathrm{ij}} \cdot \mathrm{P}(\mathrm{EOS}|\mathrm{w_i}, \mathrm{w_j}), \tag{3.15}$$

Where w_i is the word token in the original sentence. *BOS* and *EOS* are begin and end token respectively. α_i, $\beta_{i,j}$, and γ_{ijk} are subject to 0 or 1.

Sequence-to-sequence learning with attention

Sequence-to-sequence leaning framework with attention mechanism [15] show promises in lots of NLP tasks in particular in language generation problem such as machine translation, image caption, abstractive summarization, etc. The architecture consists of two parts; encoder part is a recurrent neural network with gated recurrent units that has been proved to be able to track long-term dependencies effectively. Furthermore, bi-directional RNN is even more powerful, capable of capturing both forward and backward information. Here, we also consider dependency label features $\mathbf{x}_i^{dep}$, part-of-speech tag feature $\mathbf{x}_i^{pos}$, and named entity tag feature $\mathbf{x}_i^{ne}$. Formally, the hidden layer $\mathbf{h}_i^f$ in the forward direction and the hidden layer $\mathbf{h}_i^b$ in the backward direction are

$$\mathbf{h}_{\mathrm{i}}^{\mathrm{f}} = \mathrm{GRU}^{\mathrm{f}}([\mathbf{x}_{\mathrm{i}}^{\mathrm{emb}}; \mathbf{x}_{\mathrm{i}}^{\mathrm{dep}}; \mathbf{x}_{\mathrm{i}}^{\mathrm{pos}}; \mathbf{x}_{\mathrm{i}}^{\mathrm{ne}}], \mathbf{h}_{\mathrm{i}-1}^{\mathrm{f}}), \tag{3.16}$$

$$\mathbf{h}_{\mathrm{i}}^{\mathrm{b}} = \mathrm{GRU}^{\mathrm{b}}([\mathbf{x}_{\mathrm{i}}^{\mathrm{emb}}; \mathbf{x}_{\mathrm{i}}^{\mathrm{dep}}; \mathbf{x}_{\mathrm{i}}^{\mathrm{pos}}; \mathbf{x}_{\mathrm{i}}^{\mathrm{ne}}], \mathbf{h}_{\mathrm{i}+1}^{\mathrm{b}}), \tag{3.17}$$

[6] https://pypi.python.org/pypi/python-crfsuite

[7] Our implementation is based on: https://github.com/cnap/sentence-compression

where $\mathbf{x}_i^{emb}$ is word embedding of input $\mathbf{x}_i$. we obtain $\mathbf{h}_i^{fb} = [\mathbf{h}_i^f; \mathbf{h}_i^b]$ for the i-th word in input sentences. Here, [ ] means the concatenation operation of vectors. We take the final hidden state $[\mathbf{h}_n^f; \mathbf{h}_1^b]$ as the initial hidden state for the decoder that consists of unidirectional recurrent neural network with gated recurrent units. However, one flaw of vanilla sequence to sequence learning is that model is unable to focus on different parts of the input in each time step. Thus, attention mechanisms [5, 49] are proposed to tackle this issue. We performed the keep or remove decision by pay attention to the different context information. In this work, Bahdanau's attention mechanism [5] is used.

$$\alpha_{\mathrm{t}} = \mathrm{V}^{\mathrm{T}} \tanh(\mathrm{W}\mathbf{s}_{\mathrm{t}}^{\mathrm{decoder}} + \mathrm{U}\mathbf{h}_{\mathrm{encoder}}^{\mathrm{fb}}), \tag{3.18}$$

$$\alpha_{\mathrm{t}} = \frac{\exp(\alpha_{\mathrm{t}})}{\sum_{\mathrm{i=1}}^{\mathrm{k}} \exp(\alpha_{\mathrm{i}})}, \mathbf{c}_{\mathrm{t}} = \sum_{\mathrm{i=1}}^{\mathrm{k}} \alpha_{\mathrm{i}} \mathbf{h}_{\mathrm{i}}^{\mathrm{fb}}, \tag{3.19}$$

$$\mathbf{s}_{\mathrm{t}}^{\mathrm{decoder}} = \mathrm{GRU}^{\mathrm{decoder}}(\mathbf{s}_{\mathrm{t-1}}, \mathbf{c}_{\mathrm{t-1}}, \mathbf{e}(\mathrm{y}_{\mathrm{t-1}})), \tag{3.20}$$

where V, W, U are parameters to be learned. $\mathbf{s}^{decoder}$ is the hidden state of decoder, followed by a Softmax layer for prediction. α_t is the attention distribution; $\mathbf{c}_t$ is the context information from encoder. $\mathbf{e}(y_{t-1})$ is the embedding (vector representation) of the previous output y_{t-1} in the decoder.

Stacked Long Short-Term Memory Networks

Vertically stacked long short-term memory networks layers allow for greater model representation ability, thus setting a relatively strong baseline for sentence compression task [28, 38]. In general, the hierarchy of hidden layers enables more complex representation of sequential data, capturing information at different scales. Therefore, we take it as the baseline method in this work. It is worth pointing out that we also implemented the linguistic knowledge-enhanced LSTMs as described in [76], which also takes the dependency label and part-of-speech tags as the inputs[8] for a comprehensive comparison.

3.4.3 Training details

LSTM [35] has shown promises in many natural language processing (NLP) tasks because of its ability to overcome the vanishing gradient problem, thus retaining long-distance dependency. Filippova et al. [28] and Klerke et al. [38] applied three-layer bi-LSTM to the sentence compression problem, achieving very competitive performance. As such, we implemented this three-layer bi-LSTM as the baseline. Furthermore, we implemented a sequence-to-sequence (seq2seq) framework whose encoder and decoder are both LSTM, since LSTM networks show promising performance in a variety of NLP tasks. As for the parameter settings, the dimension of each input feature was set to 50. The number of dimensions of hidden layer dimensions was 150 in the case of GOOGLE. For other datasets, this number of dimensions will be decreased accordingly due to the relatively small size of the training corpus.

The models were trained by using the stochastic gradient descent and regularization hyper-parameter λ equals 10^{-4}. The iterations stopped when the

[8] On top of input features, their LSTMs structure is a slightly different from us; we concatenate the outputs from the previous layer in both forward and backward direction as the input for the following layer, while they concatenate the outputs in both directions only from the last layer of LSTMs.

accuracy of the development set no longer increased for a period of time. All the experiments were conducted using the Theano (version 0.8.2) deep learning framework. We found that there were 45 distinct dependency labels, at most 38 distinct POS tags, and three named entity tags in the training datasets. For the ILP method, the hyperparameter compression rate changed according to different corpora.

3.5 Evaluation and Analysis

3.5.1 Quantitative Evaluation

We adopt the token-level F_1 score to calculate whether an individual word was correctly removed or kept in the sentence, compared with the ground-truth compression. Table 3.1 presents F_1 score results. Here are our observations:

(1) the proposed models (LK-GNN and LK-RNN) led to improvements over other comparison methods across all datasets, suggesting that the introduction of linguistic knowledge did help the models to exploit syntactic information to determine which words should be kept or dropped. The LK-GNN achieved better or comparable performance over the baseline (#6 and #7), validating the effectiveness of the proposed method.

(2) with respect to two proposed models, the LK-GNN model show improvements over the LK-RNN model (although not significantly) across all datasets (#7 vs #8). This may be due to that the gating-mechanism-based LK-GNN is capable of selectively exploiting features according to different target words in the sentence, dynamically combining features for better compression prediction.

(3) The performance of the LSTM+SynFeature model is quite comparable to the performance of the baseline model (#5 vs #6). Meanwhile, RNN-based model seems better than LSTM-based model (#6 vs #7). We speculate that dataset size could possibly be the cause since the most datasets are limited at size, thus leading to the lower results for LSTMs compared to RNNs.

(4) the widely used sequence labeling CRF method seems to perform poorly in decision-making (keeping or removing), even if we incorporated unigrams, dependency labels, POS tags, and named entity tags as features. This may be because CRFs lack the ability to capture long-term dependency between words. In contrast, seq2seq learning with the attention model was able to maintain long-term dependency using gated recurrent units (GRUs), thus achieving better results (F_1 score).

(5) the traditional ILP exhibited poor performance, which was also observed by [76]. It should be noted that the ILP method in this work was unsupervised, and thus relied heavily on recourse that we exploited. A pretrained language model (we chose a news-based one) was one of the important external recourses. In the future, we will attempt to investigate the effects of external recourses, such as language models, on ILP performance.

To further investigate the contributions of each linguistic feature, we conducted experiments using different combinations of features, as listed in Table 3.2. The results show that the combination of all the features performed the best, suggesting that all of the features contributed to the final prediction. Adding dependency labels also seems to provide some improvements over the model using only word embedding, although not significantly. This is also true for the POS feature. These results imply that the dependency label and POS features are relatively independent. This is consistent with the fact that a dependency parse connects words according to their dependency relationships, while a constituency

Model	GOOGLE	Written
#1 CRFs	72.1*†	68.2*†
#2 ILP (Clarke, 08)	56.0*†	63.5*†
#3 Seq2seq with att (Cho, 15)	74.8*†	64.2*†
#4 LSTMs+eye (Klerke, 16)	81.0*	–
#5 LSTM+SynFeature (Wang, 17)	80.1*	68.0*†
#6 Stack-LSTMs (Baseline)	79.9*†	70.3
#7 Our LK-RNN	82.3†	71.4†
#8 Our LK-GNN	**82.7***	**72.0***
Model	**Ziff-davis**	**Spoken1**
#1 CRFs	63.5*†	72.1*†
#2 ILP (Clarke, 08)	54.9*†	61.8*†
#3 Seq2seq with att (Cho, 15)	62.6*†	73.2*†
#4 LSTMs+eye (Klerke, 16)	**74.2**	75.2
#5 LSTM+SynFeature (Wang, 17)	66.3*†	74.5†
#6 Stack-LSTMs (Baseline)	67.4*†	73.9*†
#7 Our LK-RNN	70.3†	75.8*
#8 Our LK-GNN	70.5*	**76.4†**
Model	**Spoken2**	**Spoken3**
#1 CRFs	77.0*†	66.0*†
#2 ILP (Clarke, 08)	68.1*†	59.3*†
#3 Seq2seq with att (Cho, 15)	79.9*†	67.5*†
#4 LSTMs+eye (Klerke, 16)	**82.2**	70.1†
#5 LSTM+SynFeature (Wang, 17)	80.5	67.8*†
#6 Stack-LSTMs (Baseline)	79.9*†	68.3*†
#7 Our LK-RNN	82.1*	71.7*
#8 Our LK-GNN	82.0†	**72.3†**

Table 3.1: F_1 Results. * and † respectively stand for significant difference with 0.95 confidence between results yielded by our methods and results yielded by comparison methods in the same column. "−" stands for too low results or no data found. Best results are in bold font.

parse aims to use subphrases in text to form a tree. However, named entity features do not seem to bring much improvement, which can be observed across most datasets. This could be attributed to two possible reasons. Firstly, this feature may be not quite discriminative for whether a word should be removed or kept as other features did. Secondly, this feature may not be independent as part of speech feature contains this named entities feature in some cases. For example, with respect to the phrase *Hong Kong*, the POS tags are (*PROPN*, *PROPN*) while the named entity tags are (*LOCATION*, *LOCATION*). we found that for the GOOGLE and spoken corpuses, the GNN without named entity feature yields even better results (we removed this feature from the experiments accordingly). Therefore, adding the named entity feature may not be helpful to the performance.

Parsing-based Evaluation

A parsing-based evaluation measure was proposed by [66]. Similar to the previous

Feature	GOOGLE	Written	Ziff-davis
Emb	80.1	70.3	63.3
Emb + Dep	81.9	71.2	66.7
Emb + Pos	81.5	71.3	69.7
Emb + Ne	81.0	70.3	63.8
All features	82.3	71.4	70.3
Feature	**Spoken1**	**Spoken2**	**Spoken3**
Emb	74.0	80.4	70.9
Emb + Dep	74.1	81.6	71.4
Emb + Pos	74.7	81.6	71.2
Emb + Ne	74.2	80.9	70.7
All features	75.8	82.1	71.7

Table 3.2: F_1 Results. "Emb" means using the word embedding feature only; "Emb+Dep" means using both word embedding and dependency label as features; "Emb+Pos" means using word embedding and part-of-speech tags as features; "Emb+Ne" means using word embedding and named entity tags as features;"All features" is the our LK-RNN model that uses all the word embedding, dependency label, part-of-speech tags, and named entity tags.

study, we use the robust accurate statistical parser (RASP)[9] [12], a domain-independent, robust parsing system for English. This parsing-based evaluation compared grammatical relations (such as ncsubj and dobj) found in the system compressions with those found in a ground truth, providing a means to measure the semantic aspects of compression quality, as syntactic parse trees is able to reflect semantic properties to some extend [32] . According to [18], this measure reliably correlates with human judgements. For example, assume that our gold standard compression is "Aaron Donald won the 2013 Bronko Nagurski Trophy," while the system output is "Aaron Donald won."

Gold	**Aaron donald won the 2013 bronko nagurski trophy .**
1	(ncsubj, win+*ed* : 3_VVD, donald: 2_$NP1$)
2	(dobj, win+*ed* : 3_VVD, trophy: 8_$NN1$)
3	(det, trophy: 8_$NN1$, the: 4_AT)
4	(ncmod, trophy: 8_$NN1$, 2013: 5_MC)
5	(ncmod, trophy: 8_$NN1$, bronko: 6_JJ)
6	(ncmod, trophy: 8_$NN1$, nagurski: 7_JJ)
7	(ncmod, donald: 2_$NP1$, Aaron: 1_$NP1$)
System	**Aaron donald won .**
1	(ncsubj, win+*ed* : 3_VVD, *donald* : 2_$NP1$)
2	(ncmod, donald: 2_$NP1$, *Aaron* : 1_$NP1$)

Table 3.3: Robust Accurate Statistical Parsing (RASP) results of sentence in GOOGLE dataset.

As shown in Table 3.3, the system output is fine in terms of readability but is semantically unclear because of the lack of object (dobj in Table 3.3). By using RASP, we were able to determine whether the compression captured the gram-

[9]http://users.sussex.ac.uk/ johnca/rasp/

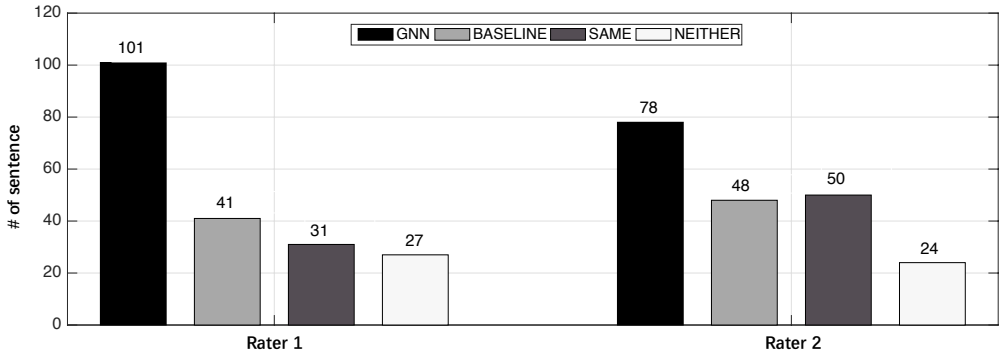


Figure 3.5: Human evaluation results of two native English raters. The number on top of each bar means the number of sentences.

matical relation captured by the ground-true compression. Thus, this measure is complementary to previous measures in terms of semantics. Following previous works, we used Precision, Recall, and F-Score.

