2021

Towards Improving the Robustness of Neural Abstractive Summarization

Kaiqiang Song
University of Central Florida

TOWARDS IMPROVING THE ROBUSTNESS OF NEURAL ABSTRACTIVE SUMMARIZATION

by

KAIQIANG SONG
B.S. Fudan University, 2016

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2021

Major Professor: Fei Liu Professor

ABSTRACT

Recent deep learning and sequence-to-sequence learning technology have produced impressive results on automatic summarization. However, the models have limited insights on the underlying language and it remains challenging for system-generated summaries to be truthful to the original input or cover the most important information. This is especially the case for generating abstractive summaries using neural models. My work aims for a flexible and controllable summarization system that can be adapted to cater to different scenarios. It is designed to incorporate linguistic structure information into deep neural networks, have the capability to produce abstracts by re-using a varying amount of source text, and take language characteristics into consideration for summary generation and selection.

My dissertation provides a comprehensive overview to the problem of text summarization. I will present a number of approaches to incorporate linguistic structure into state-of-the-art deep neural models to help system summaries remain grammatical and retain the most salient meaning of the source text. I will also describe a summarization approach that is controllable during training and produce diverse summaries during the decoding and re-ranking processes. Finally, I will conclude with a novel approach for selecting optimal summaries from a collection of candidates and discuss the opportunities and challenges in this promising area of research.

ACKNOWLEDGMENTS

I am deeply grateful to my supervisor Dr. Fei Liu for her invaluable guidance, continuous support and patience during my Ph.D study. Her immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank my mentors Dr. Bingqing Wang from Bosch, Dr. Chen Li and Dr. Xiaoyang Wang from Tencent for their support in my internships. Additionally, I would like to thank my friends, lab mates, my colleagues, Logan, Sangwoo, Kexin and Kristjan for a cherished time spent together in the lab. My appreciation also goes out to my family, especially my wife Xuejiao for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Motivation

The 21st century has an abundance of information. The latest technologies like the Internet and smart devices allow you to hear news anytime on this planet. You could obtain almost anything you need from a browser with a search engine. The Internet is becoming an immense library, full of knowledge, which not everyone could know.

Besides, the volume of total information is growing exponentially. As a matter of fact, we only had an estimate of about 3.2 million websites in 1999. In 2020, we had around 1.8 billion of them. It is a 560-time growing in the past two decades. The Internet users are not only its customers but also information creators. We write reviews for movies, products and services on E-Retailers. We record our life and share our opinions using online Medium and Social Network. People do business using the Internet. Advertising, selling, paying are all through the Internet.

As we create and receive different kinds of information from different sources daily, we are suffering the lack of information processing capacity as human beings. The solution to information collection, storage and analyzation is much more desired than before. How do we legally and properly collect the data? How do we safely and efficiently distribute such data? How do we

automatically precisely understand and make use of the data? These three questions especially the last one motivated me to conduct research on summarization.

1.2 Overview of Summarization

Automatic Summarization task aims to generate concise and informative summaries from a large collection of documents to better support fast browsing of textual content. There are two general approaches for summarization, extractive summarization and abstractive summarization. The extractive approaches extract content from the original text without any modification, while the abstract methods rephrase the input with a shortened summary.

1.2.1 Extractive Systems

The original idea was conceived in the 1950s, when Luhn [1] proposed a method of computing the significance of using word statistical information, such as word frequency and distribution, and then extracting sentences with highest scores to create an ”auto-abstract”.

Most of the extractive system follows this Information Extraction(IE) idea of first scoring the fragments of original text and then selecting them according to their order.

At the early stage, a branch of unsupervised systems [1, 2, 3] studied the salience of text fragments relying on human-designed features such as positional information, word frequency, key phrase indicators and IDF Scores. At that time, summarization was a sub-field of IE.

Later on, some researches [4, 5, 6] also took text structural and content relations into account. As more features and data became available, another branch of studies, the supervised methods [7, 8, 9] became the trend. Such methods attempted to predict a set of human annotated or rule-based binary labels indicating whether those fragments are selected or not.

In recent years, as computational capacity becomes much faster and cheaper, deep-learning based methods are getting more popular. Researchers start to use word embedding [10, 11, 12, 13], Deep Neural Networks[14, 15] instead of handcraft features. This helps in capturing the context. The extractive summarization is then formulated as a binary sequence labeling problem[16, 17, 18, 19].

1.2.2 Abstractive Systems

Unlike extractive systems, the abstractive summarization aims to comprehend and rephrase the original expressions with more understanding and less words. As mentioned in [20]:

> Most of these summarization approaches aim for selecting the most informative sentences, while less attempt has been made to generate abstractive summaries, or compress the extracted sentences and merge them into a concise summary. Simply

concatenating extracted sentences may not comprise a good summary, especially for spoken documents, since speech transcripts often contain many disfluencies and are redundant.

Before deep-learning shows its dominant performance, a set of methods [21, 20, 22, 23] tent to apply deletion based sentence compression after the extraction, removing redundancy. However, other methods focused on summary generations based on semantic representations [24, 25, 26].

Later, as sequence-to-sequence learning schema achieved a remarkable performance on generation tasks, several abstractive methods were proposed[27, 28, 29]. Such methods are impressive in generating fluent rephrased summaries. However, they are usually not true-to-original and sometimes ungrammatical. More specifically, such systems may create unexpected new meanings or bias the original one. We will discuss this *hallucination* issue and the way of reducing its effect in Chapter 2 & 3.

To date, pre-trained Language Models promote almost every downstream NLP tasks. To be more specific, BERT[30] and its variant RoBERTa[31] use fully visible Transformer based encoder to encode context from both left and right. GPTs[32, 33, 34] propose language model decoders with an auto-regressive way. UniLM[35] attempted to extend BERT with generation capability with a delicate attention mask. MASS[36], Bart[37], PEGASUS[38], ProphetNet[39] and T5[40] made attempts on different training strategies for efficiently pre-training on sequence-to-sequence tasks with a encoder-decoder Transformer. Such models are fed with huge amount of data (e.g. billions/trillions of tokens) containing lots of common sense knowledge, thus having a higher capability of abstraction.

1.2.3 Evaluation Metrics

To better evaluate the performance of a summarization system, researchers usually use one or more human created summaries as references.

Pyramid[41] is a metric designed for human evaluation with multiple references. A pyramid is built based on how many times a concept unit appears in different references. The more the concept appears, the larger the concept weighs. Human annotators will find out those concept units and align them among input, summary and references. Then, the summary will be evaluated using the weighted recall value of those concept units.

Instead of employing human annotators to figure out the concept units and align them, ROUGE[42] is proposed to automatically evaluate the n-gram overlap between summary and references. It is much easier when adapting to single reference case.

The above methods are designed for evaluating how much information is preserved in the summary compared to the references. However, such evaluation cannot measure the overall performance. Highly abstract metrics like Fluency, Grammaticality, Informativeness and Truthfulness are much more complicated. To evaluate the overall performance of a summary, researchers usually employ human annotators to rate the summary with these metrics.

1.2.4 Challenge of abstractive summarization systems

There are two major challenges for current summarization systems. First, current abstractive systems suffer from hallucination. Those systems are mostly data-driven, which remember the training data distribution very well; however they may not work well with new data. This could introduce the new meanings or bias the original meaning. Second, current abstractive summarization systems are lack of flexibility. They are mostly sequence-to-sequence models equipped with beam search method and decode the summaries in an auto-regressive manner. Such methods neither provide a natural way of length and copy-ratio controlling, nor generate diverse summaries. Lack of flexibility is also a key reason why abstractive summarization have not been widely used yet.

1.3 Content Layout

In Chapter 2, I will introduce an approach accepted at COLING 2018 for reducing the hallucination by incorporating structural information from the source side. In Chapter 3, I will illustrate a work accepted at AAAI 2020 also for reducing the hallucination and improving the grammaticality using target side structural information. In Chapter 4, I will describe a method on how to control the length and copy ratio of generated summaries. This work is also accepted at AAAI 2020. In Chapter 5, I will demonstrate a new pipeline of overgenerate-then-select for length and quality controlling. This could also extend to more desired properties of summary, coupled with customized selectors. This work is accepted at NAACL 2021. In Chapter 6, I will conclude my

research on summarization field, discuss the future work and my expectations for summarization in the future.

CHAPTER 2
INCORPORATE SOURCE STRUCTURAL INFORMATION:

STRUCTURE-INFUSED COPY MECHANISMS FOR ABSTRACTIVE

SUMMARIZATION

In this Chapter, I will introduce one of my work accepted at COLING 2018. In this work, we introduce a novel architecture that encourage salient source words/relations to be preserved in summaries. This helps the model to better learn those rare but important words/relations in the training data.

2.1 Background

Recent years have witnessed increasing interest in abstractive summarization. The systems seek to condense source texts to summaries that are concise, grammatical, and preserve the important meaning of the original texts [43]. The task encompasses a number of high-level text operations, e.g., paraphrasing, generalization, text reduction and reordering [44], posing a considerable challenge to natural language understanding.

Table 2.1: Example source sentences, reference and system summaries produced by a neural attentive seq-to-seq model. The main verbs are *italicized* and marked in red. System summaries fail to preserve summary-worthy content of the source (e.g., main verbs) despite their syntactic importance.

Src	A Mozambican man suspect of murdering Jorge Microsse, director of Maputo central prison, has ***escaped*** from the city's police headquarters, local media reported on Tuesday.
Ref	Mozambican suspected of killing Maputo prison director ***escapes***
Sys	mozambican man ***arrested*** for murder
Src	An Alaska father who was too drunk to drive ***had*** his 11-year-old son take the wheel, authorities said.
Ref	Drunk Alaska dad ***has*** 11 year old drive home
Sys	alaska father who was too drunk to drive

The sequence-to-sequence learning paradigm has achieved remarkable success on abstractive summarization [27, 28, 29, 45]. While the results are impressive, individual system summaries can appear unreliable and fail to preserve the meaning of the source texts. Table 2.1 presents two examples. In these cases, the syntactic structure of source sentences is relatively *rare* but perfectly normal. The first sentence contains two appositional phrases ("suspect of murdering Jorge Microsse," "director of Maputo central prison") and the second sentence has a relative clause ("who was too drunk to drive"), both located between the subject and the main verb. The system, however, fails to identify the main verb in both cases; it instead chooses to focus on the first few words of the source sentences. We observe that *rare syntactic constructions* of the source can pose

problems for neural summarization systems, possibly for two reasons. First, similar to *rare words*, certain syntactic constructions do not occur frequently enough in the training data to allow the system to learn the patterns. Second, neural summarization systems are not explicitly informed of the syntactic structure of the source sentences and they tend to bias towards sequential recency.

In this paper we seek to address this problem by incorporating source syntactic structure in neural sentence summarization to help the system identify summary-worthy content and compose summaries that preserve the important meaning of the source texts. We present structure-infused copy mechanisms to facilitate copying source words and relations to the summary based on their semantic and structural importance in the source sentences. For example, if important parts of the source syntactic structure, such as a dependency edge from the main verb to the subject ("father" $\leftarrow$ "had," shown in Figure 2.1), can be preserved in the summary, the "missing verb" issue in Table 2.1 can be effectively alleviated. Our model therefore learns to recognize important source words and source dependency relations and strives to preserve them in the summaries. Our research contributions include the following:

- we introduce novel neural architectures that encourage salient source words/relations to be preserved in summaries. The framework naturally combines the dependency parse tree structure with the copy mechanism of an abstractive summarization system. To the best of our knowledge, this is the first attempt at comparing various neural architectures for this purpose;
- we study the effectiveness of several important components, including the vocabulary size, a coverage-based regularizer [29], and a beam search with reference mechanism [46];

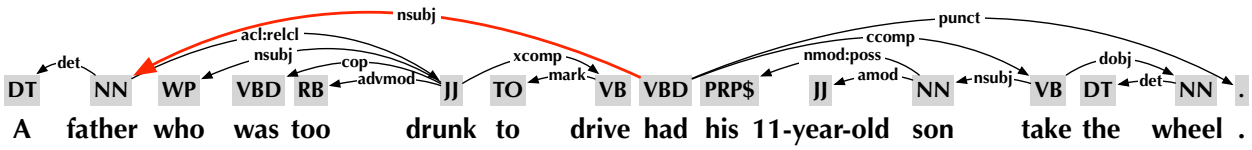

Figure 2.1: An example dependency parse tree created for the source sentence in Table 2.1. If important dependency edges such as "father ← had" can be preserved in the summary, the system summary is likely to preserve the meaning of the original.

- through extensive experiments we demonstrate that incorporating syntactic information in neural sentence summarization is effective. Our approach surpasses state-of-the-art published systems on the benchmark dataset.[1]

2.2 Related Work

Prior to the deep learning era, sentence syntactic structure has been utilized to generate summaries with an "*extract-and-compress*" framework. Compressed summaries are generated using a joint model to extract sentences and drop non-important syntactic constituents [47, 48, 49, 50], or a pipeline approach that combines generic sentence compression [51, 52, 53] with a sentence pre-selection or post-selection process [54, 55, 56, 22, 57]. Although syntactic information is helpful for summarization, there has been little prior work investigating how best to combine sentence syntactic structure with the neural abstractive summarization systems.

[1] We made our system publicly available at: `https://github.com/KaiQiangSong/struct_infused_summ`

Existing neural summarization systems handle syntactic structure only implicitly [58, 59, 60, 46, 45]. Most systems adopt a "*cut-and-stitch*" scheme that picks words either from the vocabulary or the source text and stitch them together using a recurrent language model. However, there lacks a mechanism to ensure structurally salient words and relations in source sentences are preserved in the summaries. The resulting summary sentences can contain misleading information (e.g., "mozambican man arrested for murder" flips the meaning of the original) or grammatical errors (e.g., verbless, as in "alaska father who was too drunk to drive").

Natural language generation (NLG)-based abstractive summarization [61, 62, 63, 64, 65] also makes extensive use of structural information, including syntactic/semantic parse trees, discourse structures, and domain-specific templates built using a text planner or an OpenIE system [66]. In particular, Cao et al. [67] leverage OpenIE and dependency parsing to extract fact tuples from the source text and use those to improve the faithfulness of summaries.

Different from the above approaches, this paper seeks to directly incorporate source-side syntactic structure in the copy mechanism of an abstractive sentence summarization system. It learns to recognize important source words and relations during training, while striving to preserve them in the summaries at test time to aid reproduction of factual details. Our intent of incorporating source syntax in summarization is different from that of neural machine translation (NMT) [68, 69], in part because NMT does not handle the information loss from source to target. In contrast, a summarization system must selectively preserve source content to render concise and grammatical summaries. We specifically focus on sentence summarization, where the goal is to reduce the first

sentence of an article to a title-like summary. We believe even for this reasonably simple task there remains issues unsolved.

2.3 Our Approach

We seek to transform a source sentence $\mathbf{x}$ to a summary sentence $\mathbf{y}$ that is concise, grammatical, and preserves the meaning of the source sentence. A source word is replaced by its Glove embedding [70] before it is fed to the system; the vector is denoted by $\mathbf{x}_i$ ($i \in [S]$; 'S' for source). Similarly, a summary word is denoted by $\mathbf{y}_t$ ($t \in [T]$; 'T' for target). If a word does not appear in the input vocabulary, it is replaced by a special '$\langle \mathrm{unk} \rangle$' token. We begin this section by describing the basic summarization framework, followed by our new copy mechanisms used to encourage source words and dependency relations to be preserved in the summary.

2.3.1 The Basic Framework

We build an encoder-decoder architecture for this work. An encoder condenses the entire source text to a continuous vector; it also learns a vector representation for each unit of the source text (e.g., words as units). In this work we use a two-layer stacked bi-directional Long Short-Term Memory [71] networks as the encoder, where the input to the second layer is the concatenation of hidden states from the forward and backward passes of the first layer. We obtain the hidden states of the second layer; they are denoted by $\mathbf{h}_i^e$. The source text vector is constructed by averaging over

all $\mathbf{h}_i^e$ and passing the vector through a feedforward layer with tanh activation to convert from the encoder hidden states to an initial decoder hidden state ($\mathbf{h}_0^d$). This process is illustrated in Eq. (2.2).

$$\mathbf{h}_i^e = f_e(\mathbf{h}_{i-1}^e, \mathbf{x}_i) \quad \mathbf{h}_t^d = f_d(\mathbf{h}_{t-1}^d, \mathbf{y}_{t-1}) \tag{2.1}$$

$$\mathbf{h}_0^d = \tanh(\mathbf{W}^{h_0} \frac{1}{S} \sum_{i=1}^{S} \mathbf{h}_i^e + \mathbf{b}^{h_0}) \tag{2.2}$$

A decoder unrolls the summary by predicting one word at a time. During training, the decoder takes as input the embeddings of ground truth summary words, denoted by $\mathbf{y}_t$, while at test time $\mathbf{y}_t$ are embeddings of system predicted summary words (i.e., teacher forcing). We implement an LSTM decoder with the attention mechanism. A context vector $\mathbf{c}_t$ is used to encode the source words that the system attends to for generating the next summary word. It is defined in Eqs (2.3-2.5), where $[\cdot||\cdot]$ denotes the concatenation of two vectors. The α matrix measures the strength of interaction between the decoder hidden states $\{\mathbf{h}_t^d\}$ and encoder hidden states $\{\mathbf{h}_i^e\}$. To predict the next word, the context vector $\mathbf{c}_t$ and $\mathbf{h}_t^d$ are concatenated and used as input to build a new vector $\widetilde{\mathbf{h}}_t^d$ (Eq. (2.6)). $\widetilde{\mathbf{h}}_t^d$ is a surrogate for semantic meanings carried at time step t of the decoder. It is

subsequently used to compute a probability distribution over the output vocabulary (Eq. (2.7)).

$$e_{t,i} = \mathbf{v}^\top \tanh(\mathbf{W}^e[\mathbf{h}_t^d || \mathbf{h}_i^e] + b^e) \tag{2.3}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i'=1}^{S} \exp(e_{t,i'})} \tag{2.4}$$

$$\mathbf{c}_t = \sum_{i=1}^{S} \alpha_{t,i} \mathbf{h}_i^e \tag{2.5}$$

$$\widetilde{\mathbf{h}}_t^d = \tanh(\mathbf{W}^h[\mathbf{h}_t^d || \mathbf{c}_t] + \mathbf{b}^h) \tag{2.6}$$

$$P_{vocab}(w) = \text{softmax}(\mathbf{W}^y \widetilde{\mathbf{h}}_t^d + \mathbf{b}^y) \tag{2.7}$$

The copy mechanism [72, 29] allows words in the source sequence to be selectively copied to the target sequence. It expands the search space for summary words to include both the output vocabulary and the source text. The copy mechanism can effectively reduce out-of-vocabulary tokens in the generated text, potentially aiding a number of applications such as MT [73] and text summarization [74, 75, 76].

Our copy mechanism employs a 'switch' to estimate the likelihood of generating a word from the vocabulary (p_{gen}) vs. copying it from the source text ($1 - p_{gen}$). The basic model is similar to that of the pointer-generator networks [29]. The switch is a feedforward layer with sigmoid activation (Eq. (2.8)). At time step t, its input is a concatenation of the decoder hidden state $\mathbf{h}_t^d$, context vector $\mathbf{c}_t$, and the embedding of the previously generated word $\mathbf{y}_{t-1}$. For predicting the next word, we combine the generation and copy probabilities, shown in Eq. (2.9). If a word w appears once or more in the input text, its copy probability ($\sum_{i:w_i=w} \alpha_{t,i}$) is the sum of the attention weights over all its occurrences. If w appears in both the vocabulary and source text, $P(w)$ is a weighted sum of the two probabilities.

$$p_{gen} = \sigma(\mathbf{W}^z[\mathbf{h}_t^d||\mathbf{c}_t||\mathbf{y}_{t-1}]) + b^z) \tag{2.8}$$

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_{t,i} \tag{2.9}$$

2.3.2 Structure-Infused Copy Mechanisms

The aforementioned copy mechanism attends to source words based on their "semantic" importance encoded in $\{\alpha_{t,i}\}$, which measures the semantic relatedness of the encoder hidden state $\mathbf{h}_i^e$ and the decoder hidden state $\mathbf{h}_t^d$ (Eq. (2.4)). However, the source syntactic structure is ignored. This is problematic, because it hurts the system's ability to effectively identify summary-worthy source words that are syntactically important. We next propose three strategies to inject source syntactic structure to the copy mechanism.

2.3.3 Learning Objective and Beam Search

We next describe our learning objective, including a coverage-based regularizer [29], and a beam search with reference mechanism [46]. We want to investigate the effectiveness of these techniques on sentence summarization, which has not been explored in previous work.

Learning objective. Our training proceeds by minimizing a per-target-word cross-entropy loss function. A regularization term is applied to the α matrix. Recall that $\alpha_{t,i} \in [0,1]$ measures the

Table 2.2: Six categories of structural labels. Example labels are generated for word 'had' in Figure 2.1. Relative word positions are discretized into ten buckets.

Structural info	**Example**
(1) depth in the dependency parse tree	0
(2) label of the incoming edge	'root'
(3) number of outgoing edges	3
(4) part-of-speech tag	'VBD'
(5) absolution position in the source text	9
(6) relative position in the source text	(0.5, 0.6]

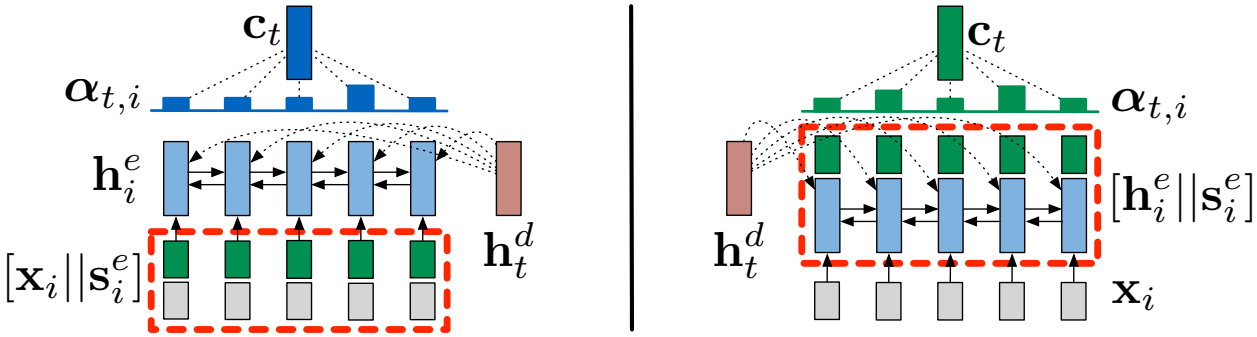

Figure 2.2: System architectures for 'Struct+Input' (left) and 'Struct+Hidden' (right). A critical question we seek to answer is whether the structural embeddings ($\mathbf{s}_i^e$) should be supplied as input to the encoder (left) or be exempted from encoding and directly concatenated with the encoder hidden states (right).

interaction strength between the t-th output word and the i-th input word. Naturally, we expect a 1-to-1 mapping between the two words. The coverage-based regularizer, proposed by See et al., [29], encourages this behavior by tracking the historical attention values attributed to the i-th input word (up to time step t-1), denoted by $\widetilde{\alpha}_{t,i} = \sum_{t'=0}^{t-1} \alpha_{t',i}$. The approach then takes the minimum between $\widetilde{\alpha}_{t,i}$ and $\alpha_{t,i}$, which has the practical effect of forcing $\alpha_{t,i}$ ($\forall t$) to be close to either 0 or 1, otherwise a penalty will be applied. The regularizer Ω is defined in Eq. (2.10), where M is the size of the mini-batch, S and T are the lengths of the source and target sequences. For two-way copy mechanisms, δ replaces α to become the new attention values, we therefore apply regularization to δ instead of α. When the regularizer applies, the objective becomes minimizing $(\mathscr{L} + \Omega)$.

$$\Omega = \lambda \sum_{m=1}^{M} \frac{1}{T^{(m)} S^{(m)}} \sum_{t=1}^{T^{(m)}} \sum_{i=1}^{S^{(m)}} \Big(\min(\widetilde{\alpha}_{t,i}, \alpha_{t,i}) \Big) \tag{2.10}$$

Beam search with reference. During testing, we employ greedy search to generate system summary sequences. For the task of summarization, the ground truth summary sequences are usually close to the source texts. This property can be leveraged in beam search. Tan et al., [46] describe a beam search with reference mechanism that rewards system summaries that have a high degree of bigram overlap with the source texts. We describe it in Eq. (2.11), where where $\mathscr{S}(w)$ denotes the score of word w. $\mathscr{B}(\mathbf{y}_{<t}, \mathbf{x})$ measures the number of bigrams shared by the system summary (up to time step t-1) and the source text; $\{\mathbf{y}_{<t}, w\}$ adds a word w to the end of the system summary. The shorter the source text (measured by length S), the more weight a shared bigram will add to the score of the current word w. A hyperparameter η controls the degree of closeness between the system summary and the source text.

$$\mathscr{S}(w) = \log P(w) + \eta \frac{\mathscr{B}(\{\mathbf{y}_{<t}, w\}, \mathbf{x}) - \mathscr{B}(\mathbf{y}_{<t}, \mathbf{x})}{S} \tag{2.11}$$

2.4 Experiments

We evaluate the proposed structure-infused copy mechanisms for summarization in this section. We describe the dataset, experimental settings, baselines, and finally, evaluation results and analysis.

2.4.1 Data Sets

We evaluate our proposed models on the Gigaword summarization dataset [77, 27]. The task is to reduce the first sentence of an article to a title-like summary. We obtain dependency parse trees for source sentences using the Stanford neural network parser [78]. We also use the standard train/valid/test data splits. Following [27], the train and valid splits are pruned[2] to improve the data quality. Spurious pairs that are repetitive, overly long/short, and pairs whose source and summary sequences have little word overlap are removed. No pruning is performed for instances in the test set. The processed corpus contains 4,018K training instances. We construct two (non-overlapped) validation sets: "valid-4096" contains 4,096 randomly sampled instances from the valid split; it is used for hyperparameter tuning and early stopping. "valid-2000" is used for evaluation; it allows

[2] `https://github.com/facebookarchive/NAMAS/blob/master/dataset/filter.py`

the models to be trained and evaluated on pruned instances. Finally, we report results on the standard Gigaword test set [27] containing 1,951 instances ("test-1951").

2.4.2 Experimental Setup

We use the Xavier scheme [79] for parameter initialization, where weights are initialized using a Gaussian distribution $\mathbf{W}_{i,j} \sim \mathcal{N}(0, \sigma)$, $\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$; n_{in} and n_{out} are numbers of the input and output units of the network; biases are set to be 0. We further implement two techniques to accelerate mini-batch training. First, all training instances are sorted by the source sequence length and partitioned into mini-batches. The shorter sequences are padded to have the same length as the longest sequence in the batch. All batches are shuffled at the beginning of each epoch. Second, we introduce a variable-length batch vocabulary containing only source words of the current mini-batch and words of the output vocabulary. $P(w)$ in Eq. (2.9) only needs to be calculated for words in the batch vocabulary. It is magnitudes smaller than a direct combination of the input and output vocabularies. Finally, our input vocabulary contains the most frequent 70K words in the source texts and summaries. The output vocabulary contains 5K words by default. More network parameters are presented in Table 2.3.

Table 2.3: Parameter settings of our summarization system.

Input vocabulary size	70K
Output vocabulary size	5K (default)
Dim. of word embeddings	100
Dim. of structural embeddings	16
Num. of encoder/decoder hidden units	256
Adam optimizer [80]	lr = 1e-4
Coeff. for coverage-based regularizer	$\lambda = 1$
Coeff. for beam search with reference	$\eta \approx 13.5$
Beam size	$K = 5$
Minibatch size	$M = 64$
Early stopping criterion (max 20 epochs)	valid. loss
Gradient clipping [?]	g $\in$ [-5, 5]

2.5 Results

ROUGE results on valid set We first report results on the Gigaword valid-2000 dataset in Table 2.4. We present R-1, R-2, and R-L scores [42] that respectively measures the overlapped unigrams, bigrams, and longest common subsequences between the system and reference summaries[3]. Our baseline system ("Baseline") implements the seq2seq architecture with the basic copy mechanism (Eq. (2.1-2.9)). It is a strong baseline that resembles the pointer-generator net-

[3]w/ ROUGE options: -n 2 -m -w 1.2 -c 95 -r 1000

Table 2.4: Results on the Gigaword valid-2000 set (full-length F1). Models implementing the structure-infused copy mechanisms ("Struct+*") outperform the baseline.

	Gigaword Valid-2000		
System	**R-1**	**R-2**	**R-L**
Baseline	42.48	21.34	40.18
Struct+Input	42.44	21.75	40.46
Struct+Hidden	42.88	21.81	40.63
Struct+2Way+Word	**43.21**	21.84	**40.86**
Struct+2Way+Relation	42.83	**21.85**	40.60

Table 2.5: Example system summaries. 'S:' source; 'T:' target; 'B:' baseline; 'I:' Struct+Input; 'H:' Struct+Hidden; 'W:' 2Way+Word; "R:" 2Way+Relation. "2Way+Relation" is able to preserve important source relations in the summary, e.g., "government $\xleftarrow{\text{nsubj}}$ files," "files $\xrightarrow{\text{dobj}}$ round," and "round $\xrightarrow{\text{nmod}}$ charges."

S:	the government filed another round of criminal charges in a widening stock options scandal
T:	options scandal widens
B:	government files more charges in stock options scandal
I:	another round of criminal charges in stock options scandal
H:	charges filed in stock options scandal
W:	another round of criminal charges in stock options scandal
R:	government files another round of criminal charges in options scandal

Table 2.6: Example system summaries. “Struct+Hidden” and “2Way+Relation” successfully preserve salient source words (“emergency food shipments”), which are missed out by other systems. We observe that copying “hold talks” from the source also makes the resulting summaries more informative than using the word “meet.”

S:	red cross negotiators from rivals north korea and south korea held talks wednesday on emergency food shipments to starving north koreans and agreed to meet again thursday
T:	koreas meet in beijing to discuss food aid from south eds
B:	north korea , south korea agree to meet again
I:	north korea , south korea meet again
H:	north korea , south korea meet on emergency food shipments
W:	north korea , south korea hold talks on food shipments
R:	north korea , south korea hold talks on emergency food shipments

works described in [29]. The structural models ("Struct+*") differ from the baseline only on the structure-infused copy mechanisms. All models are evaluated without the coverage regularizer or beam search (§2.3.3) to ensure fair comparison. Overall, we observe that models equipped with the structure-infused copy mechanisms are superior to the baseline, suggesting that combining source syntactic structure with the copy mechanism is effective. We found that the "Struct+Hidden" architecture, which directly concatenates structural embeddings with the encoder hidden states, outperforms "Struct+Input" despite that the latter requires more parameters. "Struct+2Way+Word" also demonstrates strong performance, achieving 43.21%, 21.84%, and 40.86% F_1 scores, for R-1, R-2, and R-L respectively.

ROUGE results on test set We compare our proposed approach with a range of state-of-the-art neural summarization systems. Results on the standard Gigaword test set ("test-1951") are presented in Table 2.7. Details about these systems are provided in Table 2.8. Overall, our proposed approach with structure-infused pointer networks perform strongly, yielding ROUGE scores that are on-par with or surpassing state-of-the-art published systems. Notice that the scores on the valid-2000 dataset are generally higher than those of test-1951. This is because the (source, summary) pairs in the Gigaword test set are not pruned (see §2.4.1). In some cases, none (or very few) of the summary words appear in the source. This may cause difficulties to the systems equipped with the copy mechanism. The "Struct+2Way+Word" architecture that respectively models the semantic and syntactic importance of source words achieves the highest scores. It outperforms its counterpart of "Struct+2Way+Relation," which seeks to preserve source dependency relations in summaries. We conjecture that the imperfect dependency parse trees generated by the parser

may affect the "Struct+2Way+Relation" results. However, because the Gigaword dataset does not provide gold-standard annotations for parse trees, we could not easily verify this and will leave it for future work. In Table 2.5 and 2.6, we present system summaries produced by various models.

Linguistic quality To further gauge the summary quality, we hire human workers from the Amazon Mechanical Turk platform to rate summaries on a Likert scale of 1 to 5 according to three criteria [85]: *fluency* (is the summary grammatical and well-formed?), *informativeness* (to what extent is the meaning of the original sentence preserved in the summary?), and *faithfulness* (is the summary accurate and faithful to the original?). We sample 100 instances from the test set and employ 5 turkers to rate each summary; their averaged scores are presented in Table 2.9. We found that "Struct+2Way+Relation" outperforms "Struct+Input" on all three criteria. It also compares favorably to ground-truth summaries on "fluency" and "faithfulness." On the other hand, the ground-truth summaries, corresponding to article titles, are judged as less satisfying according to human raters.

Dependency relations We investigate the source dependency relations preserved in the summaries in Table 2.10. A source relation is considered preserved if both its words appear in the summary. We observe that the models implementing structure-infused copy mechanisms (e.g., "Struct+2Way+Word") are more likely to preserve important dependency relations in the summaries, including nsubj, dobj, amod, nmod, and nmod:poss. Dependency relations that are less important (mark, case, conj, cc, det) are less likely to be preserved. These results show that our

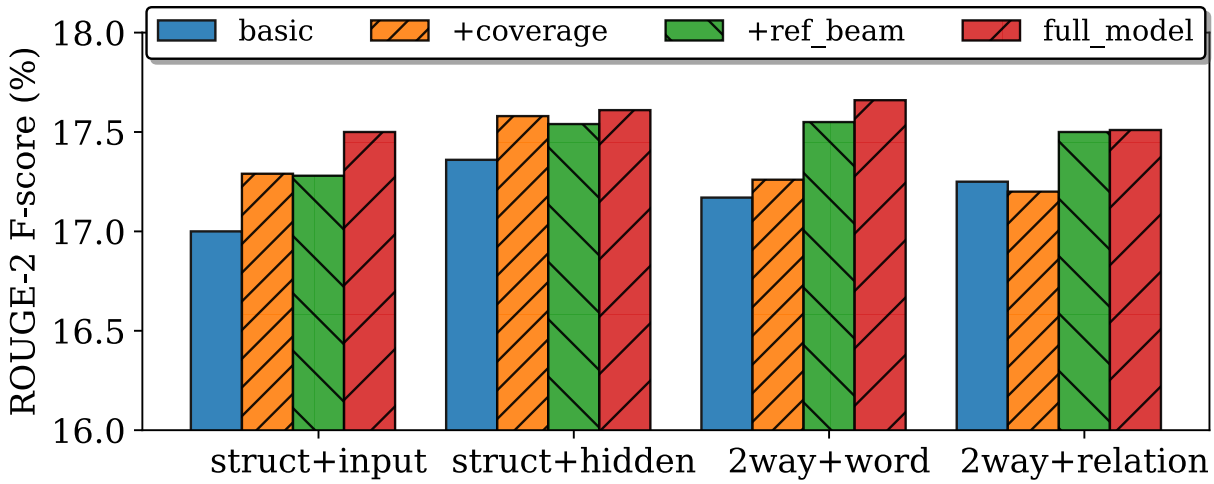

Figure 2.3: Effects of applying the coverage regularizer and the reference beam search to structural models, evaluated on test-1951. Combining both yields the highest scores.

structure-infused copy mechanisms can learn to recognize the importance of dependency relations and selectively preserve them in the summaries.