Model	**Precision**	**Recall**	**F-Score**
Stacked-LSTMs (Baseline)	77.3	67.7	72.2
GNN	80.2	70.5	75.0

Table 3.4: RASP-based Precision, Recall, and F-score results based on 200 compressions in GOOGLE dataset.

We applied this evaluation measure to compressions generated by the baseline (Stacked-LSTMs), GNN, and the ground-truth compression. As shown in Table 3.5, the proposed GNN outperforms the baseline model based on the micro-average of Precision, Recall, and F-Score, suggesting that our model is able to retain more grammatical relations associated with the ground-truth compression. This also indicates that important grammatical relations, such as subject or object, are more likely to be kept in the compression by the proposed GNN, leading to the improvement of both syntactic and semantic aspects of compression.

3.5.2 Human Evaluation

We conducted a human evaluation where the compressions (of the first 200 sentences in the GOOGLE dataset) generated by LK-GNN and the baseline (Stacked-LSTMs) method were shown to two native English speakers. Subjects were asked to rate them in terms of grammatical correctness without being shown the original sentences. Then, they were to assign one label out of GNN, BASELINE, BOTH, NEITHER to each pair of sentences, according to the criteria above. Here, GNN means that the compression generated by LK-GNN was better compared to BASELINE, and vice versa. BOTH means that the compressions generated by the two models were the same or nearly the same, and NEITHER means that neither made sense or neither was complete. The results, which are summarized in Figure 3.5, show that, on average, in 45% of cases (90/200), the proposed LK-GNN performed better than the baseline in terms of syntactically correctness, while in 22.5% of cases (45/200), the baseline performed better. Also, we observed that, in 20% of cases (40/200), the two models performed nearly equally. Both failed in only 12.5% of cases (25/200) where both fail. To assess the inter-assessor agreements, we computed Cohenś unweighted κ [20]. The

computed unweighted κ was 0.485, resulting in a moderate agreement level [10]. We therefore conclude that the linguistic knowledge supervised neural network is more likely to generate more grammatically correct sentences that make sense.

3.5.3 Case Study

We herein investigated some example cases in Table 3.5, which shows that by incorporating syntactic information, gated neural networks performed better in some cases such as in compressing less important phrases and clauses such as (a1 v.s. c1 and a2 v.s. c2), but suffer to some extend from be ungrammatical in other cases (such as c3) compared with the baseline model. Also, as some examples in Spoken Dataset show, the proposed GNN can generate readable compression (c4), although the generated compression is not always satisfactory (c6), compared with the baseline model. We believe the quality of compression could possibly be further improved by incorporating a language model into compression output. We leave this as the future work.

a.Original sentence	**b.**Output of LSTMs baseline	**c.**Output of GNN
a1:	A woman accused of beating her poodle puppy to death and then burying it in her backyard was allowed to go free Thursday .	
b1:	A woman accused of beating her poodle puppy to death .	
c1:	A woman accused of beating her puppy to death was allowed to go free .	
a2:	A man who was shot in the head in front of his teammates as he walked off a soccer pitch knew his life was in danger , an inquest heard .	
b2:	A man who was shot off a knew life was .	
c2:	A man who was shot knew his life was in danger .	
a3	BWV group middle east has confirmed that the new bmw 5 series will go on sale in the middle east in september .	
b3	New bmw 5 series will go on sale .	
c3	The new bmw 5 series will go on sale east .	
a4:	Against this background , it is somewhat surprising that EPA would , on its own initiative, undertake a major new program as part of its brownfields action agenda .	
b4:	would , on its own initiative , undertake a major new program as part of its brownfields action agenda.	
c4:	It is surprising EPA would on initiative , undertake a major new program as part of its brownfields action agenda.	
a5:	I think we have heard from a very small minority , and unfortunately , they are bullies .	
b5:	think we have heard from a very small minority , and unfortunately , they are bullies .	
c5:	e have heard from a minority and , they are bullies .	
a6:	He has a plan where you go , what to do , what to talk about .	
b6 :	He has a plan where you go , what to do , what to talk about .	
c6 :	He has a plan where what to do , what to talk about .	

Table 3.5: Example sentences and its compression results.

[10]Landis and Koch [42] characterize κ values < 0 as no agreement, 0 – 0.20 as slight, 0.21 – 0.40 as fair, 0.41 – 0.60 as moderate, 0.61 – 0.80 as substantial, and 0.81–1 as almost perfect agreement.

3.5.4 Visualization of Gating Mechanism

To further investigate how the gating mechanism adaptively makes use of different linguistic features, we visualized the gates $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$ and $\mathbf{z}_5$ using a color gradient map (because our best model only involves embedding, dependency, and POS features,). Each gate was actually a vector and each dimension of this vector corresponds to a real value between 0.0 and 1.0. Gates were normalized by their sum, thus satisfying

$$\mathbf{z}_1 + \mathbf{z}_2 + \mathbf{z}_3 + \mathbf{z}_5 = \mathbf{1}. \tag{3.21}$$

All gates depended on the contextual information $\mathbf{h}_i^{12}$ of the current word. On top on that, $\mathbf{z}_1$ is further depended on word embedding; $\mathbf{z}_2$ is further depended on dependency label embedding; $\mathbf{z}_3$ is further depended on part of speech embedding; $\mathbf{z}_5$ is further depended on the fused embedding of word, dependency label, and part of speech embedding. Similar to the GRU unit [15], each gate was thought to be able to control the flow of information and we accordingly assume that the bigger the value of a gate is, the more information flow could go through that gate. For instance, if the dependency gate $\mathbf{z}_2$ had a bigger value on each of its dimensions, it would mean that the dependency label embeddings made more of a contribution to the final prediction.

Furthermore, since each gate is a vector, we average all dimensions of a gate vector for simplicity and took its mean as the representative to draw the color gradient map. Thus, each gate vector was turned into a real value (scalar) between 0 and 1.

Figure 3.6 shows three example sentences processed by our LK-GNN model: (a), (b), and (c). Emb, Dep, Pos, and Cadi respectively refer to word embedding gate $\mathbf{z}_1$, dependency label embedding gate $\mathbf{z}_2$, part of speech embedding gate $\mathbf{z}_3$, and fused embedding gate $\mathbf{z}_5$. The redder the corresponding box gets, the more information that flows through the gate; the less the information that flows through the gate, the more green the box gets. In all example sentences, dependency gate $\mathbf{z}_2$ allows more information to go through it, suggesting that dependency information may play an important role in the decision to keep (words with an underline) or drop (word without underline), we did not do statistical significance tests though. To be more specific, for case (b) and (c), the words modified by (nsubj, ROOT, and dobj) were respectively (*Beatrix*, *undergone*, *surgery*) and (*google*, *introduced*, *bean*), which can be viewed as the core of those sentences. This implies that the model learned that these dependency labels are crucial for sentence compression by having more information flow go through that gate. On the other hand, word *free* in (a) made more use of the part-of-speech gate to decide whether it should be kept or not. This means that the proposed model seemed to consider different feature sources to make the best prediction. However, a further and more exact explanation would require more experiments in future.

As a matter of fact, the gating mechanism and attention mechanism share a lot in common. For instance, they both allocate different weights to different information sources, allowing neural network models to focus on the information important to the prediction. However, it is worth pointing out the difference between both. The attention mechanism was originally developed to solve the word alignment problem in neural machine translation. To condition words in a decoder on more relevant hidden states in an encoder, attention weights are used to be as relevant coefficients for each hidden states in the encoder. Its weight is

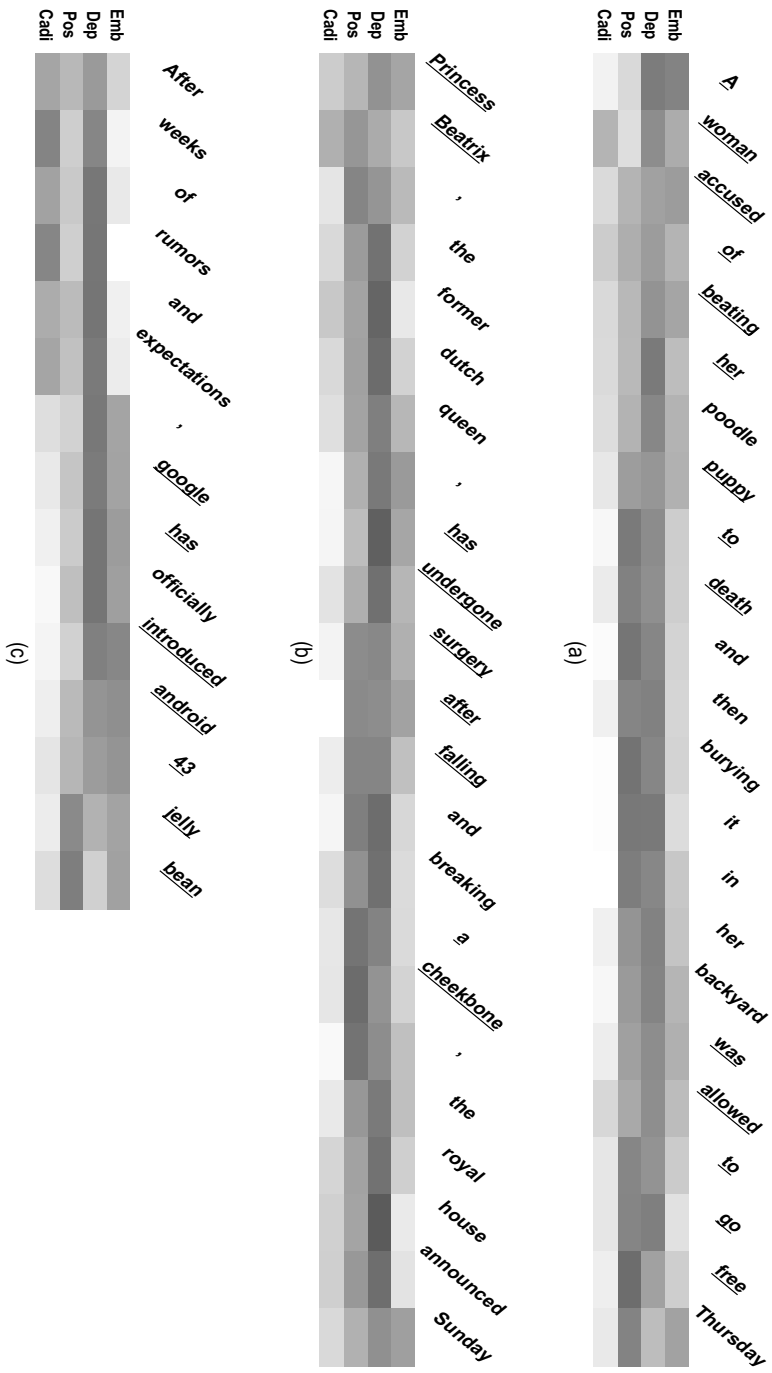


Figure 3.6: Emb, Dep, Pos, and Cadi (short for candidate) respectively refers to $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$ and $\mathbf{z}_5$ of our proposed LK-GNN model. The darker each gate is, the more information that flows through each gate; the lighter each gate is, the less information flows through each gate. Words kept by LK-GNN are underlined.

usually a scalar (0-dimensional tensor) in many works such as [5], while it could be a vector (1-dimensional tensor) in a few works such as [81]. In contrast, a gate is a vector capable of dynamically controlling the neural network architecture in that gates seem to change the direction in which the information flow goes by opening or closing the gates. Here, element-wise product operation plays an important role. More importantly, due to the introduction of gates, the neural network has both control information (gate) and data information (such as a neural network layer), which make it a bit different from the commonly used McCulloch-Pitts neural network [52]. Our experimental results also show that through the neural network learning (training), a gate, as a controller of information, is capable of conditioning the prediction on different information sources (inputs), achieving a dynamic network architecture in a sense.

3.5.5 Analysis of Gating Mechanism

Despite the qualitative analysis of the three example sentences above, we are more interested how the gating mechanism performs quantitatively in an attempt to draw the statistical conclusion. To this end, we adopt a widely used technique, Saliency Map, employed by [44, 25, 71] to measure how much each input unit in a gate contributes to the final decision, which can be approximated by first derivatives of loss function with respect to each gate. Formally, for the binary classification of neural sentence compression task, the class score $S_c(\mathbf{z})$ is a highly non-linear function. Following the previous study [44], $S_c(\mathbf{z})$ was approximated with a linear function of $\mathbf{z}$ through computing the first-order Taylor expansion:

$$\mathrm{S_c}(\mathbf{z}) \approx \mathbf{w}(\mathbf{z})^{\mathrm{T}}\mathbf{z} + b, \tag{3.22}$$

$$\mathrm{w}(\mathbf{z}) = \frac{\partial(\mathrm{S_c})}{\partial \mathbf{z}}, \tag{3.23}$$

where $w(\mathbf{z})$ is the derivative of S_c with respect to the embedding $\mathbf{z}$. The euclidean norm of this derivative further tells us how much a small change in one specific dimension of the embedding could cause a biggest change in the output. The saliency score is finally defined as $|w(\mathbf{z})|$, indicating the sensitiveness of the binary classification (delete or keep). High saliency score indicates the corresponding feature (embedding) makes a bigger contribution to the final classification.

We herein have five gates, $\mathbf{z}_1$ refers to control signal for word embedding; $\mathbf{z}_2$ refers to control signal for dependency label embedding; $\mathbf{z}_3$ refers to control signal for part-of-speech tag embedding; $\mathbf{z}_4$ refers to control signal for named entity tag embedding; $\mathbf{z}_5$ refers to control signal for interpolation embedding of all features. The test set of six benchmark datasets is employed to calculate the $|w(\mathbf{z})|$ of each in all sentences. We take the average of all $|w(\mathbf{z})|$ and show the result in Table 3.6.

As we can observe, the interpolation gate $\mathbf{z}_5$ has the significant impact on the final binary decision by comparison with other gates, suggesting that the fused feature makes the primary contribution to the final binary classification. This validates the effectiveness of the proposed gate mechanism which is able to combine the features from the different sources.

Feature	GOOGLE	Ziff-davis	Written	Spoken1	Spoken2	Spoken3
Word gate ($\mathbf{z}_1$)	.018	.023	.016	.024	.021	.021
Dep gate ($\mathbf{z}_2$)	.009	.032	.011	.017	.017	.013
POS gate ($\mathbf{z}_3$)	.006	.035	.012	.019	.019	.013
NE gate ($\mathbf{z}_4$)	.008	.033	.009	.018	.018	.012
Inter gate($\mathbf{z}_5$)	**.029**	**.081**	**.042**	**.044**	**.044**	**.036**

Table 3.6: $|w(\mathbf{z})|$ results. The euclidean norm of the derivative of final output S_c with respect to the each gate $\mathbf{z}_i$.