Coverage and reference beam In Figure 2.3, we investigate the effect of applying the coverage regularizer ("coverage") and reference-based beam search ("ref_beam") (§2.3.3) to our models. The coverage regularizer is applied in a second training stage, where the system is trained for an extra 5 epochs with coverage and the model yielding the lowest validation loss is selected. Both coverage and ref_beam can improve the system performance. Our observation suggests that ref_beam is an effective addition to shorten the gap between different systems.

Output vocabulary size Finally, we investigate the impact of the output vocabulary size on the summarization performance in Table 2.11. All our models by default use an output vocabulary of

5K words in order to make the results comparable to state-of-the-art-systems. However, we observe that there is a potential to further boost the system performance (17.25→17.62 R-2 F1-score, w/o coverage or ref_beam) if we had chosen to use a larger vocabulary (10K) and can endure a slightly longer training time (1.2x). In Table 2.11, we further report the percentages of reference summary words covered by the output vocabulary ("InVcb") and covered by either the output vocabulary or the source text ("InVcb+Src"). The gap between the two conditions shortens as the size of the output vocabulary is increased.

2.6 Conclusion

In this paper, we investigated structure-infused copy mechanisms that combine source syntactic structure with the copy mechanism of an abstractive summarization system. We compared various system architectures and showed that our models can effectively preserve salient source relations in summaries. Results on benchmark datasets showed that the structural models are on-par with or surpass state-of-the-art published systems.

Table 2.7: Results on the Gigaword test-1951 set (full-length F1). Models with structure-infused copy mechanisms ("Struct+*") perform well. Their R-2 F-scores are on-par with or outperform state-of-the-art published systems.

	Gigaword Test-1951		
System	**R-1**	**R-2**	**R-L**
ABS [?]	29.55	11.32	26.42
ABS+ [?]	29.76	11.88	26.96
Luong-NMT [81]	33.10	14.45	30.71
RAS-LSTM [81]	32.55	14.70	30.03
RAS-Elman [81]	33.78	15.97	31.15
ASC+FSC1 [82]	34.17	15.94	31.92
lvt2k-1sent [?]	32.67	15.59	30.64
lvt5k-1sent [?]	35.30	16.64	32.62
Multi-Task [83]	32.75	15.35	30.82
DRGD [84]	36.27	17.57	33.62
Baseline (this paper)	35.43	17.49	33.39
Struct+Input (this paper)	35.32	17.50	33.25
Struct+2Way+Relation (this paper)	35.46	17.51	33.28
Struct+Hidden (this paper)	**35.49**	17.61	33.33
Struct+2Way+Word (this paper)	35.47	**17.66**	**33.52**

Table 2.8: Existing summarization methods.

- **ABS** and **ABS+** [27] are the first work introducing an encoder-decoder architecture for summarization.
- **Luong-NMT** [81] is a re-implementation of the attentive stacked LSTM encoder-decoder of Luong et al. [**?**].
- **RAS-LSTM** and **RAS-Elman** [81] describe a convolutional attentive encoder that ensures the decoder focuses on appropriate words at each step of generation.
- **ASC+FSC1** [82] presents a generative auto-encoding sentence compression model jointly trained on labelled/unlabelled data.
- **lvt2k-1sent** and **lvt5k-1sent** [28] address issues in the attentive encoder-decoder framework, including modeling keywords, capturing sentence-to-word structure, and handling rare words.
- **Multi-Task w/ Entailment** [83] combines entailment with summarization in a multi-task setting.
- **DRGD** [84] describes a deep recurrent generative decoder learning latent structure of summary sequences via variational inference.

Table 2.9: Informativeness, fluency, and faithfulness scores of summaries. They are rated by Amazon turkers on a Likert scale of 1 (worst) to 5 (best). We choose to evaluate Struct+2Way+Relation (as oppose to 2Way+Word) because it focuses on preserving source relations in the summaries.

System	**Info.**	**Fluency**	**Faithful.**
Struct+Input	2.9	3.3	3.0
Struct+2Way+Relation	3.0	3.4	3.1
Ground-truth Summ.	3.2	3.5	3.1

Table 2.10: Percentages of source dependency relations (of various types) preserved in the system summaries.

System	**nsubj**	**dobj**	**amod**	**nmod**	**nmod:poss**	**mark**	**case**	**conj**	**cc**	**det**
Baseline	7.23	12.07	20.45	8.73	12.46	15.83	14.84	9.72	5.03	2.22
Struct+Input	7.03	11.72	19.72	**9.17**↑	12.46	15.35	14.69	9.55	4.67	1.97
Struct+Hidden	**7.78**↑	**12.34**↑	**21.11**↑	**9.18**↑	**14.86**↑	14.93	**15.84**↑	9.47	3.93	**2.65**↑
Struct+2Way+Word	**7.46**↑	**12.69**↑	**20.59**↑	**9.03**↑	**13.00**↑	15.83	14.43	8.86	3.48	1.91
Struct+2Way+Relation	**7.35**↑	**12.07**↑	**20.59**↑	8.68	**13.47**↑	15.41	14.39	9.12	4.30	1.89

Table 2.11: Results of the "Struct+2Way+Relation" system trained using output vocabularies of various sizes ($|V|$), evaluated on test-1951 w/o coverage or ref_beam. The training speed is calculated as the elapsed time (hours) per epoch, tested on a GTX 1080Ti GPU card.

$\|V\|$	**R-2**	**Train Speed**	**InVcb**	**InVcb+Src**
1K	13.99	2.5h/epoch	60.57	76.04
2K	15.35	2.7h/epoch	69.71	80.72
5K	17.25	3.2h/epoch	79.98	86.51
10K	17.62	3.8h/epoch	88.26	92.18

CHAPTER 3
INCORPORATE TARGET STRUCTURAL INFORMATION: JOINT PARSING AND GENERATION FOR ABSTRACTIVE SUMMARIZATION

In the second Chapter, I will demonstrate another paper accepted at AAAI 2020. Instead of emphasizing those important content from the input, we try to simultaneously decode the summary and its parsing tree by employing an additional tree-decoder. From the this research, we found the model can better preserve the dependency relations from both original text and reference summary. Besides, the model generates better grammatical summaries under human judgment.

3.1 Background

It is crucial for a summary to not only condense the source text but also render itself grammatical. Without grammatical sentences, a summary can be ineffective, because human brain derives meaning from the sentence as a whole rather than individual words. Abstractive summarization has made considerable recent progress [29, 86, 87]. Nonetheless, studies suggest that system summaries remain imperfect. A summary sentence can be ungrammatical and fail to convey the intended meaning, despite its local fluency [88, 89]. In Table 3.1, we show example abstractive summaries produced by neural abstractive summarizers. The first summary has failed to con-

Table 3.1: Example summaries generated by neural abstractive summarizers. They are manually re-cased for readability.

Source	Today, because of a CNN story and the generosity of donors from around the world, Kekula wears scrubs bearing the emblem of the Emory University ...
Summ.	CNN story and generosity of donors from around the world, Kekula wears scrubs ...
Source	In its propaganda, ISIS has been using Abu Ghraib and other cases of Western abuse to legitimize its current actions in Iraq as the latest episodes ...
Summ.	In its propaganda, ISIS is being used by the Islamic State in Iraq and Syria ...
Source	Both state and foreign investments in Vietnam's agriculture have been not sufficient enough, while local farmers have to pay fees to contribute to building rural roads ...
Summ.	Vietnam's agriculture not sufficient enough

form to grammar and other summaries changed the original meanings. These summaries not only mislead the reader but also hinder the applicability of summarization techniques in real-world scenarios.

In this paper, we attempt to remedy this problem by introducing a new architecture to jointly generate a summary sentence and its syntactic parse, while performing abstraction. This is a non-trivial task, as the method must tightly couple summarization and parsing algorithms, which are two significant branches of NLP. A joint model for generating summary sentences and parse trees can be more appealing than a pipeline method. The latter may suffer from error propagation, e.g., an ill-formed summary sentence can lead to more parsing errors. Further, a joint method mimics

the human behavior, e.g., an editor writes a summary and makes corrections instantly as the text is written. She needs not to finish the whole summary in order to correct errors. A method that incrementally produces a summary sentence and its syntactic parse aligns with this observation.

Our proposed joint model seeks to transform the *source* sequence to a linearized parse tree of the *summary* sequence. The model seamlessly integrates a shift-reduce dependency parser into a summarization system employing the encoder-decoder architecture. A "SHIFT" operation leads the summarizer to generate a new word by copying it from the source text or choosing a word from the vocabulary; whereas a "REDUCE" operation adds a dependency arc between words of the partial summary. The challenge of this task is to construct effective representations that support both tasks, as they require different contextual representations. We propose to couple a sequential decoder for predicting new summary words and a tree-based decoder for predicting dependency arcs, and ensure both decoders work in a synchronized fashion. We also introduce an important addition making use of *topological sorting* of tree nodes to accelerate the training procedure, making the framework computationally feasible. Our research contributions can be summarized as follows:

- we propose to simultaneously decode sentences and their syntactic parses while performing abstraction. Our work represents a first attempt toward joint abstractive summarization and parsing that holds promise for improved sentence grammaticality and truthful summaries;
- we present a novel neural architecture coupling a sequential and a tree decoder to generate summary sentences and parse trees simultaneously. Experiments are performed on a variety of summarization datasets to demonstrate the effectiveness of the proposed method;

- we describe a new human evaluation protocol to assess if an abstractive summary has preserved the original meanings, and importantly, if it has introduced any new meanings that are nonexistent in the original text. The last factor is largely under-investigated in the literature.[1]

3.2 Related Work

Recent years have seen increasing interest in summarization using encoder-decoder models [27, 28, 29, 90, 91]. An encoder condenses the source text to a fix-length vector and a decoder unrolls it to a summary. An encoder (or decoder) can be realized using recurrent networks [59, 46, 92, 93, 94], convolutional networks [81, 95], or Transformer [30, 96, 97]. To generate a summary word, a decoder can copy a word from the source text or select an unseen word from the vocabulary. This flexibility allows for diverse lexical choices. Nevertheless, with greater flexibility comes the increased risk of producing ill-formed summary sentences that are ungrammatical and fail to preserve the original meanings.

Parsing the source text to identify summary-worthy textual units has been exploited in the past. Marcu [98, 99] utilizes discourse structure generated by an RST parser to identify summary units that are central to the claims of the document. A number of recent studies have explored constituency and dependency grammars [47, 52, 100, 101, 48, 56, 50], rhetorical structure [102, 103, 104], and abstract meaning representation [64, 105, 106] to generate compressive and abstractive summaries. In this paper we emphasize that *target-side* syntactic analysis is es-

[1] We make our implementation and models publicly available at https://github.com/ucfnlp/joint-parse-n-summarize

pecially important to ensure the well-formedness of abstractive summaries, because generating summary words and predicting relations between words are interleaved operations.

Summarization and parsing are traditionally regarded as separate tasks. These systems are now both realized using neural sequence-to-sequence models, making it possible to tackle both tasks in a single framework. There have been a variety of studies examining neural dependency parsers using transition- and graph-based algorithms [107, 108, 109, 110]. Our method, inspired by the recurrent neural network grammar (RNNG)[111] that describes a generative probabilistic model for parsing and language modeling [112], offers a way to perform summary generation and parsing in a synchronized manner. Incorporating syntax is found to improve translation [68, 113, 114, 115]. But to date, there has been little work to simultaneously generate a sentence and its syntactic parse, combining summarization with parsing techniques. Our aim is not to improve existing parsers but to leveraging parsing for abstractive summarization. Parsing is essentially a structured prediction problem, whereas summarization involves *information reduction* from source to target, which poses an important challenge. In the following section, we describe our model in detail.

3.3 Our Approach

Our goal is to transform a source text $\mathbf{x}$ containing one or more sentences to a target sequence containing a linearized parse tree of the summary, represented by $\mathbf{y}^{\mathscr{T}}$. We expect a summary to contain a single sentence, as our focus is to improve sentence grammaticality.[2] We use dependency

[2]When a multi-sentence summary is desired, it is possible to generate summary sentences repeatedly from selected subsets of source sentences, as suggested by recent studies [86, 94].

grammar as syntactic representation of the summary. Dependency is useful for semantic tasks and transition-based parsing algorithms are efficient, linear-time in the sequence length.[3]

Problem formulation Our target sequence $\mathbf{y}^{\mathscr{T}}$ consists of interleaved GEN(w) and REDUCE-L/R operations that incrementally build a dependency parse tree. Table 3.2 shows an example. The second column contains $\mathbf{y}^{\mathscr{T}}$ and the third column contains partial dependency trees stored in a *stack*. A GEN(w) operation pushes a summary word w to the stack; REDUCE-L creates a left arc between the top and second top word in the stack, where the top word is the head; REDUCE-R creates a right arc where the top word is the dependent. We choose not to label the arcs, as this work focuses on generating well-structured sentences but not on predicting labels. The decoding process comes to an end when there is a single tree remaining in the stack. A summary $\mathbf{y}$ can be obtained from $\mathbf{y}^{\mathscr{T}}$ by retrieving all GEN operations.

We aim to predict the target sequence $\mathbf{y}^{\mathscr{T}}$ conditioned on the source $\mathbf{x}$. The process proceeds incrementally. As illustrated in Eq. (3.1), $P(\mathbf{y}^{\mathscr{T}}|\mathbf{x})$ is factorized over time steps. $P(\mathbf{y}_t^{\mathscr{T}} = o|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})$ denotes the probability of a parsing operation, where $o \in \{\text{REDUCE-L}, \text{REDUCE-R}, \text{GEN}\}$ and GEN is unlexicalized. $P(\mathbf{y}_t^{\mathscr{T}} = w|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})$ represents the probability of generating a summary word w at the t-th step; the word can either be copied from the source text or selected from the vocabulary.

$$P(\mathbf{y}^{\mathscr{T}}|\mathbf{x}) = \prod_t \Big[\underbrace{P(\mathbf{y}_t^{\mathscr{T}} = o|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})}_{\text{parsing}} \tag{3.1}$$

$$\times \underbrace{P(\mathbf{y}_t^{\mathscr{T}} = w|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})^{\mathbb{1}[o=\text{GEN}]}}_{\text{summarization}} \Big] \tag{3.2}$$

[3] Our method is also general enough to allow other syntactic/semantic formalisms such as the constituency grammar or abstract meaning representation [116, 117] to be exploited in future work.

Table 3.2: Illustration of the decoding process. A summary sentence "*a man escaped from prison*" and its dependency structure are simultaneously generated. The second column shows the target sequence $\mathbf{y}^{\mathscr{T}}$ and the third column contains partial parse trees stored in a *stack*.

t	$\mathbf{y}^{\mathscr{T}}$	**Stack**
1	–	R (root node)
2	GEN(*a*)	R *a*
3	GEN(*man*)	R *a* *man*
4	REDUCE-L	R *a*←*man*
5	GEN(*escaped*)	R *a*←*man* *escaped*
6	REDUCE-L	R *a*←*man*←*escaped*
7	GEN(*from*)	R *a*←*man*←*escaped* *from*
8	GEN(*prison*)	R *a*←*man*←*escaped* *from* *prison*
9	REDUCE-L	R *a*←*man*←*escaped* *from*←*prison*
10	REDUCE-R	R *a*←*man*←*escaped* *from*⇄*prison*
11	REDUCE-R	R *a*←*man*⇄*escaped* *from*⇄*prison*

At training time, the ground-truth sequence $\hat{\mathbf{y}}^{\mathscr{T}}$ is available, $P(\hat{\mathbf{y}}_t^{\mathscr{T}} = w|\hat{\mathbf{y}}_{<t}^{\mathscr{T}}, \mathbf{x})$ needs only be computed for certain steps where the parsing operation is GEN, as indicated by $\mathbb{1}[o = \text{GEN}]$. Our loss term corresponds to the conditional log-likelihood which can be separately calculated for parsing and summarization operations (Eq. (3.3)). During inference, we calculate $P(\mathbf{y}_t^{\mathscr{T}}|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})$ as a joint distribution over parsing and summarization operations, where $\mathbf{y}_t^{\mathscr{T}} \in \{\text{REDUCE-L}, \text{REDUCE-R}, \text{GEN}(w)\}$.

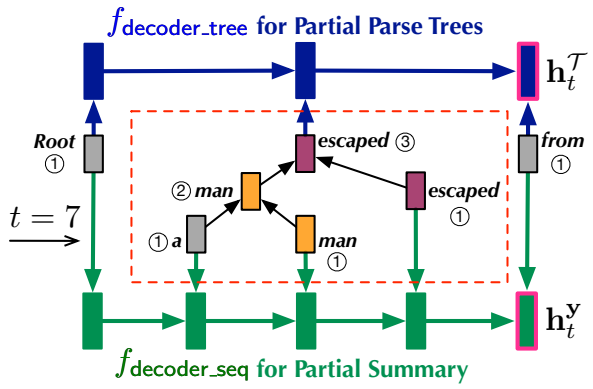

Figure 3.1: $f_{\text{decoder\_tree}}$ (top) consumes the *partial tree* representations of time t one by one to build hidden representation $\mathbf{h}_t^{\mathcal{T}}$; $f_{\text{decoder\_seq}}$ (bottom) consumes the embeddings of summary words to build *partial summary* representation $\mathbf{h}_t^{\mathbf{y}}$.

$$\log P(\hat{\mathbf{y}}^{\mathcal{T}}|\mathbf{x}) = \Big[\sum_t \log P(\hat{\mathbf{y}}_t^{\mathcal{T}} = o|\hat{\mathbf{y}}_{<t}^{\mathcal{T}}, \mathbf{x})\Big] \tag{3.3}$$

$$+\Big[\sum_{t:o_t=\text{GEN}} \log P(\hat{\mathbf{y}}_t^{\mathcal{T}} = w|\hat{\mathbf{y}}_{<t}^{\mathcal{T}}, \mathbf{x})\Big] \tag{3.4}$$

Neural representations A crucial next step is to build neural representations to support both tasks. Predicting the next parsing operation requires us to build an effective representation for *partial parse trees*, denoted by $\mathbf{h}_t^{\mathcal{T}}$ at the t-th step, whereas predicting the next summary word suggests an effective representation for the *partial summary*, represented by $\mathbf{h}_t^{\mathbf{y}}$. We envision both tasks to benefit from a context vector $\mathbf{c}_t^{\mathbf{x}}$ that encodes source content that is deemed important for the t-th decoding step. We next describe a new architecture building representations for $\mathbf{h}_t^{\mathcal{T}}$, $\mathbf{h}_t^{\mathbf{y}}$, and $\mathbf{c}_t^{\mathbf{x}}$.