3.6 Discussion

Despite that the integration of linguistic knowledge, e.t., part of speech, dependency relation, and named entity tags are beneficial as shown above, the linguistic pipeline requires external resources e.t., part-of-speech tagger, dependency parser, and named entity recognizer, which inevitably contains errors. Therefore, the model might condition on incorrect linguistic prior to make prediction.

3.7 Conclusion and Future Directions

In this work, we proposed a linguistic knowledge-enhanced gated neural network (LK-GNN) for deletion-based sentence compression. Three kinds of linguistic knowledge were considered: dependency labels, POS tags, and named entity tags. Experimental results on popular sentence compression datasets show that the proposed LK-GNN method yielded comparable or better performance in terms of F_1 score, and generated more readable compression, compared to the baseline method. Meanwhile, the proposed LK-GNN method retained more grammatical relations associated with the ground-truth compression, compared to the baseline method (Stacked-LSTMs). Furthermore, visualization of the gating mechanism suggests that the proposed model can selectively employ different features to make the best prediction.

In the future, we will consider additional semantic features, such as combinatory categorical grammar tags or dependency minimal recursion semantics relations, in order to enrich our feature extractions. We will also take into account tree structure information, and explicitly incorporate it into the neural model. Furthermore, an existing automatic evaluation metric is still not comparable to human judgement, and a better automatic evaluation metric is thus needed for both text summarization and compression. Recent research [13] may shed light on this. We will leave it as our future studies.

Chapter 4

Improving Readability using Sentence-Level Linguistic Knowledge for Single Sentence Compression

Last chapter shows that integration of word-level linguistic knowledge is beneficial to improving Readability[1], while in this chapter, we incorporate sentence-level linguistic knowledge into neural sentence compression model. Further, we directly model and optimize readability and informativeness.

In Chapter 4.1, we extend the word-level linguistic knowledge to sentence-level linguistic knowledge and inject Sentence Structural bias into neural model. In Chapter 4.2, we investigate the informativeness modeling. Since the informativeness is a subjective concept, we examined ten quantity able to correlate with human informativeness judgements. In Chapter 4.3, we present a neural evaluator capable of rewarding well-formed compression via reinforcement learning framework. We give a summary in Chapter 4.4.

[1] In this work, grammaticality/readability/fluency are exchangeable in terms.

4.1 Exploiting Language Model as Evaluator

We herein present a language-model-based evaluator for deletion-based sentence compression [82], and viewed this task as a series of deletion-and-evaluation operations using the evaluator. More specifically, the evaluator is a syntactic neural language model that is first built by learning the syntactic and structural collocation among words. Subsequently, a series of trial-and-error deletion operations are conducted on the source sentences via a reinforcement learning framework to obtain the best target compression. An empirical study shows that the proposed model can effectively generate more readable compression, comparable or superior to several strong baselines. Furthermore, we introduce a 200-sentence test set for a large-scale dataset, setting a new baseline for the future research.

4.1.1 Introduction

Deletion-based sentence compression aims to delete unnecessary words from source sentence to form a short sentence (compression) while retaining grammatical and faithful to the underlying meaning of the source sentence. Previous works used either machine-learning-based approach or syntactic-tree-based approaches to yield most readable and informative compression [37, 39, 17, 53, 19, 30, 8, 28, 9, 3, 76]. For example, [19] proposed a syntactic-tree-based method that considers the sentence compression task as an optimization problem by using integer linear programming, whereas [28] viewed the sentence compression task as a sequence labeling problem using the recurrent neural network (RNN), using maximum likelihood as the objective function for optimization. The latter sets a relatively strong baseline by training the model on a large-scale parallel corpus. Although an RNN (e.g., Long short-term memory networks) can implicitly model syntactic information, it still produces ungrammatical sentences. We argue that this is because (i) the labels (or compressions) are automatically yielded by employing the syntactic-tree-pruning method. It thus contains some errors caused by syntactic tree parsing error, (ii) more importantly, the optimization objective of an RNN is the likelihood function that is based on individual words instead of readability (or informativeness) of the whole compressed sentence. A gap exists between optimization objective and evaluation. As such, we are of great interest that: (i) can we take the readability of the whole compressed sentence as a learning objective and (ii) can grammar errors be recovered through a language-model-based evaluator to yield compression with better quality?

To answer the above questions, a syntax-based neural language model is trained on large-scale datasets as a readability evaluator. The neural language model is supposed to learn the correct word collocations in terms of both syntax and semantics. Subsequently, we formulate the deletion-based sentence compression as a series of trial-and-error deletion operations through a reinforcement learning framework. The policy network performs either RETAIN or REMOVE action to form a compression, and receives a reward (e.g., readability score) to update the network.

The empirical study shows that the proposed method can produce more readable sentences that preserve the source sentences, comparable or superior to several strong baselines. In short, our contributions are two-fold: (i) an effective syntax-based evaluator is built as a post-hoc checker, yielding compression with better quality based upon the evaluation metrics; (ii) a large scale news dataset with 1.02 million sentence compression pairs are compiled for this task in addition to 200 manually created sentences.

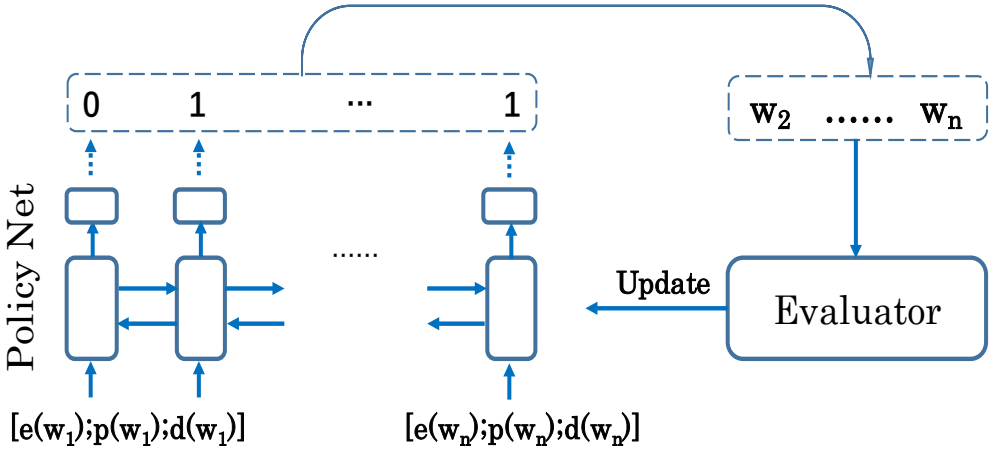


Figure 4.1: Graphical illustration of the framework.

4.1.2 Methodology

Task and Framework

Formally, deletion-based sentence compression translates word tokens, $(w_1, w_2, ..., w_n)$ into a series of ones and zeros, $(l_1, l_2, ..., l_n)$, where n refers to the length of the original sentence and $l_i \in \{0, 1\}$. Here, "1" refers to RETAIN and "0" refers to REMOVE. We first converted the word sequence into a dense vector representation through the parameter matrix E. Except for word embedding, $(\mathbf{e}(w_1), \mathbf{e}(w_2), ..., \mathbf{e}(w_n))$, we also considered the part-of-speech tag and the dependency relation between w_i and its head word as extra features. Each part-of-speech tag was mapped into a vector representation, $(\mathbf{p}(\mathrm{w}_1), \mathbf{p}(\mathrm{w}_2), ..., \mathbf{p}(\mathrm{w}_n))$ through the parameter matrix P, while each dependency relation was mapped into a vector representation, $(\mathbf{d}(\mathrm{w}_1), \mathbf{d}(\mathrm{w}_2), ..., \mathbf{d}(\mathrm{w}_n))$ through the parameter matrix D. Three vector representations are concatenated, $[\mathbf{e}(\mathrm{w_i}); \mathbf{p}(\mathrm{w_i}); \mathbf{d}(\mathrm{w_i})]$ as the input to the next part, policy network.

Figure 4.10 shows the graphical illustration of our model. The policy network is a bi-directional RNN that uses the input $[\mathbf{e}(\mathrm{w_i}); \mathbf{p}(\mathrm{w_i}); \mathbf{d}(\mathrm{w_i})]$ and yields the hidden states in the forward direction, $(\mathbf{h}^{\mathrm{f}}_1, \mathbf{h}^{\mathrm{f}}_2, ..., \mathbf{h}^{\mathrm{f}}_{\mathrm{n}})$, and hidden states in the backward direction, $(\mathbf{h}^{\mathrm{b}}_1, \mathbf{h}^{\mathrm{b}}_2, ..., \mathbf{h}^{\mathrm{b}}_{\mathrm{n}})$. Then, concatenation of hidden states in both directions, $[\mathbf{h}^{\mathrm{f}}_{\mathrm{i}}; \mathbf{h}^{\mathrm{b}}_{\mathrm{i}}]$ are followed by a nonlinear layer to turn the output into a binary probability distribution, $y_i = \sigma(W[\mathbf{h}^f_i; \mathbf{h}^b_i])$ where σ is a nonlinear function sigmoid, and W is a parameter matrix.

The policy network continues to sample actions from the binary probability distribution above until the whole action sequence is yielded. In this task, binary actions space is {RETAIN, REMOVE}. We turn the action sequence into the predicted compression, $(w_1, w_2, ..., w_m)$, by deleting the words whose current action is REMOVE. Then the $(w_1, w_2, ..., w_m)$ is fed into a pre-trained evaluator which will be described in the next section.

Syntax-based Evaluator

The syntax-based evaluator should assess the degree to which the compressed sentence is grammatical, through being used as a reward function during the reinforcement learning phase. It needs to satisfy three conditions: (i) grammatical compressions should obtain a higher score than ungrammatical compressions, (ii)

for two ungrammatical compressions, it should be able to discriminate them through the score despite the ungrammaticality, (iii) lack of important parts (such as the primary subject or verb) in the original sentence should receive a greater penalty.

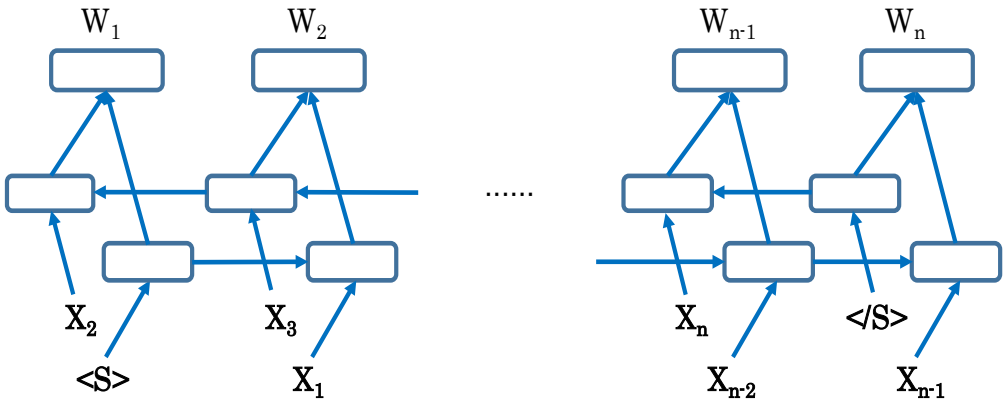


Figure 4.2: Graphical illustration of bi-directional recurrent neural network language model.

We therefore considered an ad-hoc evaluator, i.e., the syntax-based language model (evaluator-SLM) for these requirements. It integrates the part-of-speech tags and the dependency relations in the input, while the output to be predicted is the next word token. We observed that the prediction of the next word could not only be based on the previous word but also the syntactic components, e.g., for the part-of-speech tag, the noun is often followed by a verb instead of an adjective or adverb and the integration of the part-of-speech tag allows the model to learn such correct word collocations. Figure 4.11 shows the graphical illustration of the evaluator-SLM where the input is $x_i = [\mathbf{e}(w_i); \mathbf{p}(w_i); \mathbf{d}(w_i)]$, followed by a bi-directional RNN whose last layer is the Softmax layer used to represent word probability distribution. Similar to [57], we added two special tokens, <S> and </S> in the input so as to stagger the hidden vectors, thus avoiding self-prediction. Finally, we have the following formula as one part of the reward functions in the learning framework.

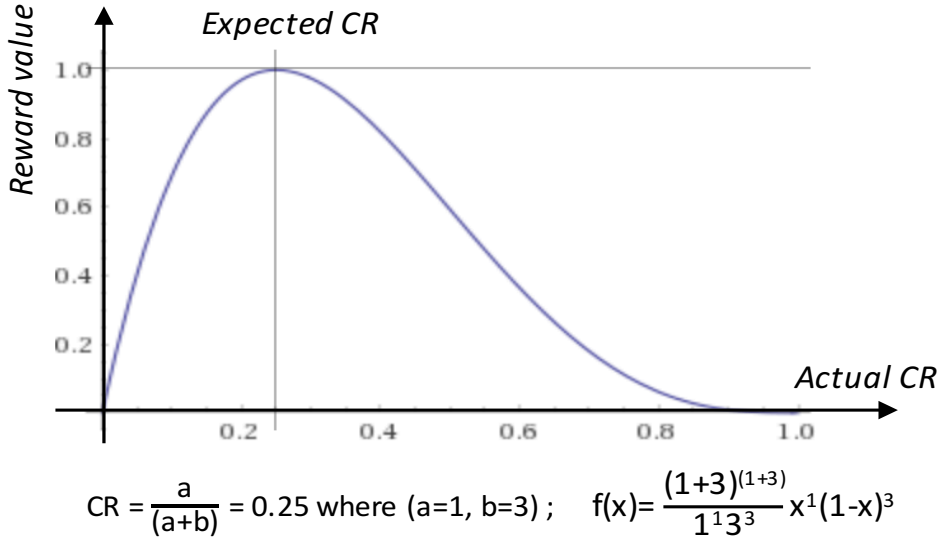
$$\mathrm{R_{SLM}}(\widehat{\mathrm{Y}}) = \mathrm{e}^{(\frac{1}{|\widehat{\mathrm{Y}}|}\sum_{\mathrm{t}=1}^{|\widehat{\mathrm{Y}}|}\log \mathrm{P_{LM}}(\mathrm{y_t}|\mathrm{y_{0:t-1}}))}, \tag{4.1}$$

where $\mathrm{R_{SLM}} \in [0,1]$ and $\widehat{\mathrm{Y}}$ is the predicted compression by the policy network. Further, it is noteworthy that the performance comparison should be based on a similar compression rate[2] (CR) [58], and a smooth reward function $\mathrm{R_{CR}}$ (both a, b are positive integers; e.g. $a = 2$, $b = 2$ could lead the compression rate to $\frac{a}{a+b} = \frac{2}{2+2} = 0.5$) is also used to attain a compressed sentence of similar length. We called it length reward function.

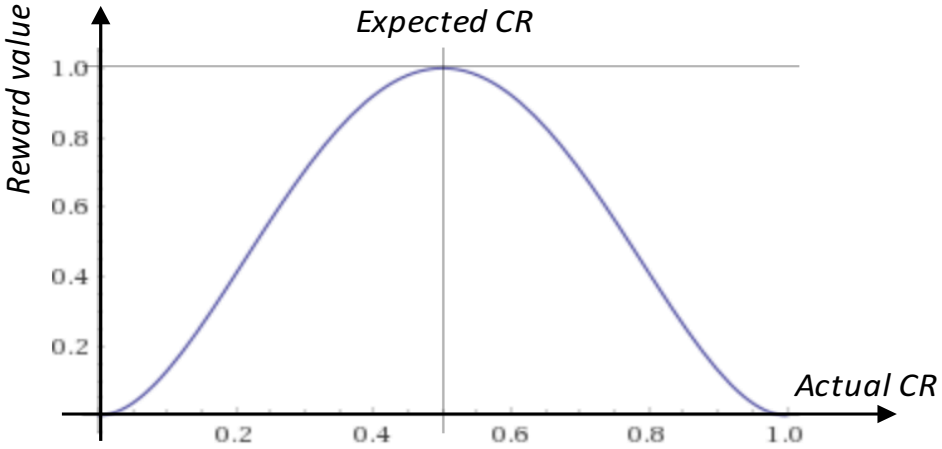
$$\mathrm{R_{CR}} = \frac{(\mathrm{a}+\mathrm{b})^{(\mathrm{a}+\mathrm{b})}}{\mathrm{a^a b^b}}\mathrm{x^a}(1-\mathrm{x})^{\mathrm{b}}. \tag{4.2}$$

To elaborate how the $\mathrm{R_{CR}}$ works, we give several graphical illustration of length reward function with different a and b as follows. Motivated by that we hope to specify compression according to different cases, we specific

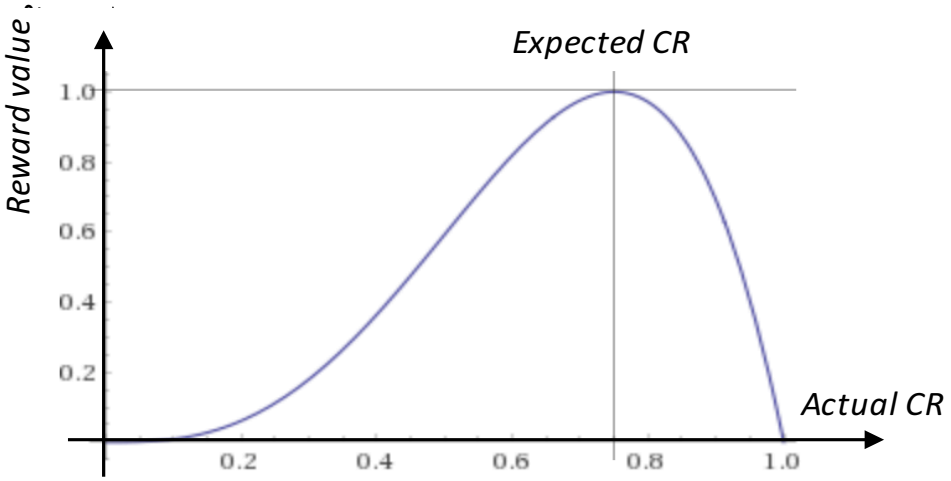
[2]compression rate is the length of compression divided by the length of the sentence.



$$CR = \frac{a}{(a+b)} = 0.25 \text{ where } (a=1, b=3); \quad f(x) = \frac{(1+3)^{(1+3)}}{1^1 3^3} x^1 (1-x)^3$$



$$CR = \frac{a}{(a+b)} = 0.50 \text{ where } (a=2, b=2); \quad f(x) = \frac{(2+2)^{(2+2)}}{2^2 2^2} x^2 (1-x)^2$$



$$CR = \frac{a}{(a+b)} = 0.75 \text{ where } (a=3, b=1); \quad f(x) = \frac{(3+1)^{(3+1)}}{3^3 1^1} x^3 (1-x)^1$$

Figure 4.3: Graphical illustration of compression rate reward function with different setting.

The total reward is $\mathrm{R} = \mathrm{R_{SLM}} + \mathrm{R_{CR}}$. By using policy gradient methods [73], the policy network is updated with the following gradient:

$$\nabla\mathcal{L}(\theta) = \sum_{\mathrm{t}=1}^{|\widehat{\mathrm{Y}}|} \mathrm{R}(\widehat{\mathrm{Y}})\nabla\log\pi_\theta(\mathrm{a_t}|\mathbf{s_t}), \tag{4.3}$$

Where $a_t \in \{\text{RETAIN, REMOVE}\}$, is the action token by the policy network, and $\mathbf{s}_t$ refers to hidden state of the network, $[\mathbf{h}_i^f; \mathbf{h}_i^b]$.

4.1.3 Experiments

Data

As neural network-based methods require a large amount of training data, we for the first time considered using Gigaword[3], a news domain corpus. More specifically, the first sentence and the headline of each article are extracted. After data cleansing, we finally compiled 1.02 million sentence and headline pairs (see details here[4]). It is noteworthy that the headline is not the extractive compression. Further, we asked two near native English speakers to create 200 extractive compressions for the first 200 sentences of this dataset. We modified the original Annotator Sentence Compression Instructions in clarke2008global and present it as follow:

> *This experiment is concerned with sentence compression. You will be presented with a selection of sentences from a news paper article. Your task is to compress each sentence by removing words.*
>
> *Compressing a sentence involves taking a the original sentence and producing a shorter version while retaining the most important information contained within the sentence. The compressions you will produced should be constrained such that the compressed sentence can only be composed of words found in the original sentence and the ordering of words must not change. Words can only be removed from the sentence, there is no opportunity for the addition or reordering of words.*
>
> *Ideally the compressed sentence will be grammatical and retain the most important information of the original sentence. All compressions produced are considered valid provided they have been made while considering:*
>
> - *The most important information in the original sentence.*
> - *The grammaticality of the compressed sentence.*

we use the annotated data set as the testing set, the first 1,000 sentences (excluding the testing set) is the development set, and the remainder is the training set. To assess the inter-assessor agreements, we computed Cohen 's unweighted κ. The computed unweighted κ was 0.423, reaching a moderate agreement level[5].

The second dataset we used was the Google dataset that contains 200,000 sentence compression pairs [28]. For the purpose of comparison, we used the

[3] https://catalog.ldc.upenn.edu/ldc2011t07

[4] https://github.com/code4conference/Data

[5] [42] characterize κ values <0 as no agreement, 0 $\sim$ 0.20 as slight, 0.21 $\sim$ 0.40 as fair, 0.41 $\sim$ 0.60 as moderate, 0.61 $\sim$ 0.80 as substantial, and 0.81 $\sim$ 1 as almost perfect agreement.)

very first 1,000 sentences as the testing set, the next 1,000 sentences as the development set, and the remainder as the training set.

Comparison Methods

We choose several strong baselines; the first one is the dependency-tree-based method that considers the sentence compression task as an optimization problem by using integer linear programming[6]. Inspired by [30], [19], and [76], we defined some constrains: (1) if a word is retained in the compression, its parent should be also retained. (2) whether a word w_i is retained should partly depend on the word importance score that is the product of the TF-IDF score and headline score $\mathrm{h}(w_i)$, $\text{tf-idf}(w_i) \cdot \mathrm{h}(w_i)$ where $\mathrm{h}(w_i)$ represents that whether a word (limited to nouns and verbs) is also in the headline. $\mathrm{h}(w_i)$=5 if w_i is in the headline; $\mathrm{h}(w_i)$=1 otherwise. (3) the dependency relations, ROOT, dobj, nsubj, pobj, should be retained as they are the skeletons of a sentence. (4) the sentence length should be over than α but less than β. (5) the depth of the node (word), $\lambda\mathrm{dep}(w_i)$, in the dependency tree. (6) the word with the dependency relation *amod* is to be removed. It is noteworthy that the method is unsupervised.

Gigaword Dataset	Annotator 1		Annotator 2		
	F_1	RASP-F_1	F_1	RASP-F_1	CR
#1 Seq2seq with attention	54.9*	60.3*	58.6*	64.6*	0.53
#2 Dependency tree+ILP	58.0*	65.1*	61.0*	70.9*	0.55
#3 LSTMs+pseudo label	60.3*	64.1*	64.1*	69.2*	0.51
#4 Evaluator-LM	64.5	67.3	66.9	72.2	0.50
#5 Evaluator-SLM	65.0*	69.6*	68.2*	73.9*	0.51

Table 4.1: F_1 and RASP-F_1 results for Gigaword dataset. * stands for significant difference between proposed methods and comparison methods.

The second method is the long short-term memory networks (LSTMs) which showed strong promise in sentence compression by [28]. The labels were obtained using the dependency tree pruning method [29] and the LSTMs were applied in a supervised manner. Following their works, we also consider the labels yielded by our dependency-tree-based method as pseudo labels and employ LSTMs as a baseline.

Furthermore, for a comprehensive comparison, we applied the sequence-to-sequence with attention method widely used in abstractive text summarization for sentence compression. Previous works such as [67, 16] have shown promising results with this framework, although the focus was generation-based summarization rather than extractive summarization. More specifically, the source sequence of this framework is the original sentence, while the target sequence is a series of zeros and ones (zeros represents REMOVE and ones represents RETAIN). Further, we incorporated dependency labels and part-of-speech tag features in the source side of the sequence-to-sequence method.

Training

The embedding size for word, part-of-speech tag, and the dependency relation is 128. We employed the vanilla RNN with a hidden size of 512 for both the policy network and neural language model. The mini-batch size was chosen from

[6] we use http://pypi.python.org/pypi/PuLP

[5, 50, 100]. Vocabulary size was 50,000. The learning rate for neural language model is 2.5e-4, and 1e-05 for the policy network. For policy learning, we used the REINFORCE algorithm [78] to update the parameters of the policy network and find an policy that maximizes the reward. Because starting from a random policy is impractical owing to the high variance, we pre-trained the policy network using pseudo labels in a supervised manner. For the comparison methods, the hyperparameters and were set to 0.4 and 0.7, respectively, and was set to 0.5.

4.1.4 Result and Discussion

This section demonstrates the experimental results on both datasets. As the Gigaword dataset has no ground truth, we evaluated the baseline and our method on the 200-sentence test sets created by two human annotators. For the automatic evaluation, we employed F_1 and RASP-F_1 [11] to measure the performances. The latter compares grammatical relations (such as ncsubj and dobj) found in the system compressions with those found in the gold standard, providing a means to measure the semantic aspects of the compression quality. For the human evaluation, we asked two near native English speakers to assess the quality of 50 compressed sentences out of the 200-sentence test set in terms of readability and informativeness. Here are our observations:

Gigaword	*Readability*	*Informativeness*
$1 LSTMs	3.56	3.10
$2 SLM	4.16*	3.16

Table 4.2: Human Evaluation for Gigaword dataset. * stands for significant difference with 0.95 confidence in the column.

Google Dataset	F_1	**RASP-F_1**	**CR**
&1 Seq2seq with attention	71.7	63.8	0.34
&2 LSTM (Filippova, 2015)	82.0	-	0.38
&3 LSTMs (our implement)	84.8	81.9	0.40
&4 Evaluator-LM	85.0	82.0	0.41
&5 Evaluator-SLM	85.1	82.3	0.39

Table 4.3: F_1 and RASP-F_1 results for Google dataset.

1 As shown in Table 4.14, our Evaluator-SLM-based method yields a large improvement over the baselines, demonstrating that the language-model-based evaluator is effective as a post-hoc grammar checker for the compressed sentences. This is also validated by the significant improvement in the readability score in Table 4.15 ($1 vs $2). To investigate the evaluator in detail, a case study is shown in Figure 4.13.

2 by comparing annotator 1 with annotator 2 in Table 4.14, we observed different performances for two annotated test sets, showing that compressing a text while preserving the original sentence is subjective across the annotators.

3 As for Google news dataset, LSTMs (LSTM+pos+dep) (&3) is a relatively strong baseline, suggesting that incorporating dependency relations and

part-of-speech tags may help model learn the syntactic relations and thus make a better prediction. When further applying Evaluator-SLM, only a tiny improvement is observed (&3 vs &4), not comparable to the improvement between #3 and #5. This may be due to the difference in perplexity of the our Evaluator-SLM. For Gigaword dataset with 1.02 million instances, the perplexity of the language model is 20.3, while for the Google news dataset with 0.2 million instances, the perplexity is 76.5.

4 To further explore the degree to which syntactic knowledge (dependency relations and part-of-speech tags) is helpful to evaluator (language model), we implemented a naive language model, i.e., Evaluator-LM, which did not include dependency relations and part-of-speech tags as input features. The results shows that small improvements are observed on two datasets (#4 vs #5; &4 vs &5), suggesting that incorporating syntactic knowledge may help evaluator to encourage more unseen but reasonable word collocations.

4.1.5 Case Study

We show several example sentences from test set of Gigaword corpus in Table 4.17.

a.Original sentence **b.**Output of LSTMs baseline **c.**Output of SLM	
a1:	Indonesia issued a tsunami warning Thursday after a powerful quake rocked an eastern island chain, sending residents fleeing from their homes, authorities and witnesses said.
b1:	Indonesia issued a tsunami warning thursday quake rocked chain sending residents authorities and witnesses said.
c1:	Indonesia issued a tsunami warning thursday after a powerful quake rocked an eastern island.
a2:	The French airline Air Liberty is to complain to the commission of the European union to obtain equal treatment for all airlines at London Heathrow and Paris Orly airports, the company said on Monday.
b2:	The airline air liberty is to complain to commission to obtain treatment for airlines airports the company said on monday.
c2:	The french airline air liberty is to complain to the commission of the European union to obtain treatment.
a3	Russian President Vladimir Putin Wednesday sent a congratulatory message to South Korean President Kim Dae Jung on the national liberation day, or the Day of Revival, said the Kremlin.
b3	President vladimir putin wednesday sent a message to dae jung on day or day said the kremlin press service.
c3	Russian president vladimir putin wednesday sent a congratulatory message to south korean president kim dae jung on.

Table 4.4: Example sentences and its compression results.

Regarding the sentence a1, "Indonesia issued a tsunami warning Thursday after a powerful quake rocked an eastern island chain, sending residents fleeing from their homes, authorities and witnesse said." When comparing b1 with c1, b1 is unreadable as a result of lack of the conjunction "after", while the output of SLM c1 that keeps the word "after" and removes the adverbial clause "sending

residents ..." is much more readable. Another interesting point is that with respect to sentence a2, both compression outputs of the LSTMs and the proposed SLM drop the *amod* component, "equal", to make sentence concise.

However, by comparison with c2, b2 lacks preposition, "at" and thus yielded not such readable compression. In addition, for some cases such as sentence a3 and its corresponding compression b3 as well as c3, the outputs from both models is either making no sense or ungrammatical.

4.1.6 Analysis of Evaluator

Qualitative Analysis

To further analyze the Evaluator-SLM performance, we used an example sentence, "The Dalian shipyard has built two new huge ships" to observe how a language model scores different word deletion operations. We converted the reward function R_{SLM} to $e^{-\log R_{SLM}}$ for a better observation (similar to "sentence perplexity", the higher the score is, the worse is the sentence). As shown in Figure 4.13, deleting the object(#2), verb(#3), or subject(#4) results in a significant increase in "sentence perplexity", implying that the syntax-based language model is highly sensitive to the lack of such syntactic components. Interestingly, when deleting words such as "new" or/and "huge," the score becomes lower, suggesting that the model may prefer short sentences, with unnecessary parts such as *amod* being removed. This property makes it quite suitable for the sentence compression task aiming to shorten sentences by removing unnecessary words.