We model partial trees using stack-LSTM [107, 111]. Our stack maintains a set of partial trees at any time t; they are shown in the t-th row of Table 3.2. For each partial tree, we build a vector representation for it by recursively applying a syntactic composition function (Eq. (3.5)). The representation is built from bottom up, shown in the dotted circle of Figure 3.1. A left arc (REDUCE-L) pops two elements from the stack. It then applies the composition function to create a new representation $\mathbf{g}_{\mathsf{new\_head}}$ and push it onto the stack; similarly for right arc (REDUCE-R). A GEN(w) operation pushes the embedding of a summary word $\mathbf{e}(w)$ to the stack.[4] Kuncoro et al. [112] report that the composition function learns to compute a tree representation by preserving the semantics of the head word, which fits our task.

$$\mathbf{g}_{\mathsf{new\_head}} = \tanh(\mathbf{W}^g[\mathbf{g}_{\mathsf{head}}||\mathbf{g}_{\mathsf{dependent}}] + \mathbf{b}^g) \tag{3.5}$$

We introduce an LSTM, denoted by $f_{\mathsf{decoder\_tree}}$, to consume the *partial tree* representations of time t one by one to build the hidden representation $\mathbf{h}_t^{\mathscr{T}}$. An illustration is presented in Figure 3.1. E.g., when t=7, the stack contains 3 partial trees and we build a vector representation for each. $f_{\mathsf{decoder\_tree}}$ is unrolled 3 steps and its last hidden state is $\mathbf{h}_t^{\mathscr{T}}$. Similarly, we build the *partial summary* representation $\mathbf{h}_t^{\mathbf{y}}$ using an LSTM denoted by $f_{\mathsf{decoder\_seq}}$, which consumes the embeddings of summary words. For example, when t=7, there are 5 words in the partial summary. $f_{\mathsf{decoder\_seq}}$ is unrolled 5 steps and its last hidden state is used as $\mathbf{h}_t^{\mathbf{y}}$. Note that for some steps, e.g., t=9, no summary words are generated, we copy $\mathbf{h}_t^{\mathbf{y}}$ from its previous step $\mathbf{h}_{t-1}^{\mathbf{y}}$.

[4] $\mathbf{e}(w)$ has the same size as partial tree representations $\mathbf{g}$.

A context vector ($\mathbf{c}_t^{\mathbf{x}}$) encoding the source content that is deemed important for the t-th decoding step is crucial to our method. Important source content can not only aid in the prediction of future summary words but also parsing operations. We build the context vector $\mathbf{c}_t^{\mathbf{x}}$ in two steps. First, we encode the source text $\mathbf{x}$ using a two-layer bidirectional LSTM denoted by f_{encoder}. We use $\{\mathbf{h}_i^{\mathbf{x}}\}$ to denote the encoder hidden states, where i is the index of source words. Next, we characterize the interaction between encoder and decoder hidden states using an attention mechanism (Eq. (3.6)). We concatenate the partial tree and partial summary representations $[\mathbf{h}_t^{\mathscr{T}}||\mathbf{h}_t^{\mathbf{y}}]$ to form the decoder state. The score $S_{t,i}$ measures the importance of the i-th source word to the t-th decoding step. A context vector $\mathbf{c}_t^{\mathbf{x}}$ is then constructed as the weighted sum of source representations (Eq. (3.7)).

$$S_{t,i} = \mathbf{w}^\top \tanh(\mathbf{W}^d[\mathbf{h}_t^{\mathscr{T}}||\mathbf{h}_t^{\mathbf{y}}] + \mathbf{W}^e\mathbf{h}_i^{\mathbf{x}}) \tag{3.6}$$

$$\mathbf{c}_t^{\mathbf{x}} = \text{softmax}(\mathbf{S}_t)\mathbf{h}^{\mathbf{x}} \tag{3.7}$$

Prediction We predict summary words $P(\mathbf{y}_t^{\mathscr{T}} = w|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})$ and parsing operations $P(\mathbf{y}_t^{\mathscr{T}} = o|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x})$ with these representations. We expect historical parsing operations to be helpful for the latter task, i.e., the sequence of $\{$REDUCE-L, REDUCE-R, GEN(w)$\}$ operations shown in Table 3.2. We thus use an LSTM to encode the sequence of past operations and its last hidden state is denoted by $\mathbf{h}_t^{\mathscr{O}}$. A parsing operation is predicted based on $[\mathbf{h}_t^{\mathscr{T}}||\mathbf{h}_t^{\mathscr{O}}||\mathbf{c}_t^{\mathbf{x}}]$, and we apply the softmax to obtain a distribution over parsing operations (Eq. (3.9)).

$$\widetilde{\mathbf{h}}_t^{\mathscr{T}} = \tanh(\mathbf{W}^a[\mathbf{h}_t^{\mathscr{T}}||\mathbf{h}_t^{\mathscr{O}}||\mathbf{c}_t^{\mathbf{x}}] + \mathbf{b}^a) \tag{3.8}$$

$$P(\mathbf{y}_t^{\mathscr{T}} = o|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x}) = \text{softmax}(\mathbf{W}^o\widetilde{\mathbf{h}}_t^{\mathscr{T}}) \tag{3.9}$$

A summarizer should allow a summary word to be copied from the source text or generated from the vocabulary. We implement a soft switch following See et al. [29], where $\lambda = \sigma(\mathbf{W}^z[\mathbf{h}_t^{\mathbf{y}}||\mathbf{h}_t^{\mathscr{T}}||\mathbf{c}_t^{\mathbf{x}}]) + b^z)$ is the likelihood of generating a summary word from the vocabulary. The generation probability is defined in Eqs. (3.10-3.11). If a word w occurs once or more times in the source text, its copy probability ($\sum_{i:w_i=w} \alpha_{t,i}$) is the sum of its attention scores over all the occurrences, where $\alpha_{t,i}$=softmax$_i(\mathbf{S}_t)$. If a word w appears in both the vocabulary and source text, $P(\mathbf{y}_t^{\mathscr{T}} = w|\cdot)$ is a weighted sum of the generation and copy probabilities.

$$\widetilde{\mathbf{h}}_t^{\mathbf{y}} = \tanh(\mathbf{W}^c[\mathbf{h}_t^{\mathbf{y}}||\mathbf{h}_t^{\mathscr{T}}||\mathbf{c}_t^{\mathbf{x}}] + \mathbf{b}^c) \tag{3.10}$$

$$\widetilde{P}(\mathbf{y}_t^{\mathscr{T}} = w|\mathbf{y}_{<t}^{\mathscr{T}}, \mathbf{x}) \ = \text{softmax}(\mathbf{W}^w\widetilde{\mathbf{h}}_t^{\mathbf{y}}) \tag{3.11}$$

$$P(\mathbf{y}_t^{\mathscr{T}} = w|\cdot) = \lambda\widetilde{P}(\mathbf{y}_t^{\mathscr{T}} = w|\cdot) + (1-\lambda)\sum_{i:w_i=w} \alpha_{t,i} \tag{3.12}$$

Acceleration Obtaining partial tree representations ($\mathbf{h}_t^{\mathscr{T}}$) can be computationally expensive, because $\mathbf{h}_t^{\mathscr{T}}$ has to be computed bottom-up according to the topology of a parse tree. Further, parse trees in a mini-batch exhibit distinct topology, making it difficult to execute parallely; frameworks such as DyNet [118] often process one instance at a time. In this work we instead propose to arrange the tree nodes of all instances into groups according to their topological order; representations for nodes of the same group ($\mathbf{h}_t^{\mathscr{T}}$) are computed in parallel. For example, in Figure 3.1, the nodes marked with "1" are first processed, followed by nodes marked with "2" and so forth. This strategy allows for mini-batch training with parse trees of distinct topology and maximizing the usage of computing resources.

3.4 Experiments

We present our datasets, settings, baselines, qualitative and quantitative evaluation of our proposed method. We then discuss our findings and shed light on future work.

3.4.1 Data and Hyperparameters

We conduct experiments on a variety of datasets to gauge the effectiveness of our proposed method. We experiment with GIGAWORD [77] and NEWSROOM [119]. GIGAWORD contains about 10M articles gathered from seven news sources (1995-2010); NEWSROOM is a more recent effort containing 1.3M articles (1998-2017) collected from 38 news agencies. We use the standard data splits and follow the same procedure as Rush et al. [27] to process both datasets. The task of GIGAWORD and NEWSROOM is to reduce the first sentence of a news article to a title-like summary.

The CNN/DM dataset [120] has been extensively studied. We use the version provided by See et al. [29] but formulate it as a sentence summarization task. We aim to condense a source sentence to a well-formed summary sentence. The source sentences are obtained by pairing each summary sentence with its most similar sentence in the article according to averaged R-1, R-2, and R-L F-scores [42]. We denote this *reduced* dataset as "CNN/DM-R." It is distinct from GIGAWORD and NEWSROOM because its ground-truth summaries are full grammatical sentences, whereas the latter are article titles that appear enticing but not necessarily be full sentences.

Table 3.3: Statistics of our datasets. $|\mathbf{y}|$ is number of words.

	$\lvert\mathbf{y}\rvert$	**Train**	**Dev**	**Test**
GIGAWORD	8.41	4,020,581	4,096	1,951
NEWSROOM	10.18	199,341	21,530	21,382
CNN/DM-R	13.89	472,872	25,326	20,122
WEBMERGE	31.43	1,331,515	40,879	43,934

We further experiment on many-to-one sentence summarization, where the goal is to fuse multiple source sentences to a summary sentence. Existing datasets for sentence fusion are often small, containing thousands of instances [121]. In this work we present a novel use of a newly released dataset—WebSplit [122]. The dataset was originally developed for sentence *simplification*, where a lengthy source sentence is to be converted to multiple, simpler sentences for ease of understanding. Importantly, we swap the source and target sequences, so that the task becomes fusing multiple source sentences to a well-formed summary sentence. We name this task WEBMERGE to avoid confusion. On average, a source text contains 4.4 sentences and the target is a single sentence. A (source, target) pair is accompanied by a set of semantic triples in the form of "subject|property|object" and the semantics remain unchanged during merging. We utilize these triples for human evaluation (§3.4.2). In Table 3.3, we provide statistics of all datasets used in this study.

Hyperparameters We create an input vocabulary to contains word appearing 5 times or more in the dataset; the output vocabulary contains the most frequent 10k words. We set all LSTM hidden states to be 256 dimensions. Because datasets containing both summaries and human-annotated dependency parses are unavailable, we use the Stanford parser [78] to obtain parse trees for reference summaries. During training, we use a batch size of 64 and Adam [80] for parameter optimization, with lr=1e-3, betas=[0.9,0.999], and eps=1e-8. We apply gradient clipping of [-5,5], and a weight decay of 1e-6. At decoding time, we apply beam search with reference [46] to generate summary sequences. K=10 is the beam size.

3.4.2 Experimental Results

Summarization We present summarization results on all datasets. Evaluation is performed using the automatic metric of ROUGE [42], which measures the n-gram overlap between system and reference summaries, as well as human evaluation of grammaticality and preservation of meanings. We discuss our findings at the end.

In Table 3.4, we present summarization results on the Gigaword test set containing 1951 instances. We are able to compare our system, denoted by *GenParse*, with a variety of state-of-the-art neural abstractive summarizers; they are described below. Our system can be a valuable addition to existing neural summarizers, as it performs summarization and parsing jointly on the target-side to improve sentence grammaticality. We explore two variants of our system: GenParse-FULL represents the full model; GenParse-BASE is an ablated model where we drop the tree-decoder to test its

impact on summarization performance; this corresponds to removing $\mathbf{h}_t^{\mathscr{T}}$ and $\mathbf{h}_t^{\mathscr{O}}$ in all equations. All other components remain the same. As shown in Table 3.4, our GenParse system performs on par with or superior to state-of-the-art systems on the standard Gigaword test set. The full model yields the highest R-2 score of 18.85. It outperforms the GenParse-BASE model, demonstrating the effectiveness of coupling a sequential decoder with a tree-based decoder in a synchronized manner.

- *ABS* and *ABS+* [27] are the first work using an encoder-decoder architecture for summarization;
- *Luong-NMT* [81] re-implements the attentive encoder-decoder of Luong et al. [**?**];
- *RAS-LSTM* and *RAS-Elman* [81] describe a convolutional attentive encoder that ensures the decoder focuses on appropriate words at each step of generation;
- *ASC+FSC1* [82] presents a generative auto-encoding sentence compression model jointly trained on labelled/unlabelled data;
- *lvt2k-1sent* and *lvt5k-1sent* [28] address issues in the encoder-decoder model, including modeling keywords, capturing sentence-to-word structure, and handling rare words;
- *Multi-Task w/ Entailment* [123] combines entailment with summarization in a multi-task setting;
- *DRGD* [84] describes a deep recurrent generative decoder learning latent structure of summary sequences via variational inference;
- *Struct+2Way+Word* [88] describes a structure infused copy mechanism for sentence summarization;
- *EntailGen+QuesGen* [124] is a multi-task architecture to perform summarization with question generation and entailment generation in one framework.

In Table 3.5 we present summarization results on the NEWSROOM, CNN/DM-R, and WEBMERGE datasets. The task of WEBMERGE is to fuse multiple source sentences to a well-formed summary sentence while keeping the semantics unchanged; the task of NEWSROOM and CNN/DM-R is sentence summarization, but not document summarization. Because of that, the ROUGE scores presented in Table 3.5 should not be directly compared with other published results. Instead, we train the pointer-generator networks with coverage mechanism (PointerGen)[29], one of the best performed neural abstractive summarizers, on the train split of each dataset, then report results on the test split; we apply a similar process to our GenParse systems. We observe that the GenParse-FULL model consistently outperforms strong baselines across all datasets. The results are outstanding because our system jointly performs summarization and dependency parsing; it involves an increased task complexity than performing summarization only; and our full model is able to excel on this task.

Dependency parsing We expect dependency relations of a summary to be the same or similar to those of the source text or reference summary in order to preserve the original meanings. Generating a summary word means certain dependency relations are simultaneously added to the summary. For example, in Table 3.2, generating the word *escaped* leads a dependency relation *man*$\leftarrow$*escaped* to be included in the summary. In this section we demonstrate that by learning to jointly summarize and parse, our system can effectively improve the preservation of dependency relations.[5]

[5]We cannot compute parsing accuracy, because system and reference summaries use different words and their dependency structures are not directly comparable.

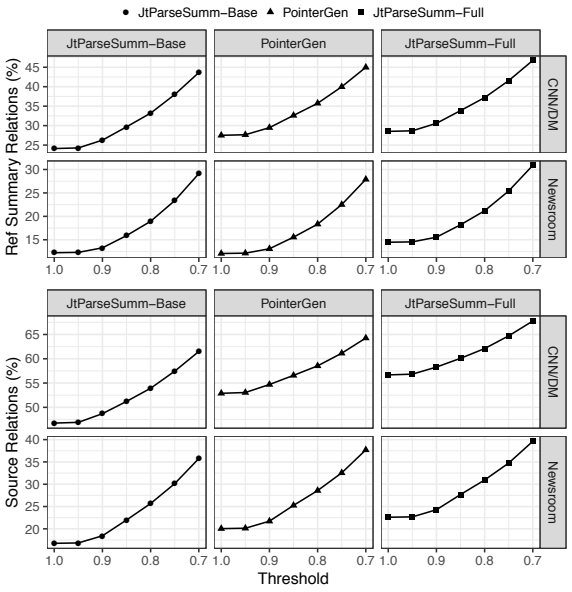

Figure 3.2: F-scores of systems on preserving relations of reference summaries (*top*) and source texts (*bottom*). We vary the threshold from 1.0 (strict match) to 0.7 in the x-axis to allow for strict and lenient matching of dependency relations.

In Figure 3.2 we demonstrate to what extent system summaries preserve relations of source texts and reference summaries. We contrast our system GenParse-BASE and GenParse-FULL that jointly performs summarization and parsing, against the strong baseline of PointerGen that first generates abstractive summaries then parses them using the off-the-shelf Stanford parser [78]. Dependency relations of source texts and reference summaries are also obtained using the Stanford parser. We calculate F-scores on preserving reference summary relations (top) and source relations (bottom) and on CNN/DM-R and NEWSROOM dataset, respectively. As shown in Figure 3.2, GenParse-FULL consistently outperforms other systems on preserving source and reference summary relations.