As the case study shows, the evaluator[7] is able to detect grammar error and sensitive to the lack of important syntactic component. To testify these hypothesis, we conduct the following two quantitative analysis.

Quantitative Analysis I

The first hypothesis is: the linguistic knowledge-based neural language model is a more accurate readability estimator by comparison with vanilla neural language model. We handcrafted readable and unreadable sentence pairs, comparing whether the language model can assign a lower perplexity score for the readable sentence than unreadable sentences. (note that for sentence-level language model perplexity, the lower, the better).

To be more specific, we use 1,000 sentences in the test set of Google dataset and applied dependency parser to identify which word is the ROOT node. Then, we take the original sentences as readable ones, and original sentences without ROOT node as unreadable ones[8]. We calculate the percentage of cases where the perplexity of the readable sentence is lower than that of the unreadable sentence for both knowledge-based language model and vanilla language model, as defined in the following equation:

$$\text{Accuracy} = \frac{\text{correct prediction}}{\text{\# of pairs}}.$$

Both language models are trained on the same Gigaword corpus, and we stop the training when the validation loss no longer drops. The linguistic knowledge refers to dependency relation labels and part-of-speech tags as detailed in section 4.3.2.

As shown in Figure 4.17, the accuracy yielded by the knowledge-based language model is higher than that produced by the vanilla language model, indi-

[7]The term, evaluator, herein specifically refers to neural language model.

[8]Removing ROOT word makes the sentence unreadable for the vast majority of cases.

SENTENCE	The	Dalian	shipyard	has	built	two	new	huge	ships	$e^{-\log R}$
POS tags	*DET*	*ADJ*	*NOUN*	*VERB*	*VERB*	*NUM*	*ADJ*	*ADJ*	*NOUN*	
DEP. rels	*det*	*compound*	***nsubj***	*aux*	***root***	*nummod*	*amod*	*amod*	***dobj***	
#1	The	Dalian	shipyard	has	built	two	new	huge	ships	59.8
#2	The	Dalian	shipyard	has	built	two	new	huge		140.5
#3	The	Dalian	shipyard	has		two	new	huge	ships	582.9
#4	The	Dalian		has	built	two	new	huge	ships	1313.5
#5	The	Dalian		has	built	two			ships	1244.8
#6	The			has	built	two			ships	1331.2
#7	The	Dalian	shipyard	has	built	two		huge	ships	46.9
#8	The	Dalian	shipyard	has	built	two			ships	18.2
#9	The		shipyard	has	built	two			ships	66.5

Figure 4.4: Case study for evaluator.

Model	Accuracy
LM w/o linguistic knowledge	85.5
LM w/ linguistic knowledge	**87.3**

Table 4.5: Accuracy for sentence-level perplexity prediction. LM refers to the neural language model and while knowledge refers to linguistic knowledge.

cating that incorporating linguistic knowledge into the language model results in a more accurate readability evaluator. It confirms our first hypothesis.

Quantitative Analysis II

Our second hypothesis is: linguistic knowledge-based language model is more sensitive to the lack of important syntactic components compared to the vanilla language model. The second hypothesis could be viewed as an extension of the first hypothesis in the sense that the second one measures not only whether the knowledge-based language model is more sensitive to the lack of important syntactic content than the vanilla language model, but also to what degree the sensitiveness is.

Dep. labels	det	nsubj	ROOT	det	amod	compound	dobj	prep	det	compound	pobj
#1. Original	A	man	suffered	a	serious	head	injury	after	a	car	crash
#2. Removing ROOT	A	man	suffered	a	serious	head	injury	after	a	car	crash
#3. Removing randomly		man	suffered	a	serious	head	injury	after	a	car	crash

Figure 4.5: Three example sentences.

To the end, we handcrafted the same test set as used in Analysis I in a similar way, as shown in Figure 4.14. The (#1) original sentence and (#2) the original sentence without the ROOT word are employed. In addition, we randomly removed a word in the sentence to form a baseline sentence (#3). We use the reword function in Equation 4.8 to measure the linguistic quality of a sentence:

$$\mathrm{R(S)} = \mathrm{e}^{(\frac{1}{|\mathrm{S}|}\sum_{\mathrm{t}=1}^{|\mathrm{S}|}\log \mathrm{P_{LM}(s_t|y_{0:t-1})})}, \tag{4.4}$$

Where S refers to the sentence, while s_t refers to the t-th word in the sentence. LM is short for language model. By definition in Equation 4.11, the domain of R(S) is [0, 1]. And, the higher R(S) is, the better-quality the sentence S is. For example, in Figure 4.14, #1 is a readable sentence, while #2 is an unreadable sentence as a result of a lack of ROOT component. #3 could be readable or unreadable because it depends on whether the important syntactic component is being removed. We intentionally avoid the case where random word is ROOT word for meaningful comparison. After all of these being set up, we define the sensitiveness score as:

$$\text{Sensitiveness} = [\mathrm{R(S_{ori})} - \mathrm{R(S_{ROOT})}] - [\mathrm{R(S_{ori})} - \mathrm{R(S_{random})}], \tag{4.5}$$

Where $\mathrm{S_{ori}}$ refers to the original sentence, $\mathrm{S_{ROOT}}$ refers to the original sentence without ROOT word, and $\mathrm{S_{random}}$ refers to the original with one word being randomly removed. Table summarizes the sensitiveness score results.

Model	$R(S_{ori}) - R(S_{ROOT})$	$R(S_{ori}) - R(S_{random})$	Sen. score
LM w/o knowledge	.012(±9e-4)	.009(±7e-4)	.003
LM w/ knowledge	.021(±1.2e-3)	.015(±1.1e-3)	**.006**

Table 4.6: Sensitiveness (Sen.) score results. LM w/o knowledge refers to the language model without linguistic knowledge, while LM w/ knowledge refers to the language model with linguistic knowledge. The linguistic knowledge refers specifically to the dependency label and part-of-speech tag. Sen. score is the sensitiveness score in Equation 4.12.

As Table 4.18 shown, the language model with linguistic knowledge achieves higher sensitiveness scores, indicating that for the cases where the important syntactic components such as ROOT lack, the model enhanced by the knowledge will have a bigger penalty (or less reword), compared to the vanilla language model. This quantitative analysis once again confirms that linguistic knowledge is essential to building up an error-sensitive evaluator.

4.1.7 Section Summary

We presented a syntax-based language model for the sentence compression task. We employed unsupervised methods to yield labels to train a policy network in a supervised manner. The experimental results demonstrates that the compression could be further improved by a post-hoc language-model-based evaluator, and our evaluator-enhanced model performs better or comparable upon the evaluation metrics on two large-scale datasets.

4.2 Conclusion and Future Direction

In this chapter, we focus on integrating sentence-level linguistic knowledge. In Chapter 4.1, we present a sentence structural biased model to improve prediction accuracy. The experimental results demonstrate the effectiveness of this implicit integration of sentence-level linguistic prior. In Chapter 4.2, we move forward toward modeling informativeness and investigated which quantity correlates the best with human informativeness judgments. we found that the compression rate is strongly correlated with human judges among other nice quantities. Further, the CR also shows a weak correlation with human readability judgments. This finding, again, highlight the importance of maintaining the similar compression rate when comparing different systems of single sentence compression task. In Chapter 4.3, we present a language-model-based evaluator and viewed this task as a series of deletion-and-evaluation operations using the evaluator. Evaluator is composed of two modules, the first is a syntactic knowledge-enhanced language model; In light of the finding that CR should be similar, we put a compression rate module to control the CR during training. The empirical study shows that the proposed method can effectively generate more readable compression.

As a future direction, automatic readability assessment would be beneficial to a lot of applications including our evaluator. The current exciting progress on pre-trained model like Generative Pre-trained Transformer [64] from OpenAI show its promise in generating text. In spite of its computational cost, how to distillate or prune these huge models (the best-performing model has 1.5 billion parameters) to boost the readability evaluation would be a promising direction.

Chapter 5

External Knowledge-Enhanced Unsupervised Rewriter for Multiple Sentence Compression

Chapter 3 and 4 are concerned with single-sentence compression scenario, while this chapter extends it to the scenario where multiple sentences are compressed into one single compression. The multiple sentence compression are different from single sentence compression because multi-sentence compression yield longer compression output, which pose a greater challenge to the readability issue.

By definition, multi-sentence compression aims to generate a readable, but reduced compression from multiple input sentences while retaining key information in source. Over the past years, the extraction-based word graph approach has attracted incredible attention because of its simplicity and effectiveness. A few recent works further leveraged lexical substitution to yield more abstractive compression. However, two limitations exist in these methods. First, the word graph approach that simply concatenates fragments from multiple sentences may yield disfluent or ungrammatical compression. Second, lexical substitution is often inappropriate without consideration of context information. Furthermore, the lack of large parallel corpus prevents deep learning techniques from being readily applied. To tackle the above-mentioned issues, we present herein a neural rewriter for multi-sentence compression without any parallel corpus. Empirical studies have shown that our approaches achieve comparable results upon automatic evaluation and generate more grammatical compression based on human evaluation. A parallel corpus with more than 140,000 (sentence group, compression) pairs is also constructed as a by-product for future research.

The Chapter 5.1 gives an introduction of the work. The Chapter 5.2 details the related works in multiple sentence compression with a focus on previous dominating word-graph approach. To overcome the lack of parallel data, In Chapter 5.3, we introduce a large-scale multiple sentence compression dataset. It is followed by our proposed approach, rewriter in the Chapter 5.4. The Chapter 5.5 and 5.6 elaborate the experimental results and analysis. Lastly, we summarize this work and discuss the future direction.

5.1 Introduction

Multi-sentence compression (MSC) aims to generate a single shorter and grammatical sentence that preserves important information from a group of related sentences. Over the past decade, multiple sentence compression has attracted considerable attention owing to its potential applications, such as compressing the related news events from multiple sources. It also benefits other natural language processing tasks, such as multi-document summarization [6], opinion summarization, and text simplification. Most existing works rely on the word graph

approach initialized in [27], which offers a simple solution that copies fragments from different input sentences and concatenates them to form the final compression. Later on, a bunch of subsequent research works [10, 6, 48, 70, 63, 60] attempted to improve the word graph approach using a variety of strategies such as keyphrase re-ranking. However, such extraction-based approach may yield disfluent or ungrammatical compression. A previous study [59] has shown that word graph approaches produce more than 30% of the ungrammatical sentences, which is partly due to the non-usage of rewording by these extraction-based approaches. In fact, human annotators tend to compress a sentence through several rewriting operations, such as substitution and rewording [21].

% of Sentences	Paraphrase Rate
90%	>1%
50%	>5%
15%	>10%

Table 5.1: Statistics of Cornell dataset. Paraphrase Rate presents how many precent of novel words (%) human annotators used to create the reference compression.

We analyzed a benchmark multi-sentence compression corpus[1], and its statistics in Table 5.1 shows that a majority of compressed sentences (90%) contain novel words[2] (paraphrases). Among others, 15% of reference compression even contains more than %10 new words. Despite some research works such as [59] that attempt to do the lexical substitution, it is often inappropriate without the consideration of context information.

To tackle the above-mentioned problems, we present herein an unsupervised rewriter to improve the grammaticality of compression while introducing an appropriate amount of novel words [84]. Inspired by the unsupervised machine translation [69, 26], we adopted the back-translation technique to our setting. Unlike machine translation, in the case of compression task, multiple input sentences and single output compression usually do not have semantic equivalence, which complicates the application of the back-translation technique. Thus, we propose a rewriting scheme that first exploits word graph approach to produce coarse-grained compression (B), based on which we substitute words with their shorter synonyms to yield paraphrased sentence (C). A neural rewriter is subsequently applied to the semantically equivalent (B, C) pairs in order to improve the grammaticality and encourage more novel words in compression. Our contributions are two-fold:

1. We present a neural rewriter for multi-sentence compression without any parallel data. This rewriter significantly improves the grammaticality and novel word rate, while maintaining the information coverage (informativeness) according to automatic evaluation.

2. A large-scale multi-sentence compression corpus is introduced along with a manually created test set for future research.

[1] This corpus consists of 300 parallel instances where one instance refers to a pair of (sentence, five human references).

[2] % of novel words $= 1 - \frac{|S \cap C|}{|C|}$, where S refers to words of all input sentences, while C refers to words of compression.

5.2 Background

Multiple Sentence Compression (MSC) aims to produces a single readable sentence covering all or a fraction of the relevant information from the input, given a group of related sentences. [7] presented probably the earliest work in the field of multiple sentence compression. They introduced a text-to-text generation technique of expressing content common to most of the input sentences in one single compressed sentence. [31] viewed multiple sentence compression problem as an integer linear programming task by merging the multiple dependency graph, showing promising results for German language. Later on, a more lightweight approach, the word graph approach was proposed by the same author [27], which only requires a part-of-speech tagger. Due to its simplicity and effectiveness, this approach has quickly become the de facto standard for multiple sentence compression.

considering the following three related sentences about the same event (e.t., John mccain passed away).

1. Tributes from former US presidents have poured in for Republican Senator John McCain, who has died aged 81.

2. Senator John McCain, an American Original, Died at Age 81.

3. US Senator John McCain died aged 81 after battle with brain cancer.

As shown in Figure 5.1, a directed word graph is constructed from above related sentences in which nodes represent unique words, and edges express the adjacent relation of words. Edge weights are calculated using the pre-defined weighting function. Then, a K-shortest path algorithm is applied to find the top-k shortest path from the START node to the END node in the graph. A plausible candidate compression (a path in yellow color) was shown in Figure 5.2.

In spite of the simplicity of this approach, [27] reported that the original word graph approach missed 48%–60% important information, which motivated later works, such as [10], to identify the keyphrase in the source using unsupervised keyword detection method [75] to cover as much important information (informativeness) as possible. However, the price for higher informativeness scores is a drop in grammaticality. To simultaneously improve readability and informativeness, [6] further defined the linguistic quality (readability) and informativeness function as the constraints via the integer linear programming approach. Despite the promise brought about by the above-mentioned studies, the conflict between readability and informativeness still exists.

A few recent works started to consider paraphrasing for multi-sentence compression at the lexical level. [59] substitute words with their paraphrasing counterparts, while [60] considered word embedding to overcome the word graph drawback of two sentences possibly talking about the same topic yet without word overlap. Nevertheless, 25% of the generated compression remained ungrammatical according to [59]. Compared to these works, our work is different for two aspects:

- First, we not only substitute the words using external knowledge but also present a rewriter to polish the coarse-grained compression into more readable compression.

- Secondly, a large-scale dataset was collected to alleviate the lack of parallel corpus, making the generation-based neural model readily applied.