Abstractive summaries can contain paraphrases of source descriptions and we thus compare relations using both strict and lenient measures. A strict measure requires exact match of words. E.g., two relations $w_{1A} \leftarrow w_{1B}$ and $w_{2A} \leftarrow w_{2B}$ are equal if w_{1A} is the same as w_{2A}, and w_{1B} is the same as w_{2B}. A lenient measure computes Sim(w_{1A},w_{2A}) and Sim(w_{1B},w_{2B}) and it requires both scores to be greater than a threshold σ. We vary the threshold value along the x-axis to produce the plots in Figure 3.2. We define Sim($\cdot$,$\cdot$) as the cosine similarity of word embeddings; and a value of 1.0 corresponds to strict match. Overall, we notice that the GenParse-FULL method performs exceptionally well on retaining relations on the CNN/DM-R dataset. It achieves an F-score of 56.7% (σ=1) / 67.8% (σ=0.7) for source relations, and 28.5% (σ=1) / 46.8% (σ=0.7) for reference summary relations. This finding suggests that the proposed joint summarization and parsing method performs the best on summaries that contain full grammatical sentences, as is the case with CNN/DM-R, and this matches our expectation.

Human evaluation We proceed by introducing a novel human evaluation protocol assessing system summaries for grammaticality and preservation of original meanings. A quantitative measure is important because it allows us to compare different systems regarding to what extent their abstractive summaries preserve the original meanings and whether the summaries contain any falsified content that are nonexistent in the original texts. The latter is particularly under-investigated in the past. Our evaluation is made possible by utilizing RDF triples provided in WEBMERGE.

Table 3.6 illustrates the evaluation process. We present a summary to a group of human judges. They are instructed to rank this summary among four peers for grammaticality. Next, we require

the judges to answer a set of binary questions on (Q2) if the summary has conveyed the meaning of a given RDF triple, and (Q3) if the summary has conveyed any additional meanings that are not in the collection of triples. In particular, an RDF (Resource Description Format) triple is of the form *subject* | *property* | *object* and it is used for meaning representation [122]. The number of triples per instance varies from 1 to 7. A successful summary should preserve the meanings of all RDF triples and it shall not introduce any additional meanings. As an example, the summary A in Table 3.6 has introduced undesired content during abstraction (*died on July*), it thus makes factual errors that can mislead the reader.

Peer summaries are generated by GenParse-BASE, PointerGen, and GenParse-FULL. We further include human summaries for comparison; the order of presentation of summaries is randomized. We sample 100 instances from the test set of WEBMERGE and employ 5 human judges on Amazon mechanical turk (mturk.com) to perform the task; they are rewarded \$0.1 per HIT. Importantly, we are able to filter out low-quality responses from AMT judges using their answers for human summaries, as they are expected to answer unanimously *yes* for Q2 and *no* for Q3. We expect this method to improve the quality and objectivity of human evaluation.

We present evaluation results in Table 3.7. It is not surprising that human summaries are ranked 1st on grammaticality. Our GenParse-FULL method consistently outperforms its counterparts and it is ranked 2nd best followed by PointerGen and GenParse-BASE.[6] We report the system accuracy on preserving source semantics (Q2) and preventing system summaries from changing original

[6]We perform pairwise comparisons between systems. Results reveal that there is no significant difference between GenParse-BASE and PointerGen. All other differences are statistically significant according to a one-way ANOVA with posthoc Tukey HSD test ($p < 0.01$).

meanings ($\neg$Q3). Our method (GenParse-FULL) again excels in both cases. But the scores (46.8 and 53.4) suggest that ensuring abstractive summaries to preserve source content remains a challenging task, and similar findings are revealed by Cao et al. [67] and See et al. [29]. Our results are highly encouraging. The human evaluation protocol is particularly meaningful to quantitatively measure to what extent system-generated abstractive summaries remain true-to-original.

3.5 Conclusion

We propose to jointly summarize and parse to improve the grammaticality and truthfulness of summaries. We introduce a neural model combining a sequential decoder with a tree-based decoder and ensure both work in a synchronized manner. Experimental results show that our method performs on par with or superior to state-of-the-art systems on standard test sets. It surpasses strong baselines on human evaluation of grammaticality and preservation of meanings.

Table 3.4: Summarization results on Gigaword dataset. Our GenParse systems perform on par with or superior to state-of-the-art systems on the standard test set.

	Gigaword Test Set		
System	**R-1**	**R-2**	**R-L**
ABS	29.55	11.32	26.42
ABS+	29.76	11.88	26.96
Luong-NMT	33.10	14.45	30.71
RAS-LSTM	32.55	14.70	30.03
RAS-Elman	33.78	15.97	31.15
ASC+FSC1	34.17	15.94	31.92
lvt2k-1sent	32.67	15.59	30.64
lvt5k-1sent	35.30	16.64	32.62
Multi-Task	32.75	15.35	30.82
SEASS	36.15	17.54	33.63
DRGD	36.27	17.57	33.62
Struct+2Way+Word	35.47	17.66	33.52
EntailGen+QuesGen	35.98	17.76	33.63
GenParse-BASE (This work)	35.21	17.10	32.88
GenParse-FULL (This work)	**36.61**	**18.85**	**34.33**

Table 3.5: Summarization results on Newsroom, CNN/DM-R, and WebMerge datasets. Our GenParse-FULL method jointly decodes a summary and its dependency structure using a novel architecture that performs competitively against strong baselines. It outperforms both pointer-generator networks and the ablated model GenParse-BASE without using the tree-decoder.

		ROUGE-1			ROUGE-2			ROUGE-L		
	System	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
NEWSROOM	PointerGen [29]	43.73	38.83	39.94	21.82	18.97	19.56	40.15	35.65	36.66
	GenParse-BASE (This work)	41.88	36.00	37.65	20.04	16.90	17.70	38.73	33.33	34.84
	GenParse-FULL (This work)	**45.17**	**39.77**	**41.06**	**23.48**	**20.17**	**20.89**	**41.82**	**36.81**	**38.01**
CNN/DM-R	PointerGen [29]	50.91	49.82	49.26	34.73	33.32	33.16	48.10	46.95	46.49
	GenParse-BASE (This work)	48.24	46.52	46.46	31.44	29.62	29.82	45.43	43.72	43.71
	GenParse-FULL (This work)	50.15	**53.11**	**50.49**	34.51	**35.99**	**34.38**	47.33	**50.00**	**47.60**
WEBMERGE	PointerGen [29]	54.73	49.22	50.90	25.80	23.08	23.89	40.60	36.67	37.84
	GenParse-BASE (This work)	37.79	35.86	36.23	12.63	11.99	12.09	28.87	27.59	27.77
	GenParse-FULL (This work)	**62.26**	**54.69**	**57.24**	**32.10**	**28.41**	**29.58**	**48.13**	**42.54**	**44.41**

Table 3.6: We present a summary to a group of human judges. They are instructed to assess the summary for grammaticality and preservation of original meanings.

	Albert B White was born in 1856 and died on July in Parkersburg, West Virginia.
Q1	How would you rank this summary for grammaticality? ☐ 1st (best) ☑2nd ☐ 3rd ☐ 4th (worst)
Q2a	Does this summary convey the following meaning? (Albert B. White | birthYear | 1856) ☑Yes ☐ No
Q2b	Does this summary convey the following meaning? (Albert B. White | deathPlace | Parkersburg, West Virginia) ☑Yes ☐ No
Q3	Does this summary convey any additional meanings not covered by the above triples? ☑Yes ☐ No

Table 3.7: Human assessment of grammaticality and semantic accuracy of various summaries. Our GenParse-FULL achieves the best results on both aspects among all systems.

	Grammaticality				**Meaning**	
	1st	**2nd**	**3rd**	**4th**	**Q2**	**¬Q3**
Human	**73.7**	15.3	5.9	5.1	100	100
GenParse-BASE	8.5	23.7	30.5	37.3	15.8	12.7
PointerGen	5.1	18.6	**35.6**	**40.7**	38.5	50.8
GenParse-FULL	12.7	**42.4**	28.0	17.0	**46.8**	**53.4**

CHAPTER 4
CONTROL COPYING BEHAVIOR: CONTROLLING THE AMOUNT OF VERBATIM COPYING IN ABSTRACTIVE SUMMARIZATION

In this Chapter, I will describe methods for controlling the copying behavior in different stages of summarization: training, decoding and re-ranking. The proposed methods are flexible for balancing the trade-off between remaining true-to-original and being creative while generating summaries.

4.1 Background

An ideal summarizer should provide the flexibility to generate summaries with varying proportions of reused text. Such summaries are required to cater to diverse usage scenarios. E.g., system abstracts may not contain excessive copied content without proper permission—11 consecutive words or longer are considered by EU standards as the author's intellectual creation and it is thus protected by copyright law [125]. Without proper control over copying, commercial summarizers can be held liable for copyright infringements. Moreover, system abstracts with an appropriate amount of copied content are more desirable than highly abstractive ones, as they are less likely to suffer from content hallucination [126] and better at preserving the meaning of the original text.

To date, it remains poorly understood whether modern abstractive summmarization can provide the needed flexibility to control over copying and generate diverse abstracts. Abstractive summarizers using encoder-decoder architectures can either copy words from the source text or generate new words unseen in the source [29, 86, 94]. Recent work further attempted to increase the use of unseen words in summaries [127, 87]. However, in all cases, the summarizers are trained on single-reference abstracts to produce single outputs with a fixed (corpus-level) copy rate. It can take multiple reference abstracts, created for the same input text with varying degrees of copying, to teach the system to generate abstracts with similar amounts of copying. However, not only can it be time-consuming and costly to create human abstracts, but this is unlikely to be how humans learn to exercise control over copying. Without an understanding of the copy mechanism of neural abstractive models, producing abstracts with varying degrees of copying can prove daunting at best and a "mission impossible" at worst.

In this paper, our goal is to generate abstractive summaries with varying amounts of reused text by developing a general framework that learns from single reference summaries. We define *copy rate* as the percentage of summary *n*-grams appearing in the source text. A high copy rate suggests that the summary is generated largely by copying verbatim from the source text. Conversely, a low copy rate indicates there are more text shortening, word reordering, paraphrasing and abstraction involved in the generation process. We argue that abstractive summarizers are not necessarily trained on *every word* of reference summaries but they ought to separate the prediction of summary words that are *seen* in the source text from those *unseen*. The underlying principle is simple and intuitively appealing. If a summarizer is trained to predict only *seen* words, it learns to copy them

from the source text, producing extractive summaries. As more *unseen* words are used for training, the summarizer gradually transforms from copying only to both copying and generating new words not present in the source text. By employing a "mix-and-match" strategy, we enable an abstractive summarizer to generate summaries with more, or less, copying.

We frame abstractive summarization as a language modeling task and present a *decoder-only* framework for it. It uses the same Transformer architecture [128] to both encode the source text and decode the summary. All network parameters are *warm-started* using pretrained deep representations. In contrast, in a typical encoder-decoder architecture, only parameters of the encoder and decoder can be warm-started but not those of the attention/copy mechanism [129]. Further, our method allows for control over copying during both training and decoding stages of the neural model. We experiment with varying proportions of seen and unseen summary words in training to teach the summarizer to favor, or not to favor, copying. At decoding time, we compare different search strategies (best-first search vs. beam search) and reranking methods to encourage system abstracts to use wording similar to the original. Despite that only single reference summaries are available in benchmark evaluations, we are able to evaluate summary quality along multiple dimensions, using automatic metrics based on lexical similarity (ROUGE; Lin, 2004) and semantic similarity (BERTScore)[130], and through human assessment of grammaticality, informativeness, and whether system abstracts remain true-to-original. Our method demonstrates strong performance, either outperforming or performing on par with the best published results. The research contributions are summarized as follows:

- we introduce a new summarization method that provides the needed flexibility to produce a spectrum of summaries for the same input and with a varying amount of copied content. Such summaries are highly desirable to cater to diverse real-world scenarios;[1]

- our method emphasizes on in-depth analysis of the copy behavior in summarization. It frames abstractive summarization as a language modeling task and exploits multiple strategies at training and decoding stages to generate diverse summary hypotheses. We show competitive results and demonstrate the effectiveness of the proposed method on exercising control over copying.

4.2 Related Work

The significance of controlling over the copying behavior in summarization should not be underestimated. Human editors often reuse the text in the original article to produce a summary [44]. But they can adjust the degree of copying to produce a wide spectrum of summaries. E.g., human-written summaries for newswire [131, 120], meetings [132, 23], scientific articles [133] and online forums [134] contain varying amounts of reused text. Moreover, the degree of copying can have a direct impact on scores of automatic evaluation metrics. ROUGE was reported to favor summaries that use the same wording as the original [135]. If reference summaries are made by copying, system summaries with less copying and perhaps more abstraction, compression, and paraphrasing will be disadvantaged when compared against other system summaries with substantial copying.

[1] We make our implementation and models publicly available at https://github.com/ucfnlp/control-over-copying

There is thus an urgent need, and this paper makes a first attempt to present a summarization framework that is capable of producing summaries with varying amounts of reused text.

To date, various extractive and abstractive summarization techniques have been investigated [43]. However, rarely has one technique been utilized to produce both extractive and abstractive summaries for any given text. Extractive summarization selects important and non-redundant sentences from the original document(s). The sentences can be optionally compressed to remove inessential phrases, leading to compressive summaries [100, 22, 56, 53, 50]. Abstractive summarization distills the source text into its essential meanings, then performs language generation from the representation to produce an abstract [136, 64, 105, 106]. These systems rarely provide the flexibility for an end user to indicate the desired amount of reused text in the summary. To eliminate the need to develop multiple systems for extractive and abstractive summarization, we attempt to introduce control into the copying behavior of a neural abstractive summarization system.

Neural abstractive summarization has demonstrated considerable recent success. It often utilizes an encoder-decoder architecture [27, 29, 86, 93, 90]; and more recently, studies have attempted to use deep contextualized representations such as BERT [30] and ELMo [137] to give a further boost to it. An encoder network converts the source text to a fix-length vector, conditioned on which a decoder network unrolls the summary one word at a time. While it is tempting to use pretrained deep representations to "warm-start" the encoder/decoder, Khandelwal et al. [129] find that results can be less satisfying as the attention weights are still not pretrained. In this paper we adopts a *decoder-only* framework [35] where the same Transformer architecture is used for both encoding the source text and decoding the summary.

Copying can help produce unseen words. It was originally introduced to the seq2seq framework for neural machine translation [72] and later for abstractive summarization [29]. Particularly, Knowles and Koehn [138] examine the influence of context and sub-words on the copying behavior of an NMT system. To suppress copying, Kryściński et al. [87] introduce a novelty metric which is to be optimized during policy learning; and Weber et al. [127] modify the scoring function of the summary sequence at decoding time. Fan, Grangier, and Auli [139] attempt to control over summary length, entities, source style and portions. But they do not address copying. In this paper, we focus on better understanding the copying behavior of a summarization system and present effective mechanisms to control the amount of reused text. We discuss what it takes for a summarizer to copy a word without an explicit copying mechanism, and how we may control the behavior to produce summaries with more, or less, copying. In the following we describe our model in great detail.

4.3 Our Approach

We frame abstractive summarization as a language modeling task and present a *decoder-only* framework for it. It uses the same Transformer architecture [128] to both encode the source text and decode the summary. Let $\mathbf{x} = \{x_1, x_2, \ldots, x_{|\mathbf{x}|}\}$, $x_i \in \mathscr{V}$ be a sequence of source tokens and $\mathbf{y} = \{y_1, y_2, \ldots, y_{|\mathbf{y}|}\}$, $y_j \in \mathscr{V}$ be summary tokens. Our goal is to model the conditional probability distribution $P(y_j|\mathbf{y}_{<j}, \mathbf{x})$ using a Transformer-inspired architecture.

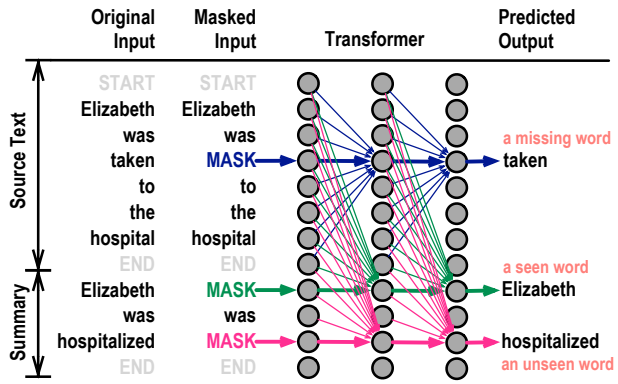

Figure 4.1: An illustration of our CopyTrans architecture. The self-attention mechanism allows (i) a source word to attend to lower-level representations of all source words (including itself) to build a higher-level representation for it, and (ii) a summary word to attend to all source words, summary words *prior to* it, as well as the token at the current position ('MASK') to build a higher-level representation.

We use byte-pair-encoding (BPE; Sennrich et al., 2016) for tokenization, with a vocabulary size of $|\mathscr{V}| = 30,522$ tokens. BPE has been shown to improve the robustness and accuracy of neural model training. We use parameter tying, allowing the same token embeddings to be used in both the input layer and final softmax layer of the Transformer model. Our method also includes three special tokens: START, END, and MASK, which respectively denote the start/end of a sequence and a "masked out" token. An illustration of our system architecture is provided in Figure 4.1.

4.3.1 Training

We construct the source sequence **x** by prepending 'START' and appending 'END' to the input text. E.g., **x** = *START Elizabeth was taken to the hospital END*, illustrated in Figure 4.1. Similarly, the target sequence **y** is constructed by appending 'END' to the summary. E.g., **y** = *Elizabeth was hospitalized END*. Our system learns to predict the target sequence one word at a time until the 'END' token has been reached. The conditional probability is shown in Eq. (4.1-4.2).

$$P(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} P(y_j|\mathbf{y}_{<j}, \mathbf{x}) \tag{4.1}$$

$$= \prod_{i=|\mathbf{x}|+1}^{|\mathbf{x}|+|\mathbf{y}|} P(z_i|\mathbf{z}_{<i}) \tag{4.2}$$

However, at training time, we argue that the system is not necessarily trained to predict *every word* of target sequences but a selected collection might suffice. Using selected target tokens provides important potential to steer the system to be more extractive than abstractive, or vice versa. We divide all tokens in the sequence $\mathbf{z} = [\mathbf{x}; \mathbf{y}]$ into three categories: (a) summary tokens *seen* in the source text, (b) summary tokens *unseen* in the source, and (c) source tokens, with the expectation that training the system to predict only *seen* summary tokens may reinforce the copying behavior, unseen tokens allow for generation, and source words enable the system to learn better token representations. By mix and matching target tokens from three categories, we enable a summarizer to generate summaries with more, or less, copying.

We randomly sample a set of tokens from each category using a Bernoulli distribution with probability p. The value of p varies by category and more analysis is provided in the experiments

section. Let $m_i \in \{0,1\}$ denote whether the i-th token of $\mathbf{z}$ is selected; its probability is defined as

$$P(m_i; p) = p^{m_i}(1-p)^{1-m_i}. \tag{4.3}$$

A selected token is replaced by 'MASK' 80% of the time, meaning that the token has been 'masked out' from the sequence $\mathbf{z}$. For 10% of the time, it is replaced by a random token from the vocabulary $\mathscr{V}$. It remains unchanged for the final 10%. In the following, we use $\mathbf{z}$ to represent the *masked* sequence, whose selected tokens are to be predicted during model training. Our loss term is defined as follows:

$$\mathscr{L}(\theta) = -\sum_{i:m_i=1} \log P(z_i|\mathbf{z}_{\leq \max(i,|\mathbf{x}|)}). \tag{4.4}$$

It is important to note that we apply a binary mask to the self-attention mechanism of the Transformer architecture to allow (a) a *source* token to attend to all source tokens including itself, and (b) a *summary* token to attend to all source tokens, summary tokens *prior to* it, as well as the current token ('MASK') in order to learn deep contextualized representations. The formulation is similar to [35]. Our binary mask is defined by Eq. (4.5). It is a square matrix whose i-th row represents the mask of the i-th token of $\mathbf{z}$. If it is a source token ($i \leq |\mathbf{x}|$), the mask allows it to attend to all source tokens ($M^{\text{att}}_{i,j} = 1$ for $j \leq |\mathbf{x}|$). If it is a summary token ($i > |\mathbf{x}|$), it can attend to all tokens prior to it as well as the current token ($M^{\text{att}}_{i,j} = 1$ for $j \leq i$).

$$M^{\text{att}}_{i,j} = \begin{cases} 1 & \text{if } j \leq \max(i,|\mathbf{x}|) \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The input of Transformer consists of embedding matrices: W_e, W_p, and W_s respectively denote the token, position, and segment embeddings [**?**]. $\mathscr{Z}$, $\mathscr{P}$ and $\mathscr{S}$ are one-hot matrices used to

retrieve embeddings for tokens in sequence $\mathbf{z}$. The token, position, and segment embeddings for the i-th token are then added up element-wisely.

$$E(\mathbf{z}) = \mathscr{Z}W_e + \mathscr{P}W_p + \mathscr{S}W_s \tag{4.6}$$

Our Transformer model takes as input embeddings $E(\mathbf{z})$ and the binary mask M^{att} to produce a sequence of deep contextualized representations $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{|\mathbf{z}|}]$. Particularly, $\mathbf{h}_i$ is used to predict the i-th 'missing' token in the sequence. We use parameter tying, allowing the same token embeddings W_e to be used in both the input layer (Eq. (4.6)) and final softmax layer of the model (Eq. (4.8)).

$$\mathbf{h} = \text{Transformer}(E(\mathbf{z}), M^{\text{att}}) \tag{4.7}$$

$$P(z_i|\mathbf{z}_{\leq \max(i,|\mathbf{x}|)}) = \text{softmax}(W_e^\top \mathbf{h}_i) \tag{4.8}$$

4.3.2 Decoding

Given a trained model and an input text, the decoding stage searches for a summary sequence that maximizes $P(\mathbf{y}|\mathbf{x})$. We present two search algorithms for this stage.

Best-first search uses a *priority heap* to keep partial summaries, which are scored according to a heuristic function f. At each iteration, the search algorithm takes the highest-scoring partial summary, extends it by one word, then pushes new summary sequences back to the priority heap. We generate k new summary sequences by selecting k words that give the highest probability of $\log P(y_j|\mathbf{y}_{<j}, \mathbf{x})$ (Eq. (4.9)) then iteratively appending the words to the partial summary. If

the highest-scoring summary in the heap concludes with an end-of-sentence symbol, it is moved to a pool of "completed summaries" for later reranking. The heap thus keeps a collection of partial summaries of *varying lengths*, which are visited according to their scores.[2] We provide an illustration of our best-first search algorithm in Algorithm 1.

In contrast, beam search is essentially breadth-first search. It maintains a beam of size k at any time step, containing partial summaries of the *same length*. For each partial summary, the algorithm extends it by one word, producing k new sequences by appending each of the k words that give the highest probability of $\log P(y_j|\mathbf{y}_{<j},\mathbf{x})$ to the partial summary. This process generates a total of $k*k$ new summary sequences by extending on each of the k partial summaries. The algorithm then selects k-best candidates, which are put in the beam for next iteration. If a candidate summary concludes with the end-of-sentence symbol, it is moved to the pool of "completed summaries".

Both best-first search and beam search employ the same scoring function that scores a candidate summary by the sum of log-likelihoods (Eq. (4.9)). However, the two differ in their search strategies—beam search visits candidate summaries according to the summary length, whereas best-first search favors candidates attaining higher scores.

$$\begin{aligned} \mathbf{y}^* &= \arg\max_{\mathbf{y}\in\mathscr{Y}} \sum_{j=1}^{|\mathbf{y}|} \log P(y_j|\mathbf{y}_{<j},\mathbf{x}) \\ \text{s.t.} \quad & y_{|\mathbf{y}|} = \textsc{End} \end{aligned} \tag{4.9}$$

[2]The size of the priority heap is capped at 1e5. If the heap has reached capacity and a new summary sequence needs to be pushed in, the lowest-scoring one will be removed from the heap.

Algorithm 1 Best-First Search

procedure BEST-FIRST(src, $\mathscr{M}$, K)
▷ Input sequence, model and beam size
  $init \leftarrow$ [START]$||src||$[END]
  $\mathscr{H}.push((\mathbf{0}, init))$ ▷ The priority queue
  $\mathscr{A}.reset()$ ▷ The answer collector
  while ($\mathscr{H}$ *is not Empty*)*and*($\mathscr{A}$ *is not full*) **do**
    $current \leftarrow \mathscr{H}.pop()$
    if *current* ends with END **then**
      $\mathscr{A}.append(current)$
      Continue
    $Candidates.reset()$
    for *each* $\mathbf{w} \in \mathscr{V}$ **do**
      $extended \leftarrow current \oplus \mathbf{c}$
      $\mathscr{S} \leftarrow -log.\mathscr{M}(\mathbf{w}|current \oplus$ MASK$)$
      $Candidates.append((\mathscr{S}, extended))$
    $topK \leftarrow K\text{-}argmin\ (Candidates)$
    $\mathscr{H}.pushAll(topK)$
  return $\mathscr{A}$

We compute $P(y_j|\mathbf{y}_{<j}, \mathbf{x})$ using our trained CopyTrans model. Importantly, the 'MASK' token is used as a prompt for the model to predict the next word. E.g., "*START Elizabeth was taken to the hospital END Elizabeth was MASK*" is a concatenation of the source text, partial summary and 'MASK' token; it is fed to the CopyTrans model where the contextualized representation of 'MASK' is used as input to a softmax layer to predict the next token $y_j \in \mathscr{V}$. In experimental results, we demonstrate that a dynamic, contextualized representation of 'MASK' performs reliably at predicting the next token. This represents an important distinction from shifting the target sequence by one position for prediction, which is common in encoder-decoder models.

Reranking A reranking step is necessary, in part because candidate summaries decoded using beam search or best-first search do not always meet the length requirement. E.g., an overly short summary containing only two words is rarely an informative summary, despite that it may give a high log-likelihood score. Below we compare three reranking strategies to offset this limitation.

Length normalization is adopted by See et al. [29] and it is frequently used in many other systems. It divides the original log-likelihood score, denoted as $\mathscr{S}(\mathbf{x},\mathbf{y}) = \log P(\mathbf{y}|\mathbf{x})$, by the total number of tokens in the summary to effectively prevent a long summary from being penalized.

$$\hat{\mathscr{S}}_{ln}(\mathbf{x},\mathbf{y}) = \mathscr{S}(\mathbf{x},\mathbf{y})/|\mathbf{y}| \tag{4.10}$$

BP-norm introduces a brevity penalty to summaries that do not to meet length expectation. As illustrated in Eq. (4.11), BP-norm performs length normalization, then adds a penalty term $\log bp$ to the scoring function. We modify the original penalty term of [141] to make it favor summaries using more copying. In Eq. (4.12), we define r to be the copy rate, i.e., the percentage of summary tokens seen in the source text, scaled by a factor c. When the copy rate r is set to 1, the penalty is dropped to 0. Yang, Huang, and Ma [141] provides a nice proof showing that this penalty term can

directly translate to a coefficient multiplied to the log-likelihood score (Eq. (4.14)).

$$\hat{\mathscr{S}}_{bp}(\mathbf{x},\mathbf{y}) = \log bp + \mathscr{S}(\mathbf{x},\mathbf{y})/|\mathbf{y}| \tag{4.11}$$

$$bp = \min(e^{1-1/r}, 1) \tag{4.12}$$

$$\exp(\hat{\mathscr{S}}_{bp}(\mathbf{x},\mathbf{y})) = bp \cdot \exp(\sum_{j=1}^{|\mathbf{y}|} \log P(y_j|\mathbf{y}_{<j},\mathbf{x}))^{1/|\mathbf{y}|} \tag{4.13}$$

$$= bp \cdot \Big[\prod_{j=1}^{|\mathbf{y}|} P(y_j|\mathbf{y}_{<j},\mathbf{x})\Big]^{1/|\mathbf{y}|} \tag{4.14}$$

Soft-bounded word reward (SBWR) is a newly introduced method by us that assigns a per-word reward to the summary. If the decoded summary is longer than expected ($i > \mathscr{L}_{\text{pred}}$), the added words receive a diminishing reward of $\sigma(\mathscr{L}_{\text{pred}} - i)$. If the summary is shorter ($i \leq \mathscr{L}_{\text{pred}}$), every word of it will receive a reward. The method thus promotes summaries of similar length to the predicted $\mathscr{L}_{\text{pred}}$. A sigmoid function is used to smooth the reward values. r is a coefficient to scale the total reward and it is tuned on the validation data.

$$\hat{\mathscr{S}}_{sbwr}(\mathbf{x},\mathbf{y}) = \mathscr{S}(\mathbf{x},\mathbf{y}) + r\sum_{i=1}^{|\mathbf{y}|} \sigma(\mathscr{L}_{\text{pred}} - i) \tag{4.15}$$

We obtain the predicted length $\mathscr{L}_{\text{pred}}$ using greedy search, then empirically offset the predicted length by three words according to validation set. In all cases, we force the decoder to never output the same trigram more than once during testing, which is a common practice to avoid repetitions [45].

4.4 Experiments

4.4.1 Data and Evaluation Metrics

We evaluate our proposed method on the sentence summarization task. The goal is to condense a lengthy source sentence to a title-like summary. Comparing to single-document summarization, sentence summarization deals less with content selection; its ground-truth summaries also contain more paraphrasing and abstraction. We conduct experiments on the Gigaword [77] and Newsroom [119] datasets. Gigaword articles were collected during 1995-2010 and Newsroom spans the range of 1998-2017. We pair the first sentence of each article with its title to form an instance. The train/valid/test splits contain 4 million/10k/1951 instances for Gigaword and 199k/21k/21k instances for Newsroom. We experiment with both datasets to understand not only the copying behavior, but also domain adaptation effects for various models. Despite that only single reference summaries are available in benchmark evaluations, we are able to evaluate summary quality along multiple dimensions, using automatic metrics based on lexical similarity (ROUGE)[42] and semantic similarity (BERTScore)[130], and through human assessment of grammaticality, informativeness, and whether system abstracts remain true-to-original.

4.4.2 Experimental Settings

We initialize the model parameters using pretrained BERT-BASE (uncased) model. The model is fine-tuned on the training split of the Gigaword (or Newsroom) dataset for abstractive summarization. Our model uses a 12-layer Transformer architecture. Its hidden state size is 768 and has 12

attention heads. We use the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate is set to *lr*=4e-5 and it is halved whenever the validation loss does not change after 40,000 training steps. We set the weight decay to be 0.01 for regular layers and no weight decay for dropout and layer-normalization. The sampling rate p is set to 0.1 for source words and 0.9 for summary words, both seen and unseen. Each model is fine-tuned for 6 epochs; an epoch takes about 5 hours on a Tesla V100 GPU. Our batch size is set to be 32.

4.5 Summarization Results

Control over copying Could we bias a summarizer to produce summaries that are more extractive than abstractive, or vice versa? If the summarizer is trained solely on summary words *seen* in the source text, will it only learn to copy words during testing but not generate new words? We seek to answer these questions in this section. Particularly, we divide all tokens selected for training into three categories: (a) summary tokens *seen* in the source text, (b) summary tokens *unseen* in the source, and (c) source tokens, with the expectation that training the system to predict only *seen* summary tokens may reinforce the copying behavior, unseen tokens allow for generation, and source words enable the system to learn richer representations. By mix-and-matching tokens, we enable a summarizer to copy more, or less.

We analyze the copy rate of various summarization models in Table 4.3. *Copy rate* is defined as the percentage of summary n-grams appearing in the source text. We set n=1/2/3/4 and the average of them. A high copy rate suggests that the summary is generated largely by copying verbatim

from the source text. We experiment with selecting varying amounts of *seen* summary tokens (●), *unseen* summary tokens (○), and *source* tokens (⊙) for training, where the number of circles is proportional to the number of tokens used in computing the loss term. All summaries in Table 4.3 are decoded using beam search (k=5) without reranking.

Our findings suggest that, the factor that makes the most impact on the copying behavior of a summarizer is the proportion of *seen* and *unseen* summary words used for training the model. If the summarizer is trained on purely *seen* words (case a. in Table 4.3), it only reuses source words during testing, despite that there is nothing to prevent the system from generating new words. The 1-gram copy rate for case a. is about 99% for both datasets, with the minor gap due to tokenization discrepancies. As more *unseen* words are used for training, the summarizer gradually transforms from copying only to both copying and generating new words not present in the source text. We observe that the ratio of seen vs. unseen words in ground-truth summaries is about 2:1 in both datasets, and NEWSROOM is slightly more extractive than GIGAWORD. Our analysis reveals that it is important to maintain a similar ratio during training in order to achieve high ROUGE scores. Pure extracts do not attain high ROUGE scores, as ground-truth summaries themselves are abstracts. Our analysis further suggests that training on source words has little impact on the copying behavior of the system, but it improves representation learning and has lead to consistently improved ROUGE-2 F-scores.

System comparison Table 4.4 shows results on benchmark summarization data containing 1951 testing instances from Gigaword. We contrast our system with summarization baselines

developed in recent years. They include lvt5k-1sent [28], Multi-Task w/ Entailment [123], SEASS (Zhou et al., 2017), DRGD [84], EntailGen+QuesGen [124], PG Networks [29], Struct+2Way+Relation [88], R3Sum [142], and BiSET [143]. Output summaries from the last four systems are graciously provided to us by the authors. We evaluate summary quality using two automatic metrics, including ROUGE[3] (Lin, 2004) that measures n-gram overlap between system and reference summaries, and BERTScore (Zhang et al., 2019) that quantifies their semantic similarity using BERT-based contextualized representations.

Results show that our system achieves competitive performance, surpassing strong systems having reported results on this dataset, as judged by both metrics. These results demonstrate the effectiveness of our Transformer-based decoder-only architecture for abstractive summarization. We observe that using beam search with reranking yields the highest results (using case g. in Table 4.3 for training). Both BP-Norm and SBWR appear to be outstanding reranking methods, better than length normalization. Our observation also suggests that best-first search and beam search can produce similar outcome, despite that the two differ in their search strategies, with beam search visiting candidates according to summary length and best-first search favoring candidates having high log-likelihood scores. We suggest future work to explore other search methods such as A* search.

Domain adaptation We investigate the effect of domain adaptation by training the model on Gigaword then testing it on Newsroom test set. Results are reported in Table 4.5. Not surprisingly,

[3] w/ options "-c 95 -2 -1 -U -r 1000 -n 4 -w 1.2 -a -m"

there is a performance degradation when testing the model in a cross-domain setting. We observe that the model with more copying (pure-extract, case e.) seem to degrade more gracefully than its counterpart (best-abstract, case f.), with a smaller performance gap in cross-domain settings. Both of our models perform competitively comparing to other baseline methods.

Human Evaluation To thoroughly analyze the quality of summaries, we ask human annotators to assess system outputs along three dimensions, including *informativeness* (Has the summary covered important content of the source text?), *grammaticality* (Is the summary sentence grammatically correct?), and *truthfulness* (Has the summary successfully preserved the meaning of the original text?). Both system and human summaries are scored according to these criteria using a Likert scale from 1 (worst) to 5 (best). We compare variants of our method generating (a) pure extracts (case e.) and (b) best abstracts (case g.), baselines of (c) PG networks, (d) R3Sum, (e) BiSET, and (f) human abstracts. Following [144], we perform Best-Worst Scaling where a human selects the best and worst summary among six candidates. The final rating of the system is computed as the percentage of times it was selected as the best minus that of the worst. We sample 200 instances from the Gigaword test set for evaluation. Each instance was assessed by five human evaluators from Amazon mechnical turk where low-quality annotations are manually removed. The results are presented in Table 4.6. We observe that human summaries (article titles) are imperfect. They can contain details that are nonexistent in the source (see Table 4.2), although they provide a means for researchers to train neural models without re-annotating reference summaries. In contrast, both of our systems perform slightly but consistently better than other baselines.