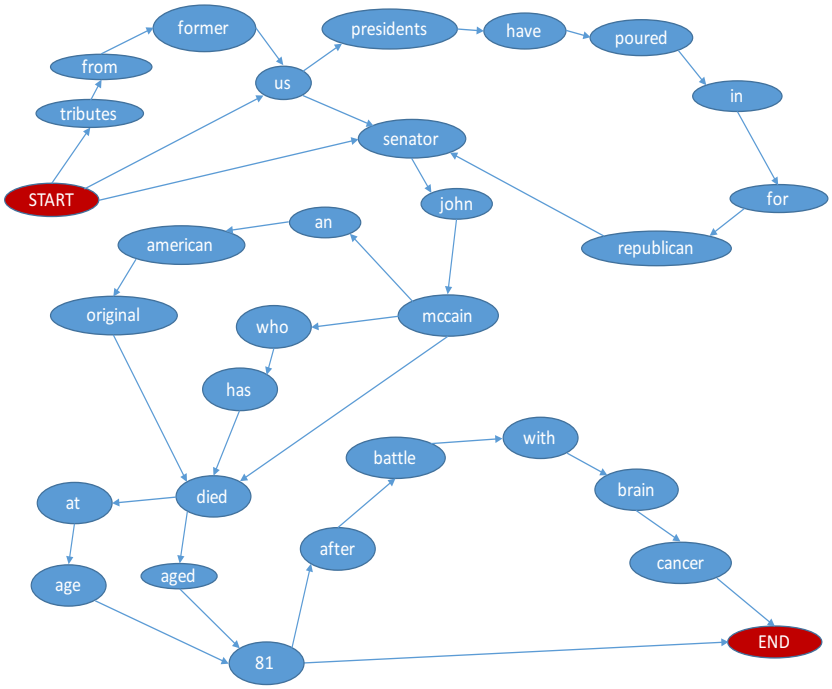


Figure 5.1: Graphical illustration of word graph approach. Nodes are words, while edges indicates a direct succession relation of words. The graph starts from node START to node END. The weight on each edge is omitted for clarity.

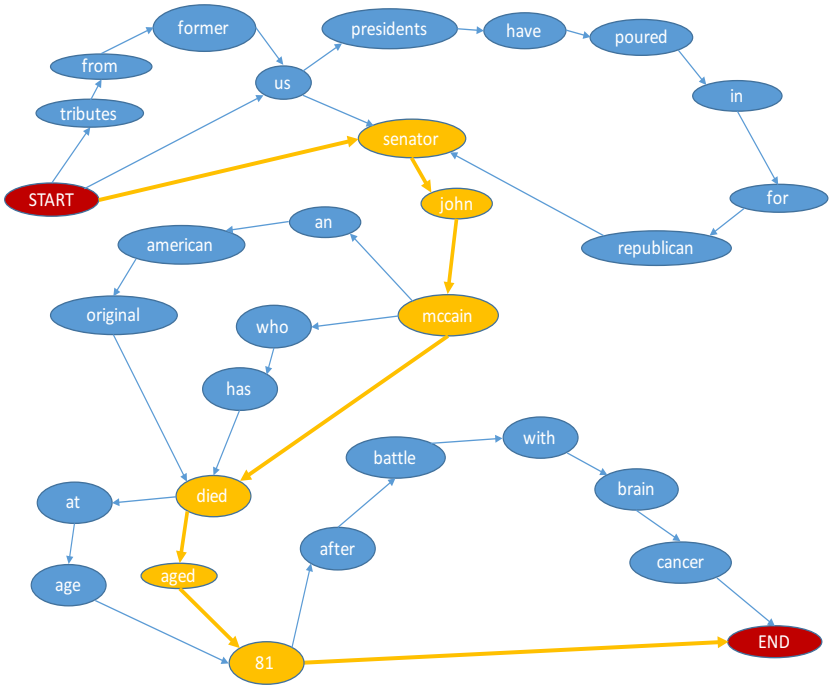


Figure 5.2: Graphical illustration of a plausible compression candidate whose path is in orange color, *Senator John McCain died aged 81.*, starting from node START to node END.

5.3 Dataset Construction

The largest existing English corpus for multi-sentence compression is the Cornell corpus [55], which has only 300 instances. We introduce herein a large-scale dataset by compiling the English Gigaword[3]. After pre-processing (e.g., filtering strange punctuations), 1.37 million news articles were yielded to group related sentences. The full procedure to obtain this dataset is given here[4].

Group Related Sentences
The prerequisite for multi-sentence compression is that all input sentences should be related to the same topic or event. Inspired by [55], if the sentences are too similar, one of the input sentences could be directly treated as a compression. In contrast, if the sentences are too dissimilar (no interaction), they may describe different events or topics. Both cases should be avoided because sentence compression would not be necessary. Here we use bi-gram similarity, which exhibited the highest accuracy (90%)[5]. We empirically arrived at 0.2 of the lower threshold of the bigram similarity to avoid very dissimilar sentences and 0.7 of the upper threshold of the bigram similarity to avoid near-identical sentences. As presented in Table 5.2, 140,572 sentence groups were finally yielded out of 1.37 million new articles. We refer to this as the Giga-MSC dataset.

# of sentences in a group	# of groups
2	133,123
3	6,633
4	816
In total	140,572

Table 5.2: Statistics of created Giga-MSC dataset.

GigaMSC Dataset Annotation
We randomly selected 150 sentences for human annotation, which were used as reference compression in the automatic evaluation. Two annotators[6] were asked to generate a single reduced grammatical compression that satisfies two conditions: (1) conveys the important content of all input sentences and (2) should be grammatically correct. We were interested in how the human annotators performed this task without vocabulary constraints. Hence, we did not tell them to introduce new vocabulary in their compression as several previous works did [10, 48]. Inter-agreement score Fleiss' Kappa [4] was also computed. The score was 0.43, demonstrating that moderate agreement was reached.

5.4 Methodology

This section introduces an unsupervised neural rewriter to improve the quality of generation by virtue of paraphrasing and neural text generation techniques. We are motivated by the fact that human annotators use novel words or synonyms to create a summary and, thus, believe that introducing new words makes the

[3]https://catalog.ldc.upenn.edu/LDC2011T07 English Gigaword, a comprehensive archive of newswire text data containing seven distinct international sources.

[4]http://github.com/code4ai/data

[5]Human judges were asked to evaluate whether the sentences in a group revolved around the same topic or event. A total of 45 out of 50 sentence groups were judged to be qualified.

[6]Both annotators are native English speakers and not authors of this paper.

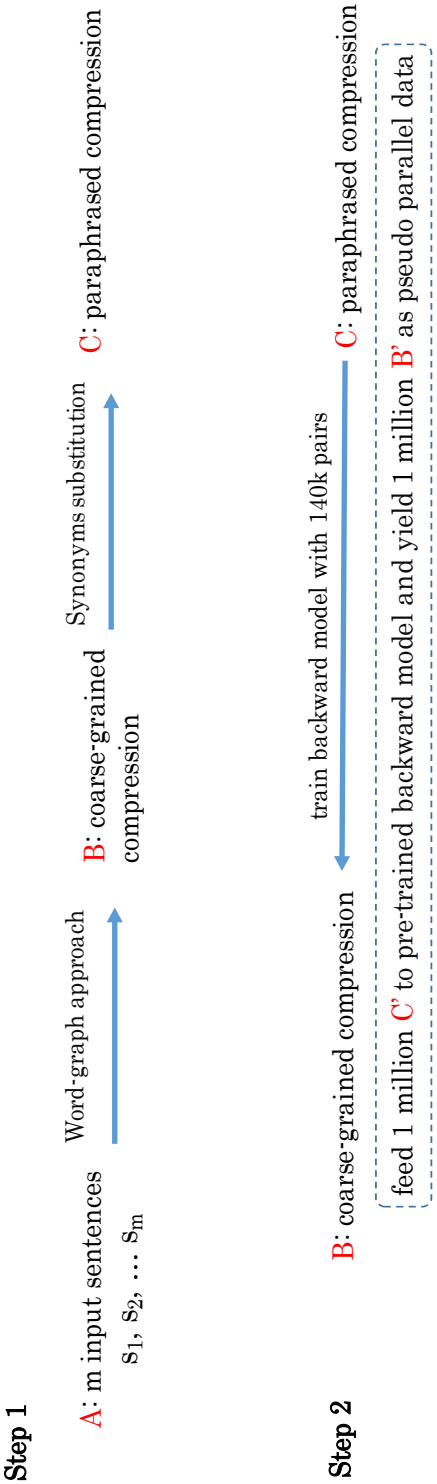


Figure 5.3: Graphic illustration for our rewriter model. A refers to multiple input sentences. B denotes a single compressed sentence using the word graph approach. C is the paraphrased sentence. C′ is a large-scale and in-domain monolingual corpus, while B′ refers to the predicted compression by a pre-trained backward model given as C′. B + B′ and C + C′ are the mixing datasets from both *step*.1 and *step*.2, respectively.

compression more human-like. Moreover, a context-unaware synonymous substitution yields ungrammatical compression; therefore, we hope that the neural language model (decoder) could help correct it in the generation progress. Figure 5.3 illustrates the whole process composed of four steps. We elaborate each one below.

Step.1(A→B):
Given m input sentences, $s_1, s_2, ..., s_m$, called A, we use the word graph approach to obtain coarse-grained compression, called B. We consider herein the keyphrase-based word graph (KWG) approach [10].

Step.1 (B→C):
C is yielded by substituting words and phrases in B with synonyms. We first identified all the multiword expressions in a sentence and determined all the synonyms in WordNet3.0[7]. Keep in mind that our goal is to shorten the sentence as much as possible, we specifically substituted multiword expressions, such as *police officer*, *united_states_of_america*, with their **shorter** synonyms policeman and u.s.. Because the size of synonyms in the WordNet dictionary is relatively limited, we also exploit PPDB 2.0[8] to replace the nouns, verbs, and adjectives with their shorter counterparts. For example, the verb *demonstrating* is converted into *proved*. By using the Giga-MSC dataset we created, 140,000 (A, B, C) tuples are yielded. Lexical substitution might lead to disfluent C but significantly increases the number of novel words. Therefore, the next steps focus on creating pseudo parallel data to boost the fluency of C while attempting to maintain the rate of novel words.

Step.2 (C→B):
Because the yielded B and C are semantically equivalent, we train a backward model (C) using 140,000 (C, B) pairs. The backward model consisted of a three-layer bi-directional LSTM encoder and a uni-directional decoder with attention mechanism. After the backward model was trained, one million grammatical in-domain sentences C′ were given as input to generate one million B′ The average length of C′ was similar to that of C (30.2 tokens). We also found that C′ maintained a novel rate of approximately 8.9, as compared to B′.

Step.3(B+B′ →C+C′):
We merge the training data (coarse-grained compression B and non-fluent paraphrasing compression C) and the pseudo parallel data (pseudo sentence B and grammatical sentence C) to jointly learn a forward model that consisted of a three-layer LSTM encoder and decoder. The vocabulary and word embedding were shared between both backward and forward models. We expect that because the grammatical C accepts the majority of training data, it will improve the fluency of C.

Encoder
Formally, we converted the source sequence consisting of $[x_1, x_2, ..., x_n]$ into the following word representation:

$$[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n] = [\mathrm{W_e x_1}, \mathrm{W_e x_2}, ..., \mathrm{W_e x_n}]. \quad (5.1)$$

[7] https://wordnet.princeton.edu/
[8] https://paraphrase.prg

This is followed by a bi-directional long short-term memory neural network [36]. W_e refers to the word-embedding matrix.

$$\mathbf{h}_{\mathrm{i}}^{\mathrm{fwd}} = \mathrm{LSTM}(\mathbf{e}_{\mathrm{i}}, (\mathbf{h}_{\mathrm{i-1}}^{\mathrm{fwd}}, \mathbf{c}_{\mathrm{i-1}}^{\mathrm{fwd}})), \tag{5.2}$$

$$\mathbf{h}_{\mathrm{i}}^{\mathrm{bwd}} = \mathrm{LSTM}(\mathbf{e}_{\mathrm{i}}, (\mathbf{h}_{\mathrm{i+1}}^{\mathrm{bwd}}, \mathbf{c}_{\mathrm{i+1}}^{\mathrm{bwd}})). \tag{5.3}$$

The concatenation of the hidden states from both directions $[\mathbf{h}_i^{fwd}; \mathbf{h}_i^{bwd}]$ can be viewed as the representation of the source sentences, which we note as $\mathbf{h}_i$.

Decoder
The decoder is uni-directional long short-term memory neural network coupled with attention mechanism. Given the target sequence B, $[y_1, y_2, ..., y_t]$,

$$\mathbf{s}_{\mathrm{t}} = \mathrm{LSTM}(\mathrm{y}_{\mathrm{t}}, \mathbf{s}_{\mathrm{t-1}}), \tag{5.4}$$

$$\mathrm{P}(\mathrm{y}_{\mathrm{t}}|\mathrm{y}_{<\mathrm{t}}, \mathrm{X}) = \mathrm{softmax}(\mathbf{s}_{\mathrm{t}}, \mathbf{c}_{\mathrm{t}}), \tag{5.5}$$

where $\mathbf{s}_i$ is the hidden state of the decoder, and $\mathbf{c}_t$ is computed as follows:

$$\mathbf{c}_{\mathrm{t}} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \alpha_{\mathrm{i}} \mathbf{h}_{\mathrm{i}}, \tag{5.6}$$

$$\alpha_{\mathrm{i}} = \frac{\exp(\mathrm{m}(\mathbf{s}_{\mathrm{t}}, \mathbf{h}_{\mathrm{i}}))}{\sum_{\mathrm{j=1}}^{\mathrm{n}} \exp(\mathrm{m}(\mathbf{s}_{\mathrm{t}}, \mathbf{h}_{\mathrm{j}}))}, \tag{5.7}$$

where m is a bilinear neural network learned during training, and α_i denotes the attention weights. The backward model is used in step 3 after being trained.

5.5 Experiments

5.5.1 Datasets

In this work, we used two datasets to evaluate our models. First is the Giga-MSC dataset, which was detailed in Chapter 5.3. A total of 150 annotated sentences were used as the ground truth for testing. Second is the Cornell dataset [55], which consists of 300 sentence groups with five manually created references each.

5.5.2 Baseline Approach

Word Graph (WG) Model
The word graph approach is initialized in [27], and is an unsupervised approach for generating compression from several input sentences. A K-shortest path algorithm is used to find the best path (compression). We considered the original word graph approach as the first baseline approach (#1).

Keyphrase-based Word Graph (KWG) Model
To produce a more informative compression, the keyphrase-based word graph model [10] re-ranks the K-shortest candidate paths to find paths that retain more important words in the input sentences. We considered it as the second baseline approach (#2).

Hard Paraphrasing (Hard-Para)
The hard paraphrasing (Hard-Para) approach (#3) directly substituted words and phrases with their shorter synonyms by using WordNet and PPDB 2.0 (size M is chosen with 463,433 paraphrasing pairs). We also take it as a comparison approach.

Seq2seq
The sequence-to-sequence learning model (#4) consists of a bi-directional LSTM encoder and a uni-directional LSTM decoder alongside attention mechanisms. This model was trained using (B, C) pairs.

Denoising Autoencoder (DA)
The denoising autoencoder is an extension of the classical autoencoder by adding noise in a sentence to decode itself. [26] applied it to single sentence compression. The goal is to shorten sentences; hence, we used a paraphrase dictionary, such as PPDB 2.0, to expand the length of the sentence by substituting words and phrases with **longer** ones. We then investigated how the paraphrase amount affects the compression generation by using different size of PPDB. We note the approach using M-size PPBD as (#5.1) and that using XL-size as (#5.2). The sentences were yielded by the KWG model as the target and its noisy (synonymous) counterpart as the source to train the sequence-to-sequence learning model.

5.5.3 Training details

Both backward and forward models were three-layer LSTMs with $1,024$ hidden cells for each layer. The hidden cells for the encoder were 512 because it was bi-directional. The embedding size was set to 768, while the vocabulary size was set to $50,000$. The batch size was chosen from [64, 128]. The learning rate was set to 0.0001, and Adam optimizer was used. All parameters were initialized by sampling from the normal distribution of mean 0 and variance 1. Gradients were clipped to avoid gradient explosion with a threshold of 5. We used pre-trained word embeddings from BERT [23], where $22,225$ words out of $50,000$ were found. We did not use BERT to encode the input sentence, although we still observed that the pre-trained word embeddings from BERT sped up the model convergence. With respect to the word graph approach, we used this implementation[9]. The number of candidates in the word graph was set to 50 because we need to produce approximately $140,000$ coarse-grained compression, and the minimal number of tokens in the compression is 10.

Out-of-Vocabulary (OOV) Word Handling
Both datasets were from the news domain; hence, there are many organizations, names, etc. out of vocabulary. We tackled this problem by following the approach below [26]. This approach is effective in our practice in spite of the existence of other approaches, such as [68]. Given an input sequence, we first identified all OOV tokens and numbered them in order. We stored the map from the numbered OOV tokens (e.g., OOV1 and OOV2) to words. The corresponding word embeddings were also assigned to each numbered OOV token. We then applied the same numbering system to the target. At inference, we replaced any

[9] https://github.com/boudinfl/takahe

output OOV tokens with their corresponding words using the map we stored beforehand, which allowed us to produce words that were not in the vocabulary.

5.6 Results and Analysis

5.6.1 Automatic Evaluation

We used the METEOR metric, which is based on n-gram overlap with synonyms, for the automatic evaluation. The Novel n-gram rate[10] (e.t., NN-1, NN-2, NN-3, and NN-4) was also computed to investigate how many novel words the models could introduce. Table 5.3 shows the results for the Giga-MSC dataset, while Table 5.4 shows the results for the Cornell dataset. Our observations are as follows:

1. The keyphrase word graph approach (#2) is a strong baseline according to the METEOR metric. In comparison, the proposed rewriter (#6) yields comparable result on the METEOR metric for the Giga-MSC dataset but lower result for the Cornell dataset. We speculate that it may be due to the difference in the ground-truth compression. 8.6% of novel unigrams exist in the ground-truth compression of the Giga-MSC dataset, while only 5.2% of novel unigrams exist in that of the Cornell dataset.

2. HardPara.(#3), Seq2seq (#4), Denoising auto (#5), and our rewriter (#6) significantly increase the number of novel n-grams, and the proposed rewriter (6) seemed to be a better trade-off between the information coverage (measured by METEOR) and the introduction of novel n-grams across all methods.

Model	Meteor	NN-1	NN-2
Ground truth	-	8.6	28.0
#1 WG (Filippova, 10)	0.29	0.0	0.0
#2 KWG (Boudin+, 13)	0.36	0.0	0.0
#3 Hard Para.	0.35	10.1	19.7
#4 Seq2seq with attention	0.33	12.7	24.0
#5.1 Denoising auto. (m)	0.25	29.3	49.1
#5.2 Denoising auto. (xl)	0.24	28.1	47.7
#6 Our rewriter	**0.36**	9.0	17.4
Model	NN-3	NN-4	CR
Ground truth	40.0	49.1	0.50
#1 WG (Filippova, 10)	2.8	6.8	0.34
#2 KWG (Boudin+, 13)	1.1	3.1	0.52
#3 Hard Para.	29.1	38.0	0.51
#4 Seq2seq with attention	34.7	44.4	0.49
#5.1 Denoising auto. (m)	64.2	75.4	0.52
#5.2 Denoising auto. (xl)	62.9	73.5	0.52
#6 Our rewriter (RWT)	25.7	33.8	0.50

Table 5.3: Results for the Giga-MSC dataset.

[10]Novel n-gram rate $= 1 - \frac{|S \cap C|}{|C|}$ where S refers to set of words from all input sentences while C refers set of words from compression.

Model	Meteor	NN-1	NN-2
Ground truth	-	5.2	15.8
#1 WG (Filippova, 10)	0.33	0.0	1.7
#2 KWG (Boudin+, 13)	**0.45**	0.0	1.8
#3 Hard Para.	0.38	9.2	19.0
#4 Seq2seq with attention	0.37	8.4	18.3
#5.1 Denoising auto. (m)	0.25	27.5	47.1
#5.2 Denoising auto. (xl)	0.24	31.0	51.6
#6 Our rewriter	0.40	8.1	17.0
Model	NN-3	NN-4	CR
Ground truth	23.2	29.6	0.49
#1 WG (Filippova, 10)	5.5	9.8	0.34
#2 KWG (Boudin+, 13)	4.6	8.0	0.52
#3 Hard Para.	28.7	37.7	0.50
#4 Seq2seq with attention	27.6	36.3	0.52
#5.1 Denoising auto. (m)	62.1	72.6	0.49
#5.2 Denoising auto. (xl)	67.1	77.4	0.49
#6 Our rewriter (RWT)	26.0	34.3	0.50

Table 5.4: Results for the Cornell dataset.

3. On comparing with Seq2seq (#4) and our rewriter (#6), we found that adding pseudo data helps to decrease the novel words rate and increase the METEOR score on both datasets.

4. As for the denoising auto-encoder (#5), we varied the amount of paraphrasing using either the middle-sized PPDB package (#5.1) or the XL-sized one (#5.2). Both yielded the highest novel n-gram rate but the lowest results on the METEOR metrics, suggesting that the two much novel n-grams give raise to a drop in performance.

Method	Informativeness	Grammaticality
KWG	1.06	1.19
RWT	1.02	1.40*

Table 5.5: Human evaluation for informativeness and grammaticality. * stands for significantly better than KWG with 0.95 confidence.

5.6.2 Human Evaluation

The METEOR metric cannot measure the grammaticality of compression; hence, we asked two human raters[11] to assess 50 compressed sentences out of the Gigа-MSC test dataset in terms of informativeness and grammaticality. With respect to grammaticality, we followed the proceeding works [7, 27] and asked the raters to give three ratings (points): excellent (2 points) if the generated compression was a completely grammatical sentence; good (1 point) if the generated compression is basically readable and needs a minor correction; and ungrammatical (0 point) if it is none of the above.

[11] Both raters are native English speakers and not authors of this paper.

Similar to the evaluation scheme of grammaticality, informativeness was assessed by giving three ratings as well: excellent (2 points) if the generated compression conveys the gist of the main event or topic; good (1 point) if it is related to the main theme, but misses something important; and unrelated (0 point) if the generated compression is not related to the main theme.

Method	Informativeness			Grammaticality		
	0	1	2	0	1	2
KWG	11%	72%	17%	5%	71%	24%
RWT	14%	67%	18%	2%	60%	38%

Table 5.6: Distribution over the possible manual ratings for informativeness and grammaticality. The ratings are expressed on a scale of 0 to 2.

Table 5.5 shows the average ratings for informativeness and readability, while Table 5.6 presents the distribution over the possible ratings. From that, we found that our rewriter significantly improved the grammaticality of compression in comparison with the keyphrase word graph approach, implying that the pseudo data may contribute to the language modeling of the decoder, thereby improving the grammaticality.

5.6.3 Context Awareness Evaluation

Because several novel words were introduced in Hard Para. (#3), Seq2seq (#4), and our rewriter (#6), we were interested to determine whether the compressions generated by these models were context-aware. We herein considered the out-of-the-box context-aware encoder, BERT [23]. The evaluation proceeded as follows:

As for a sentence with N words, $\mathrm{S} = [\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_\mathrm{N}]$, we sequentially masked each word at a time. BERT will output a probability distribution over all possible words. As knowing the current word w_i, we took the corresponding probability in distribution as the context awareness score of this word. We averaged all the words in a sentence S as the context awareness score of the sentence using the following formula:

$$\mathrm{CXT(S)} = -\frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \log \mathrm{p}(\mathrm{w_i}|\mathrm{c}), \tag{5.8}$$

where $\mathrm{c} = [\mathrm{w}_1, ...\mathrm{w}_{\mathrm{i}-1}, \mathrm{w}_{\mathrm{i}+1}, ..., \mathrm{w}_\mathrm{n}]$. We used this implementation[12].

Method	Giga-MSC		Cornell	
	Base	Large	Base	Large
Hard Para.	354.6	473.6	273.1	316.7
Seq2seq	249.1	219.1	326.1	388.3
Ours	**148.5**	**158.4**	**203.9**	**277.4**

Table 5.7: Context awareness scores computed using the out-of-the-box contextual language model. Base and Large refer to the different model configurations of BERT.

The low pseudo likelihood CTX(S) may suggest a better context awareness (e.t., the lower, the better). As presented in Table 5.7, the proposed rewriter achieves the lowest likelihood on both datasets, thereby indicating better context awareness in its generated compression.

[12] https://github.com/xu-song/bert-as-language-model

5.6.4 Case Study

We herein illustrate the proposed rewriter with some examples. As shown in Table 5.8, there are two example cases and each contains two input sentences as well as two compression outputs produced by keyword-based word graph approach (KWG) and our Rewriter (RWT) respectively. For the first case, as for the output c1 produced by the KWG, its nominal subject, *Sembcorp Marine*, lacks inflective verb, making it unreadable. By comparison, the Rewriter of the output d1 removed the conjunction, *after*, and the clause subject, *shareholders*, forming a readable sentence. However, the informativeness of d1 may drop compared to c1 because it is the *shareholders* that rejected Sembcorp industries' privatisation offer, not the company, *Sembcorp Marine*. With respect to the second case, the RWT of output d2 corrected the ungrammatical parts (e.t., *accused Tokhtakhounov, accused of,*) in c2. In the meanwhile, novel words (in bold) are introduced by paraphrasing was not always satisfactory because phrases such as *Salt Lake City* are fixed collocations, which should not be rewritten.

a. Sentence$_1$ **b.** Sentence$_2$ **c.** Output of KWG **d.** Output of RWT	
a1:	Singapore stocks closed 0.3 percent lower monday with investors focused on Sembcorp Marine after shareholders rejected a privatisation offer by parent Sembcorp industries, dealers said.
b1:	Sembcorp Marine shares fell 26 cents to 0.84, after shareholders rejected Sembcorp industries' privatisation offer, dealers said.
c1:	Sembcorp Marine after shareholders rejected Sembcorp industries' privatisation offer, dealers said.
d1:	Sembcorp Marine rejected Sembcorp industries' privatisation offer, dealers said.
a2:	Alleged Russian mobster Alimzhan Tokhtakhounov, accused of conspiring to fix skating events at the 2002 Winter Olympics in Salt Lake City, has returned to Moscow, the Kommersant daily reported Wednesday.
b2:	US prosecutors accused Tokhtakhounov of conspiring to fix the artistic events at the Salt Lake City games with the assistance of the French and Russian judges.
c2:	US prosecutors accused Tokhtakhounov, accused of conspiring to fix the artistic skating events at the Salt Lake City, has returned to Moscow, the Kommersant daily reported Wednesday.
d2:	Tokhtakhounov, accused of **conspiracy** to fix the artistic skating events at the Salt Lake **Town**, has returned to Moscow, the Kommersant daily reported.

Table 5.8: Example cases and the corresponding compression results. KWG refers to word-graph approach baseline, while RWT refers to the proposed Rewriter.

5.7 Method Limitation

There exists two limitations in the proposed approach. First, the rewriter need the output from the word graph approach as the input, which makes it not in an end-to-end fashion; second, some fixed collocations might be also paraphrased as shown in the case study. The controllable paraphrasing is thus needed to avoid these cases.

5.8 Conclusion and Future Work

In this work, we tackled the multi-sentence compression problem with a specific focus on rewriting the coarse-grained compression. To overcome the lack of a parallel corpus, we introduced a relatively large-scale dataset, namely the Giga-MSC dataset, to make the neural network-based model readily be applied. A backward model was proposed to be first trained on sentences and their paraphrasing counterpart. We then yielded large-scale pseudo parallel data to do joint training. The experimental results showed that compared with the baselines, our model generated more grammatical sentences with many novel n-grams compared with the extraction-based word graph approach. A further analysis on the context awareness validated the effectiveness of our rewriter model. However, more experiments are still needed to investigate how far this pre-trained language model can go.

As a future step, a semantic units-based graph instead of the word graph is more appealing to solving the problem that the same meaning might have multiple expressions. For example, Barack Obama is the 44th president of the U.S. and also the first African American president. These three terms, *Barack Obama, the 44th president of the U.S., the first African American president*, correspond to the same sense.

Chapter 6

Conclusion and Future Direction

In this chapter, we conclude the contribution of this thesis. In Chapter 6.1, we provided the answers to our four research questions following by a broad discussion; Chapter 6.2 describes the limitation of the proposed approaches. We end up with the discussion of the future directions in Chapter 6.3.

6.1 Summary of Contribution

In this thesis, we start by clarifying the current challenges facing the sentence compression research. Three critical evaluation aspects related to the task were highlighted, namely, readability (linguistic quality) of compression, informativeness (content quality) of compression, and compression rate (practical needs). Then, we analyzed the shortcomings of previous studies; (i) for single sentence compression, the rule-based approaches are able to define rules covering all compression cases, while the data driven-based approach leaves a gap between the optimization objective and evaluation. (ii) For multiple sentence compression, the existing technique is mostly extraction-based, giving raise to that the high informative summary that often contains more content is dropping in the readability. Based on the above-mentioned observations, we believe the hybrid approaches augmented neural network-based approach with linguistic knowledge could provide a promise for producing readable and informative compression. We thus raised four research questions and discuss them as follow:

1. Whether sentence compression neural models could be enhanced by incorporating word-level and sentence-level linguistic knowledge? And, how they contribute to the compression performance?

   Yes, incorporating word-level and sentence-level linguistic knowledge is beneficial to sentence compression task. In Chapter 3, we show that incorporating part-of-speech tag, dependency relation label, as well as named entity tag enables the neural models to be more predictive of word deletion process. Among others, dependency relation label is a strong indicator for GOOGLE Corpus, while part-of-speech tag is a strong indicator for other benchmark datasets. Also, dependency labels and part of speech tags are relatively independent features for sentence compression task because combining both features yields better results than each of them alone. In Chapter 4, structure-level linguistic knowledge, e.t., dependency structural bias from syntactic trees, also shows improvement for GOOGLE corpus. We refer readers to Chapter 4 for the details.

2. What would be the relationship among readability, informativeness, and compression rate in order to ensure the good quality of the compression?