4.6 Conclusion

In this paper we present a Transformer-based, decoder-only framework to generate summaries with more, or less, copying. The proposed method can be used to generate both extractive and abstractive summaries. Our method emphasizes on in-depth analysis of the copy behavior in summarization. It exploits multiple strategies at training and decoding stages to generate diverse summary hypotheses. We show competitive results and demonstrate the effectiveness of the proposed method on exercising control over copying.

Table 4.1: Formulating summarization as a language modeling task. The first model predicts only summary words that are *seen* in the source text; the second model predicts only *unseen* words. Our method provides flexibility to control over copying by mix-and-matching the two types of behaviors.

Question: What is the most probable next word? **Hint: the word is ****seen***** in the source text.**
A 23-month-old toddler who was reportedly abducted in Pennsylvania has been found dead, a district attorney said.
Missing ___?___ Missing Pennsylvania ___?___ Missing Pennsylvania toddler ___?___ Missing Pennsylvania toddler found ___?___
Reference Summary: *Missing Pennsylvania toddler found dead*
Question: What is the most probable next word? **Hint: the word is ****unseen***** in the source text.**
Rescuers have suspended their search off the coast of Santa Cruz Island for passengers who were trapped aboard the Conception when the diving boat caught fire and sank.
Search ___?___ Search has ___?___ Search has been suspended ___?___ Search has been suspended in the ___?___ Search has been suspended in the dive boat fire off ___?___
Reference Summary: *Search has been suspended in the dive boat fire off California coast*

Table 4.2: Example system summaries produced by 1: pointer-generator networks; 2: our method (best abstract), 3: our method (pure extract), and 4: human abstract.

Source Text: Premier Chang Chun-hsiung said Thursday he is enraged and saddened by the snail-paced progress of the reconstruction of areas hardest hit by a disastrous earthquake that rattled Taiwan on Sept. 21 , 1999 .
Summary:
1: premier expresses condolences for taiwan quake victims
2: premier angry over reconstruction of quake - hit areas
3: premier enraged and saddened by earthquake reconstruction
4: premier enraged by slow progress of post-quake reconstruction
Source Text: A blue-ribbon panel of experts said on Wednesday that German economic growth will grind to a halt next year , raising doubts about Berlin 's plans to shield Europe 's biggest economy from the global turmoil .
Summary:
1: german experts raise doubts about economic recovery
2: experts say german growth will grind to a halt next year
3: german experts to grind to halt next year
4: german economy will grind to halt in 2009 , say experts

Table 4.3: The copy rate and ROUGE-2 of various summarization models. We define *copy rate* as the percentage of summary *n*-grams appearing in the source text, where *n*=1/2/3/4 as well as an average of them. We experiment with selecting varying amounts of *seen* summary tokens (dark circle), *unseen* summary tokens (light circle), and *source* tokens (dot circle) for training. A circle corresponds to about 5 million tokens for GIGAWORD and 385k tokens for NEWSROOM, which are used to compute the loss term.

	GIGAWORD						NEWSROOM					
Training Loss	**1-gram**	**2-gram**	**3-gram**	**4-gram**	**Average**	**R-2**	**1-gram**	**2-gram**	**3-gram**	**4-gram**	**Average**	**R-2**
a. ●●●●	98.90	55.92	33.85	20.32	52.25	14.51	99.19	65.28	45.25	31.16	60.22	21.51
b. ●●●●○	86.74	46.14	27.15	16.14	44.05	19.37	92.32	57.60	38.14	25.11	53.29	**23.93**
c. ●●●●○○	80.96	40.58	23.08	13.15	39.44	**20.00**	87.80	52.67	33.84	21.17	48.87	23.90
d. ●●○○	73.98	34.89	19.19	10.55	34.65	19.20	82.23	46.55	28.54	17.37	43.67	22.97
e. ●●●●⊙⊙	98.57	56.33	35.10	21.72	52.93	15.21	98.71	64.35	44.61	30.69	59.59	21.81
f. ●●●●○⊙⊙	86.29	45.91	27.07	16.06	43.83	19.55	91.52	56.36	36.93	24.12	52.23	**24.14**
g. ●●●●○○⊙⊙	80.56	40.32	22.66	12.87	39.10	**20.37**	87.59	52.25	33.50	21.43	48.69	24.04
h. ●●○○⊙⊙	74.22	35.09	19.13	10.49	34.73	19.39	82.41	47.16	29.16	17.92	44.16	23.10

Table 4.4: Summarization results on the Gigaword test set. The lower part of the table contains results from our system.

System	**R-1**	**R-2**	**R-L**	BERT-S
lvt5k-1sent	35.30	16.64	32.62	–
Multi-Task w/ Entailment	32.75	15.35	30.82	–
SEASS	36.15	17.54	33.63	–
DRGD	36.27	17.57	33.62	–
EntailGen+QuesGen	35.98	17.76	33.63	–
PG Networks	34.19	16.92	31.81	58.32
Struct+2Way+Relation	35.61	18.02	33.45	58.84
R3Sum	36.36	18.23	33.85	56.74
BiSET	38.45	19.53	36.04	57.10
Best-first Search	39.07	20.28	36.49	61.27
Beam Search	38.87	20.37	36.52	61.47
Beam+LengthNorm	39.10	20.25	36.55	61.41
Beam+BPNorm (c=0.55)	**39.19**	20.38	**36.69**	61.46
Beam+SBWR (r=0.25)	39.08	**20.47**	36.68	**61.51**

Table 4.5: Summarization results on the Newsroom test set. The top four systems are trained on Newsroom training data, whereas the bottom two systems are trained on Gigaword.

	System	**R-1**	**R-2**	**R-L**	BERT-S
Newsroom	PG Networks	39.86	19.51	36.61	62.01
	Struct+2Way+Rel.	40.54	20.44	37.40	62.05
	Ours (pure-ext)	43.21	21.81	40.05	63.68
	Ours (best-abs)	**45.93**	**24.14**	**42.51**	**66.20**
Giga	Ours (pure-ext)	39.44	17.32	36.10	61.00
	Ours (best-abs)	**40.89**	**19.11**	**37.60**	**62.74**

Table 4.6: Human assessment of *informativeness*, *grammaticality*, *truthfulness*, and best-worst scaling.

System	**Inform.**	**Gramm.**	**Truthful.**	**Bst-Wst**
Human	2.801	2.831	2.778	-0.001
PG Networks	2.768	2.697	2.678	-0.058
R3Sum	2.748	2.680	2.709	-0.009
BiSET	2.740	2.634	2.738	-0.006
Ours (pure-ext)	**2.846**	2.817	2.785	0.032
Ours (best-abs)	2.843	**2.855**	**2.865**	**0.042**

CHAPTER 5
CONTROL OVER DESIRED PROPERTIES: A NEW APPROACH TO OVER-GENERATING AND SCORING ABSTRACTIVE SUMMARIES

In this Chapter, I will demonstrate a more general approach suitable for varying desired summary properties. It over-generates summaries with different lengths and then passes those summaries to a selector for selecting admissible summaries. Particularly, in this work we designed two selectors the ”best-overall quality” selector and the ”optimal-length” selector for quality and length controlling.

5.1 Background

The learning objective of a modern abstractive summarizer is to produce system outputs that resemble reference summaries on a word-to-word basis. It does *not* promote outputs that possess multiple desirable properties, i.e., capturing the most important information, being faithful to the original text, grammatical and fluent, though some of these properties are exhibited by system abstracts as a natural outcome of a learned summarizer [29, 65, 46, 86, 90, 94, 144, 91, 145, 146]. Without direct optimization of desired properties, system abstracts often change the meaning of the original document or fail to convey the main concepts [147].

Table 5.1: Example of alternative summaries generated from the source text *Admissible* summaries are marked by ✔. System summaries that fail to preserve the meaning of the source input are marked by ✘.

Source Text
• Police arrested five anti-nuclear protesters Thursday after they sought to disrupt loading of a French Antarctic research and supply vessel, a spokesman for the protesters said.
Summary
✔ Police arrest anti-nuclear protesters
✔ Protesters target French research ship
✘ French police arrest five anti-nuclear protesters
✘ Police arrest five anti-nuclear protesters in Antarctica
✘ Police arrest five anti-nuclear protesters at French Antarctic

Table 5.2: An example of the difference between left-to-right and confidence-driven summary generation (LEFT) A single summary is produced in a left-to-right order. (RIGHT) Four summaries are generated in a confidence-driven mode. The most confident words are generated first, less vital ones later. Our generator learns to dynamically add or remove content given a target length to produce summaries of varying lengths—short, medium and long. The output is a diverse set of alternative summaries.

Source Text: A court here Thursday sentenced a 24-year-old man to 10 years in jail after he admitted pummelling his baby son to death to silence him while watching television.	
Left to Right Generation (1 Summary)	**Confidence Driven Generation (4 Summaries)**
Man who	Man gets 10 years
Man who killed [. . .]	Man who kill the baby gets 10 years
Man who killed baby to hear television better gets 10	Man who kill the baby to hear television gets 10 years
Man who killed baby to hear television better gets 10 years	Man who kill the baby to hear television better gets 10 years

In this paper, we propose a new approach to over-generate and select *admissible* summaries, which allows a summarizer to juggle multiple objectives and strike a good balance between them [148]. Our approach consists of two stages. Given a source text, a *generator* explores the space of all possible lengths to produce multiple variants of the target summary that contain diverse content. We then devise *selectors* to validate the quality of alternative summaries to predict whether they are admissible. Our selection mechanism can be customized to suit particular needs without changing the generation space. Both stages can be effectively trained, optimized and evaluated.

Crucially, we take a confidence-driven approach to summary generation rather than using a left-to-right order. Beginning writers and language learners do not write in a strict sequential manner. In a similar vein, our generator produces a summary by "filling-in-the-blanks" with appropriate words. The most confident words are generated first, less vital ones later. With confidence-driven generation, our summarizer learns to dynamically add or remove content, and even paraphrase to produce a summary of a given length. In Table 5.2, we show an example illustrating the difference between our method and left-to-right generation. Our method dramatically enhances the capability of the generator, making it possible to explore summaries of varying lengths.

Identifying admissible summaries with desired properties is critical for a summarizer. Summaries of very short lengths may fail to capture the main concepts, and this kind of incomplete or partial information can lead to false assumptions about the original content. Moreover, summaries of moderate lengths may still contain hallucinated content that is nonexistent in the source text [149]. We present two summary selectors to combat these issues. Our first selector aims to predict what summary length is most suitable for a source text, whereas a second selector puts special emphasis on the overall quality of the system summary, in particular its faithfulness to the original text [150, 151].

A novel dataset has been introduced in this work where we associate a source text with multiple summaries, and *admissible* ones are manually labelled by human annotators. Not only can the dataset be used to judge the effectiveness of summary selectors, but it provides a new testbed for future summarizers to compare their outputs against multiple reference summaries, which is key to improve the reliability of evaluation results [152]. We have focused on generating abstractive

summaries from single source sentences, but the insights gained from this study could inform the design of summarizers of all forms. Our method also has a great potential to incorporate human-in-the-loop to teach the model to select the best summary. The main contributions of this paper are:

- We propose a new approach to generate multiple variants of the target summary that have varying lengths, then score and select the best summaries according to our needs.
- Our generator controls over the length of the summary, which is especially well-suited when space is limited. Our selectors are designed to predict the optimal summary length and put special emphasis on faithfulness to the original text.
- Our experiments on benchmark summarization datasets suggest that this paradigm can surpass results of previous studies or rival state-of-the-art. We conclude with a discussion of our key findings, which has implications for the development of robust abstractive summarizers. [1]

5.2 Related Work

It is important for neural abstractive summarizers to produce summaries that are faithful to the original texts [67, 153, 89, 154, 155, 156]. However, it remains questionable as to whether a summarizer must acquire that ability by learning from human reference summaries, or possibly

[1] Our code and annotated data are made available on Github at `https://github.com/ucfnlp/varying-length-summ`

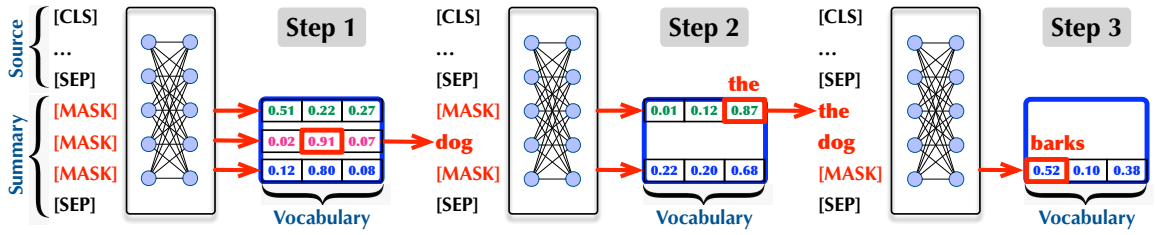

Figure 5.1: An illustration of the generation process. A sequence of placeholders ("[MASK]") are placed following the source text. Our model simultaneously predicts the most probable tokens for *all positions*, rather than predicting only the most probable *next* token in an autoregressive setting. We obtain the token that has the highest probability, and use it to replace the [MASK] token of that position. Next, the model makes new predictions for all remaining positions, conditioned on the source text and *all summary tokens seen thus far*. Our generator produces a summary having the *exact given length* and with a proper endpoint.

through external resources such as textual entailment predictions [150]. In this paper, we present a two-stage strategy to over-generate, then score system summaries externally for faithfulness and overall quality.

Previous work has sought to control various aspects of the generated summary, including the style, length and amount of reused text [58, 157, 139, 158, 159, 97]. In contrast, our generator focuses on producing *multiple* variants of the target summary that have diverse content and varying lengths. It offers precise control over the length of the summary, which has an important implication for fair comparison between different summarization systems [160, 161].

Our methodology allows for greater flexibility in designing summary selectors. The selectors may allow multiple admissible summaries to be identified for any source input according to users'

needs. On the contrary, post-editing of system summaries through a set of basic operations such as insertion and deletion [162, 163, 164, 165] may have intrinsic limitations by learning from single reference summaries to produce single outputs. In this paper, we provide a new dataset where each source text is associated with multiple admissible summaries to encourage diverse outputs.

Our generator is inspired by unsupervised pretraining of deep neural models [137, 33, 30, 39, 38, 37] and non-autoregressive machine translation [166, 167]. Distinct from these is our confidence-driven generation that goes beyond left-to-right order. It uses a denoising objective during training and is conveniently transformed into a semi-autoregressive generator at test time. We introduce a customized beam search algorithm to promote the generation of diverse outputs. In the following section, we describe in detail our two-step strategy.

5.3 A Confidence-Driven Generator

We seek to produce a highly diverse set of alternative summaries from any source input, but standard neural language generators with beam search only produce high-likelihood sequences rather than diverse ones [168]. To address this limitation, we devise a new generator that is capable of producing summaries of varying lengths. A long summary can cover more important information of the source text, whereas a short summary is easy-to-read. Moreover, it produces a summary having the *exact given length* and with a proper endpoint. This is achieved by shifting away from left-to-right generation but building a summary using a confidence-driven approach.

Our generator is illustrated in Figure 5.1. To generate a summary of L tokens, we place a number of [MASK] tokens following the source text, which serve as "placeholders" for summary tokens. Importantly, our generator simultaneously predicts the most probable tokens for *all positions*, as opposed to predicting only the most probable *next* token in an autoregressive setting. We obtain the token that has the highest probability across all positions, and use it to replace the [MASK] token of that position. Next, the model continues to make predictions for all remaining positions, conditioned on the source text and the summary tokens seen thus far of varying positions.

Let $\mathbf{x} = \{x_i\}_{i=1}^{N}$ be the source and $\mathbf{y} = \{y_j\}_{j=1}^{M}$ the summary sequence. Our confidence-driven generation process defines a new order of summary tokens, $\mathbf{o} = \{o_j\}_{j=1}^{M}$, $o_j \in [M]$, according to which $P_\theta(\mathbf{y}|\mathbf{x})$ is factorized into a product of conditional probabilities $P_\theta(y_{o_j}|y_{\mathbf{o}_{<j}}, \mathbf{x})$ (Eq. (5.1)), where θ are model parameters to be optimized during training. Our learning objective is to minimize the negative data log-likelihood (Eq. (5.2)) to predict missing tokens $y^*_{o_j}$ conditioned on the source text $\mathbf{x}$ and the summary tokens seen thus far $y_{\mathbf{o}_{<j}}$.

$$P_\theta(\mathbf{y}|\mathbf{x};\mathbf{o}) = \prod_{j=1}^{M} P_\theta(y_{o_j}|y_{\mathbf{o}_{<j}}, \mathbf{x}) \tag{5.1}$$

$$\mathscr{L}(\theta) = -\sum_{j=1}^{M} \log P_\theta(y^*_{o_j}|y_{\mathbf{o}_{<j}}, \mathbf{x}) \tag{5.2}$$

Our generator is trained with a denoising objective. It consists of a decoder-only architecture with 12 Transformer blocks [35]. Given a source text and a summary, we replace a portion of their tokens by the [MASK] token, and the model is trained to reconstruct the original data from the corrupted text. It differs from autoregressive models in that the context of each position can consist of tokens from both left and right—a source word can attend to other source words and a summary

word can attend to source words and summary words seen thus far *of varying positions*—hence capturing a bidirectional context. The training procedure is thus analogous to that of permutation-based language modeling [169].

Our training schedule begins with masking out 10% of source tokens and linearly decreases it to 0% throughout training steps. Masking out a portion of source tokens helps the model learn contextualized representations given bidirectional context. On the target side, the schedule begins with masking out 90% of summary tokens and linearly decreases it to 60%. It allows the model to learn to predict missing summary tokens and copy source tokens to the summary. When a token is chosen, it is replaced with the [MASK] token 80% of the time, a random token of the vocabulary 10% of the time, and remains unchanged otherwise.