   From the correlation analysis in Chapter 4.2, the compression rate, defined as compression length over source length, correlates positively and highly with the human informativeness judgments, reaching a strong correlation level. This is, to a large degree, in line with the previous study [58] and the intuition: the more words the compression contains, the more information it conveys. However, unlike informativeness, readability only shows a weak correlation with human readability judgments. It indicates that compared to informativeness, readability is more independent across different human rates. Furthermore, it is worth noting that since compression rate, to some extent, is predictive of the informativeness and readability, performance comparison of different systems should always be in the same or at least similar compression rate. In light of this observation, we move forward with the comparison under a similar compression rate. Interestingly, we found that JS-divergence between source sentence and reference compression correlates the best with the human judgments among other quantities like KL-divergence, holding a promise for automatic informativeness assessment (kindly refer to Chapter 4.2 for more details). The thesis is also the first work investigating candidate quantity for automatic informativeness assessment in the context of sentence compression.

3. How can we model and optimize both readability and informativeness in a more explicit way?

   The previous studies put effort in modeling and optimizing readability but mostly in a post hoc way. For example, [76] trained long short-term memory networks with maximum likelihood estimation objective function; Then, it takes the probability of word, together with integer linear programming method to search the most probable compression. By comparison, the thesis presents a novel evaluator capable of explicitly optimizing the readability through a language model module and compression rate control module. Nevertheless, informativeness in the context of sentence compression is still challenging to model due to the inherent subjectivity of this metric. A possible solution could be using extrinsic evaluation that to some extent sidesteps this difficulty, which we will discuss later in the future work.

4. Given the lack of training data, can we overcome the conflict between readability and informativeness for multiple sentence compression using deep learning techniques?

   Yes. In chapter 5, we propose an unsuperivsed rewriter to boost the readability while maintaining the informativeness. In particular, a coarse-to-fine rewriter uses external knowledge, e.t., WordNet and Paraphrasing Database to polish the compression into more readable compression. We are also aware of that this is

the first study applying deep learning techniques to multiple sentence compression. Also, a large-scale parallel corpus was yielded for the future research.

6.2 Limitation

The proposed methods are dedicated to produce readable and informative compression. However, there exist two technical issues when applying our sentence compression model. First, users do not have the hard control over the length of compression output, though they have the control over the compression rate. This might make it impractical in some applications. For example, the headline generation might have the length constraints, e.g., less than 75 byte or 10 tokens.

Secondly, when it comes to multiple sentence compression, as shown in Figure 6.1, our unsupervised rewriter still need the output from word graph approach as its input sentence.

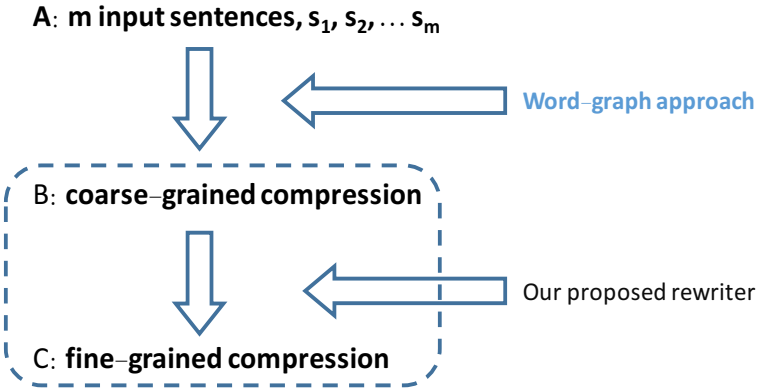


Figure 6.1: Our rewriter model, based on the output of word-graph approach, produces the fine-grained compression.

6.3 Future Directions

There are several possible directions in the sentence compression research as follow:

First, semantic-level sentence compression. The ideal sentence compression requires not only the natural language understanding of content but also natural language generation. Such kind of understanding should be at semantic level. There are at least two benefits:

- Commonsense knowledge-based sentence compression. Considering the following case:

  SENTENCE: In 2020, Japan will hold the summer Olympic Games for the second time, according to news report.

  If this is a sentence to be compressed, there are several plausible candidate compression as follow.

  CANDIDATE[1]: In 2020, Tokyo will hold the Olympic Games for the second time.
  CANDIDATE[2]: In 2020, Tokyo will hold the Olympic Games.

CANDIDATE[3]: Tokyo will hold the summer Olympic Games.

If we have the commonsense knowledge that the second time Tokyo holds the Olympic Games is 2020, information redundancy can be detected and either "in 2020" or "for the second time" can be deleted without the information loss. By comparison, CANDIDATE[3] losses timing information. In other words, the commonsense knowledge can help identify whether one piece of information entails another. The recent development in commonsense knowledge modeling and its corresponding Graph Convolutional Networks (GCN) approach afford the possibility of incorporating the commonsense knowledge into neural sentence compression.

- Semantic units-based multiple sentence compression. The current approaches operate and treat uni-gram (word) as the basic semantic unit. However, the same semantics can have multiple expressions. For example, Barack Obama is the 44th president of the U.S. and also the first African American president. These three terms, *Barack Obama, the 44th president of the U.S., the first African American president*, correspond to the same semantics. If we can detect the different terms with the same meaning, it is more efficient to remove redundant information. One possible way is to construct a semantic meaning-based graph from multiple input sentences, and generate a summary from this graph, which is still an open question worth exploring.

Secondly, automatic readability assessment. To date, most of the automatic readability assessment rely on sentence perplexity of a pre-trained language model. The current exciting progress on pre-trained model like Generative Pre-trained Transformer [64] from OpenAI show its promise in text generation. In spite of its computational cost, how to make use of these huge models to boost the readability assessment would be another interesting direction.

References

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*, 2017.

[2] Reinald Kim Amplayo and Min Song. An adaptable fine-grained sentiment analysis for summarization of multiple short online reviews. *Journal of Data & Knowledge Engineering*, 110:54–67, 2017.

[3] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *Proceedings of the 2016th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2442–2452, 2016.

[4] Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Journal of Computational Linguistics*, 34(4):555–596, 2008.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 2015th International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1208–1214, 2015.

[7] Regina Barzilay and Kathleen R McKeown. Sentence fusion for multidocument news summarization. *Journal of Computational Linguistics*, 31(3):297–328, 2005.

[8] Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. Jointly learning to extract and compress. In *Proceedings of the 2011th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 481–490, 2011.

[9] Joachim Bingel and Anders Søgaard. Text simplification as tree labeling. In *Proceedings of the 2016th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 337–343, 2016.

[10] Florian Boudin and Emmanuel Morin. Keyphrase extraction for n-best reranking in multi-sentence compression. In *Proceedings of the 2013th North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 298–305, 2013.

[11] Ted Briscoe and John Carroll. Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, 2002.

[12] Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the rasp system. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 77–80, 2006.

[13] Luis Adrián Cabrera-Diego and Juan-Manuel Torres-Moreno. Summtriver: A new trivergent model to evaluate summaries automatically without human references. *Journal of Data & Knowledge Engineering*, 113:184–197, 2018.

[14] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014th conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[15] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[16] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 93–98, 2016.

[17] James Clarke and Mirella Lapata. Constraint-based sentence compression an integer programming approach. In *Proceedings of the 2006th COLING/ACL on Main conference poster sessions*, pages 144–151, 2006.

[18] James Clarke and Mirella Lapata. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 377–384, 2006.

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

[20] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[21] Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *Proceedings of the 2008th International Conference on Computational Linguistics (COLING)*, pages 137–144, 2008.

[22] Simon Corston-Oliver. Text compaction for display on very small screens. In *Proceedings of the 2001th North American Chapter of the Association for Computational Linguistics Workshop on Automatic Summarization*, pages 89–98, 2001.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[25] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1–13, 2009.

[26] Thibault Fevry and Jason Phang. Unsupervised sentence compression using denoising auto-encoders. *arXiv preprint arXiv:1809.02669*, 2018.

[27] Katja Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 2010th International Conference on Computational Linguistics (COLING)*, pages 322–330, 2010.

[28] Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with lstms. In *Proceedings of the 2015th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 360–368, 2015.

[29] Katja Filippova and Yasemin Altun. Overcoming the lack of parallel data in sentence compression. 2013.

[30] Katja Filippova and Michael Strube. Dependency tree based sentence compression. *The 2008th International Natural Language Generation Conference (INLG)*, pages 25–32, 2008.

[31] Katja Filippova and Michael Strube. Sentence fusion via dependency graph compression. In *Proceedings of the 2008th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 177–185, 2008.

[32] Boris A Galitsky, Josep Lluis De La Rosa, and Gábor Dobrocsi. Inferring the semantic properties of sentences by mining syntactic parse trees. *Journal of Data & Knowledge Engineering*, 81:21–45, 2012.

[33] Michel Galley and Kathleen McKeown. Lexicalized markov grammars for sentence compression. In *The 2007th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 180–187, 2007.

[34] Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast and optimal decoding for machine translation. *Journal of Artificial Intelligence*, 154(1-2):127–143, 2004.

[35] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 2:850–855, 1999.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Journal of Neural Computation*, 9(8):1735–1780, 1997.

[37] Hongyan Jing. Sentence reduction for automatic text summarization. In *Proceedings of the 6th conference on Applied natural language processing*, pages 310–315, 2000.

[38] Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv:1604.03357*, 2016.

[39] Kevin Knight and Daniel Marcu. Statistics-based summarization-step one: Sentence compression. In *Proceedings of the 7th National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710, 2000.

[40] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Journal of Artificial Intelligence*, 139(1):91–107, 2002.

[41] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 2001th International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[42] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174, 1977.

[43] Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of the 2014th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701, 2014.

[44] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[45] Chin-Yew Lin. Improving summarization performance by sentence compression—a pilot study. In *Proceedings of the 6th international workshop on Information retrieval with Asian languages*, pages 1–8, 2003.

[46] Elena Lloret, María Teresa Romá-Ferri, and Manuel Palomar. Compendium: A text summarization system for generating abstracts of research papers. *Data & Knowledge Engineering*, 88:164–175, 2013.

[47] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[48] An-Vinh Luong, Nhi-Thao Tran, Van-Giau Ung, and Minh-Quoc Nghiem. Word graph-based multi-sentence compression: Re-ranking candidates using frequent words. In *Knowledge and Systems Engineering (KSE)*, pages 55–60, 2015.

[49] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

[50] Inderjeet Mani. *Advances in automatic text summarization*. MIT press, 1999.

[51] André FT Martins and Noah A Smith. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the 2009th Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9, 2009.

[52] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[53] Ryan McDonald. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 297–304, 2006.

[54] Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564, 2007.

[55] Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. Time-efficient creation of an accurate sentence fusion corpus. In *Proceedings of the 2010th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 317–320, 2010.

[56] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[57] Amr Mousa and Björn Schuller. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 2017th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1023–1032, 2017.

[58] Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97, 2011.

[59] Mir Tafseer Nayeem and Yllias Chali. Paraphrastic fusion for abstractive multi-sentence compression generation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2223–2226, 2017.

[60] Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. Abstractive unsupervised multi-document summarization using paraphrastic sentence fusion. In *Proceedings of the 2018th International Conference on Computational Linguistics (COLING)*, pages 1191–1204, 2018.

[61] Tadashi Nomoto. Discriminative sentence compression with conditional random fields. *Information processing & management*, 43(6):1571–1587, 2007.

[62] Prasad Perera and Leila Kosseim. Evaluating syntactic sentence compression for text summarisation. In *International Conference on Application of Natural Language to Information Systems*, pages 126–139, 2013.

[63] Elvys Linhares Pontes, Stéphane Huet, Thiago Gouveia da Silva, Andréa carneiro Linhares, and Juan-Manuel Torres-Moreno. Multi-sentence compression with word vertex-labeled graphs and integer linear programming. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 18–27, 2018.

[64] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Amazonaws*, 2018.

[65] Sebastian Riedel and James Clarke. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the 2006th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 129–137, 2006.

[66] Stefan Riezler, Tracy H King, Richard Crouch, and Annie Zaenen. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) on Human Language Technology-Volume 1*, pages 118–125. Association for Computational Linguistics, 2003.

[67] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 379–389, 2015.

[68] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 2017th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1073–1083, 2017.

[69] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[70] Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond K Wong, and Fang Chen. An efficient approach for multi-sentence compression. In *Asian Conference on Machine Learning (ACML)*, pages 414–429, 2016.

[71] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[72] Cheri Speier, Joseph S Valacich, and Iris Vessey. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30(2):337–360, 1999.

[73] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[74] Vincent Vandeghinste and Yi Pan. Sentence compression for automated subtitling: A hybrid approach. *Text Summarization Branches Out*, pages 89–95, 2004.

[75] Xiaojun Wan and Jianguo Xiao. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 2008th International Conference on Computational Linguistics (COLING)*, pages 969–976, 2008.

[76] Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. Can syntax help? improving an lstm-based sentence compression model for new domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1385–1393, 2017.

[77] Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 2013th Annual Meeting*

of the Association for Computational Linguistics (ACL), pages 1384–1394, 2013.

[78] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. 1992.

[79] Christopher C Yang, Hsinchun Chen, and Kay Hong. Visualization of large category map for internet browsing. *Decision support systems*, 35(1):89–102, 2003.

[80] David Zajic, Bonnie J Dorr, Jimmy Lin, and Richard Schwartz. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing & Management*, 43(6):1549–1570, 2007.

[81] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*, 2016.

[82] Yang Zhao, Zhiyuan Luo, and Akiko Aizawa. A language model based evaluator for sentence compression. In *Proceedings of the 2018th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 170–175, 2018.

[83] Yang Zhao, Hajime Senuma, Xiaoyu Shen, and Akiko Aizawa. Gated neural network for sentence compression using linguistic knowledge. In *The 2017th International Conference on Applications of Natural Language to Information Systems*, pages 480–491, 2017.

[84] Yang Zhao, Xiaoyu Shen, Wei Bi, and Akiko Aizawa. Unsupervised rewriter for multi-sentence compression. In *Proceedings of the 2019th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2235–2240, 2019.

[85] Yang Zhao, Xiaoyu Shen, Hajime Senuma, and Akiko Aizawa. A comprehensive study: Sentence compression with linguistic knowledge-enhanced gated neural network. *Data & Knowledge Engineering*, 117:307–318, 2018.

Author Publications

Journal Papers:

1. Yang Zhao, Xiaoyu Shen, Hajime Senuma, Akiko Aizawa. A comprehensive study: Sentence compression with linguistic knowledge- enhanced gated neural network, Volume 117, Pages 307 – 318, Journal of Data & Knowledge Engineering (DKE 2018). May, 2018.

International Conferences:

1. Yang Zhao, Hajime Senuma, Xiaoyu Shen, Akiko Aizawa. Gated Neural Network for Sentence Compression using Linguistic Knowledge. The 22nd International Conference on Natural Language & Information Systems (NLDB 2017), pages 480 – 491, Liège, Belgium, June 20 – June 22, 2017.

2. Yang Zhao, Zhiyuan Luo, Akiko Aizawa. A Language Model based Evaluator for Sentence Compression. The 56th annual meeting of the Association for Computational Linguistics (ACL 2018), pages 170 – 175, Melbourne, Australia. July 15 – July 20, 2018.

3. Yang Zhao, Xiaoyu Shen, Wei Bi, Akiko Aizawa. Unsupervised Rewriter for Multi-Sentence Compression. The 57th annual meeting of the Association for Computational Linguistics (ACL 2019), pages 2235 – 2240, Florence, Italy. July 28 – Augest 2, 2019.