In Table 5.3, we present example summaries produced by our new confidence-driven generator for a source input. The summaries have varying lengths and degrees of details. Our generator learns to add or remove content, and even paraphrase to produce a summary of a given length. We adjust the target summary length (L) to produce diverse summaries. Moreover, there exists more than one admissible summaries that capture the important information of the source text, while being grammatical and faithful to the original. It is important to note that, to decode the best summary of length L, our generator requires a *position-aware* beam search algorithm to explore the space of candidate summaries, which is described next.

Algorithm 2 Position-Aware Beam Search

procedure PosAwareBeam(SourceText, L, K)
    ▷ L is the summary length and K is the beam size.
    $\mathscr{S}_0 \leftarrow \{[\text{MASK}] \times L\}$ ▷ Initial summary.
    $\mathscr{M}_0 \leftarrow [\mathbf{1}]_{L \times |\mathscr{V}|}$ ▷ A binary mask of L positions.
    $\mathscr{H} \leftarrow \{(0, \mathscr{S}_0, \mathscr{M}_0)\}$ ▷ A priority queue.
    for $j = 1, \ldots, L$ **do**
        $Candidates \leftarrow \{\}$
        for $hyp \in \mathscr{H}$ **do**
            $score', \mathscr{S}', \mathscr{M}' \leftarrow hyp$
                ▷ Estimate token probabilities.
            $\mathscr{P}_{L \times |\mathscr{V}|} \leftarrow \text{Gen}(\text{SourceText}, \mathscr{S}')$
            $\mathscr{P}' \leftarrow \mathscr{P} \odot \mathscr{M}'$
                ▷ Record K-best tokens and positions.
            for $s_k, w_k, p_k \in Top\text{-}K\text{-}Scores(\mathscr{P}')$ **do**
                $score'' \leftarrow score' + s_k$
                $\mathscr{S}'' \leftarrow replace(\mathscr{S}', p_k, w_k)$
                $\mathscr{M}'' \leftarrow replace(\mathscr{M}', p_k, [\mathbf{0}]_{\mathbf{1} \times |\mathscr{V}|})$
                $Candidates.add((score'', \mathscr{S}'', \mathscr{M}''))$
        $\mathscr{H} \leftarrow Top\text{-}K\text{-}Scores(Candidates)$
    return $\mathscr{H}_0$ ▷ The best summary of length L.

5.3.1 Position-Aware Beam Search

A position-aware beam of size K not only contains the K-best candidate summaries having the highest log-likelihood at any time step, but it also records the positions of summary tokens seen thus far for each candidate summary. The tokens of candidate summaries can be decoded in any order and occur in different positions, marking an important distinction between position-aware and traditional beam search [170]. The method is realized by associating each candidate summary with a binary matrix $\mathscr{M} \in \{0,1\}_{L\times|\mathscr{V}|}$, which records what positions have been filled by which summary tokens and what positions remain available.

Concretely, we use $\mathscr{S}'$ to denote a candidate summary, $score'$ is its data log-likelihood and $\mathscr{M}'$ is a binary mask (Line 9). Our generator predicts the token probabilities $\mathscr{P}_{L\times|\mathscr{V}|}$ for all positions, conditioned on the source text and the summary tokens seen thus far. The binary mask $\mathscr{M}'$ indicates positions that remain available (Line 11–12). We obtain the top-K tokens that have the highest probability scores across all positions, record their summary hypotheses and likelihood scores. These positions are then marked as taken (Line 14–18).

The decoding process continues until all of the L positions are filled by summary tokens. This makes our method different from traditional beam search, the latter terminates when an end-of-sequence symbol [SEP] is generated for the summary. Particularly, our method is advantageous as it exerts *precise control* over the summary length. The model learns to decide what content to be included in the summary given the limited space available, yielding summaries with varying degrees of details.

5.4 The Selectors

We present two selectors to respectively assess the overall quality of the summary and predict the optimal summary length. Our selectors assume the role of a responsible agent that, when provided with a source text and multiple alternative summaries, can effectively recognize the *admissible* ones. It has the potential to incorporate human-in-the-loop in future to teach the model to select best summaries.

5.4.1 Best Overall Quality

Our goal is to build a selector to discern the difference between high and low-quality summaries. In an ideal scenario, we have human annotators to vet each source text/summary pair, the annotated data are used to train the selector. The process, however, is both expensive and time-consuming. Inspired by Kryściński et al. [147], we automatically construct a large number of minimally different pairs, where a positive instance comprises of the source text and its ground-truth summary, and a negative instance includes the source text and a corrupted summary. We experiment with various means to generate corrupted summaries from a ground-truth summary. The corruptions should resemble common mistakes made by neural abstractive summarizers, including generating factually incorrect details, failing to convey the main points of the source text, and being ungrammatical. The corruption types experimented in this paper are illustrated in Table 5.4.

Distinguishing our work from that of Kryściński et al. [147] are (i) *Search and Replace*, we swap the ground-truth summary with a similar summary in the training set that have $\geq$4 common bigrams to form a negative instance. (ii) *Swap Segments* splits a ground-truth summary into two parts of similar lengths, then swaps them to produce an ungrammatical summary. (iii) *Incomplete Summary* replaces a ground-truth summary by one of its sentence constituents, yielding a corrupted summary that fails to convey the main ideas. These corruptions are designed to emulate system summaries that are too short to capture the main concepts, or contain hallucinated content that is not found in the source text.

We next build a binary classifier to predict if a summary is admissible given the source text. To distill information from the source text and the summary, we encode them into hidden vectors using RoBERTa [31]. These are denoted by $\mathbf{h_x}$ and $\mathbf{h_y}$, respectively. We create a vector for the pair, $\mathbf{h} = \mathbf{h_x} \oplus \mathbf{h_y} \oplus (\mathbf{h_x} - \mathbf{h_y}) \oplus (\mathbf{h_x} * \mathbf{h_y})$, consisting of a concatenation of the two hidden vectors, their absolute difference $(\mathbf{h_x} - \mathbf{h_y})$ and their element-wise product $(\mathbf{h_x} * \mathbf{h_y})$. $\oplus$ is a concatenation of vectors. The output vector $\mathbf{h}$ is expected to capture the gist of the source text and the summary, and a similar approach is being used for natural language inference [86]. The vector $\mathbf{h}$ is fed to a feed-forward layer to predict whether the summary is admissible given the source text. We have chosen to design the selector as a classifier rather than a ranking model because there can exist multiple, equally valid summaries for any source input. The classifier allows us to identify admissible summaries that are not only true-to-original but has the best overall quality.

5.4.2 Best Summary Length

Finding a suitable length for the summary is one of the most important open problems in automatic summarization [161, 171]. A summary should be shorter than the original, but long enough to include the most important information. Length normalization seeks to rescale the log-likelihood score of a summary, denoted by $\mathscr{S}(\mathbf{x},\mathbf{y}) = \log p_\theta(\mathbf{y}|\mathbf{x})$, by its length $|\mathbf{y}|$, with an exponent p (Eq. (5.3)). It is used by some neural abstractive summarizers [29, 37]. However, the method does not consider the density of information in the source text and it may still generate ultra-short summaries.

$$\mathscr{S}_{ln}(\mathbf{x},\mathbf{y}) = \mathscr{S}(\mathbf{x},\mathbf{y})/|\mathbf{y}|^p \tag{5.3}$$

Instead, we attempt to estimate the appropriate length of the summary given a source text, denoted by $\mathscr{L}_{\text{pred}}$, and reward a system summary if it stays close to the estimated length [172]. Concretely, we assign a per-word reward to the summary, represented by $r\min(|\mathbf{y}|,\mathscr{L}_{\text{pred}})$ (Eq. (5.4)). A system summary continues to be rewarded until it reaches the predicted length ($|\mathbf{y}| \leq \mathscr{L}_{\text{pred}}$). Beyond that, increasing the length of the summary does not lead to additional rewards. We obtain the predicted length $\mathscr{L}_{\text{pred}}$ using a baseline abstractive summarizer, which takes the source text as input and greedily decodes a summary in a left-to-right manner until an end-of-sequence symbol is predicted; $\mathscr{L}_{\text{pred}}$ is the length of the decoding sequence. r is a coefficient to scale the reward and it is tuned on the validation data. Finally, the reward-augmented log-likelihood $\mathscr{S}_{rwd}(\mathbf{x},\mathbf{y})$ is used

as a scoring function to rank all summary hypotheses of varying lengths.

$$\mathscr{S}_{rwd}(\mathbf{x},\mathbf{y}) = \mathscr{S}(\mathbf{x},\mathbf{y}) + r\min(|\mathbf{y}|, \mathscr{L}_{\text{pred}}) \tag{5.4}$$

5.5 Experiments

Datasets We perform extensive experiments on Gigaword [77] and Newsroom [119] datasets. The goal is to generate an abstractive summary from a lengthy source sentence. For each article, we pair its first sentence with the title to form a summarization instance. Both datasets contain large collections of news articles. Gigaword (1995–2010) contains 3,810,674 / 10,000 / 1,951 instances, respectively, in the train, validation and test splits. Newsroom (1998–2017) contains 199,341 / 21,530 / 21,377 instances, respectively. We conduct experiments on both datasets to demonstrate the generality of our two-staged strategy. Our method generates a diverse set of summaries from a source sentence in stage one, then score and select admissible summaries in stage two.

The system summaries are evaluated using both automatic metrics (ROUGE; Lin, 2004) and human evaluation of information coverage, grammaticality and faithfulness to the original text. We introduce a new dataset where a source sentence is associated with multiple summaries, and admissible ones are labelled by human annotators (§5.5.1). The dataset will serve as a useful testbed for future summarization research, where multiple reference summaries is key to improve the reliability of evaluation results [152]. This paper focuses on generating abstractive summaries

[2]Our experiments are performed on the original Gigaword dataset [77] without anonymization. The data provided by Rush et al. [27] replaced all digit characters with # and replaced word types seen less than 5 times with UNK.

from single source sentences. However, we expect the insights gained from this study to inform the design of future summarizers of different kinds.

Experimental Setup Our generator is initialized with RoBERTa-BASE [31] due to its high performance on generation-related tasks. We use Byte Pair Encoding [140] with a vocabulary of 50,265 tokens. The model contains 12 Transformer blocks [15], with a hidden size of 768 and 12 attention heads, for a total of 110M parameters. We fine-tune the model on the train split of Gigaword and Newsroom, respectively, before applying it to the test sets. The model is fine-tuned for 20 epochs. Each epoch contains 24k / 1.5k batches and our batch size is 128. The model uses 10k / 1k warm-up steps, respectively, for Gigaword and Newsroom. We use the AdamW [173] optimizer with an initial learning rate of 1e-4. The momentum parameters are set to 0.9 and 0.999. On a deep learning workstation equipped with 2x Titan RTX GPUs, our model takes 64 and 5.5 hours to fine-tune on Gigaword and Newsroom. At test time, our beam size is K=20. The model produces summaries ranging from L = 7 to 16 tokens for a given source sentence.

Our selector for best overall quality is trained using 1.8M instances automatically constructed from the train split of Gigaword. The set is balanced with an equal number of positive and negative instances. 226k instances are created with the type of Search and Replace, and 400k instances are created using each of the four remaining corruption types. Our selector for best summary length is unsupervised and requires no training. The reward coefficient r is set to 2.0 across all experiments.

5.5.1 Experimental Results

Automatic Evaluation In Table 5.6, we present results on Gigaword and Newsroom test sets evaluated by ROUGE [42]. We report R-1, R-2 and R-L F_1-scores that respectively measure the overlap of unigrams, bigrams, and longest common subsequences between system and reference summaries. For each summarization instance, our generator produces multiple alternative summaries, ranging from L=7 to 16 tokens. E.g., "Daiwa Bank." corresponds to four tokens, 'Dai', 'wa', 'Bank' plus an ending period. Our BEST-QUALITY and BEST-LENGTH selectors each identifies a single best summary from the set of alternative summaries for each summarization instance.

We observe that the BEST-LENGTH selector has achieved the highest scores. It performs better than using any single target length for all summaries. Among summaries of different lengths, the highest R-2 F_1-scores are obtained when the target summary length is set to 11 and 12 tokens, respectively, for Gigaword and Newsroom. This is close to the median length of reference summaries, which are 12 and 13 tokens for these datasets. Our findings show that, the target summary length can make a non-negligible impact on automatic evaluation results. It is best for system summaries to be long enough to include the most important information to achieve satisfying results.

In Table 5.5, we report results on the Gigaword test split that contains 1,951 instances. Our approach is compared against strong neural abstractive systems, including PEGASUS [38], UniLM [35] and MASS [36]. These systems draw on large-scale unsupervised pretraining to improve the quality of summaries, yielding some of the best reported results. In comparison, our BEST-LENGTH selector either surpasses or performs comparably to these systems. The summaries

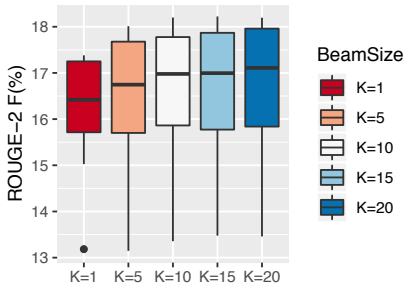

Figure 5.2: Effectiveness of position-aware beam search (§5.3.1). A larger beam tends to give better results.

selected by it achieve the highest R-2 F_1-score of 20.4%. We further choose the summary that yields the highest score for each instance, creating an oracle set of summaries, which yield a R-2 F_1-score of 33.4%. The results indicate that, with better summary selectors, there is a great potential that we can further boost summarization performance.

In Figure 5.2, we investigate the effectiveness of our position-aware beam search (§5.3.1). The beam size K is set to $\{1, 5, 10, 15, 20\}$. We report the average R-2 F_1-score across summaries of all lengths. Results show that our position-aware beam search is effective at decoding summaries and works robustly across a range of beam sizes. A larger beam (K=20) tends to give better results.

Human Evaluation We are interested in a holistic evaluation of the multiple alternative summaries produced by the generator. To accomplish this, we develop a new dataset containing 500 summarization instances randomly sampled from the Gigaword test set. Our generator produces 7 alternative summaries for each instance, which have varying lengths that range from L= 7 to 13 tokens. We recruit human evaluators to judge the quality of each summary given its source text.[3]

[3]Our annotated dataset is available on Github at `https://github.com/ucfnlp/varying-length-summ`

Our annotation interface is presented in Table 5.7. A human annotator is instructed to read over all summaries before seeing the source text. It allows him/her to effectively recognize any hallucinated content that is not found in the source text. The annotator is asked to answer three yes-no questions. They include (a) has the summary successfully convey the main points of the source text? (b) does the summary preserve the meaning of the source? (c) is the summary grammatical? A native speaker creates gold-standard annotations for multiple instances, they are shared with all annotators to provide guidance. Our annotators are recruited using Appen (`appen.com`). It is a crowdsourcing platform similar to Amazon Mechanical Turk (`mturk.com`), but provides great quality control mechanisms to ensure high-quality work.

We recruit 5 annotators to judge the quality of each summary. A summary is deemed *admissible* under a criterion if the majority answer is yes. We observe that, 74.2% of summaries produced by our generator are admissible under all three criteria. The results suggest that our generator is able to produce multiple, equally valid summaries for a given source text. We additionally examine the percentage of admissible summaries under each criterion, results are shown in Table 5.8. Grammaticality has the best performance (96.5%), followed by truthfulness (82.6%) and content coverage (80.7%). There appears to be room for improvement for the latter two aspects. Moreover, the summaries chosen by our BEST-QUALITY selector demonstrate a high admissible rate—93%, 90.8% and 97%—respectively for the three criteria, suggesting the effectiveness of the selector. Further, we observe a discrepancy between ROUGE and human judgments [174] as summaries yielding highest ROUGE scores are not always deemed admissible by human evaluators. We hope

this dataset provides a testbed for future summarizers to be judged on their ability to produce multiple summaries per instance rather than a single summary.

In Table 5.3, we show example system summaries and the order in which summary tokens are produced. E.g., {2,5} indicate the two tokens "Bo-J" (Bank of Japan) are generated the 2nd and 5th place in the summary. We find that our generator can effectively decide what content should be included in the summary given the limited space available, yielding summaries with varying degrees of details. Important spans such as "calls for calm" tend to be generated first, less vital ones later. Our findings corroborate the hypothesis that a masked language model may enable generation in a flexible word order [175]. Further, we observe that the order in which tokens are generated is related to their dependencies ("call→for"), which supports the findings of Clark et al. [176].

5.6 Conclusion

We investigate a new approach to neural abstractive summarization that focuses on producing multiple summary hypotheses with varying lengths and diverse content. Our selectors are designed to identify summaries that have the optimal length and the best overall quality. The approach obtains state-of-the-art results on summarization benchmarks and opens up a potential new avenue for customizing summary selectors to suit users' needs.

Future work includes extending this research to long documents. Our confidence-driven generator and the selectors could potentially be extended to operate on spans of text [177] rather than

individual tokens, thus allowing for efficient generation of summary hypotheses that have varying degrees of details and identification of admissible summaries or summary segments.

Table 5.3: An example of generated summaries with varying lengths. The target summary length L is adjusted to produce alternative summaries that have diverse content. Our generator can dynamically add or remove content, and paraphrase to produce a summary of a given length. The numbers indicate the order in which the summary tokens are generated. "BoJ" stands for "Bank of Japan". It maps to two tokens according to Byte Pair Encoding (BPE). Each summary has an ending period, so the last word also maps to two tokens.

Input	**The Bank of Japan appealed to financial markets to remain calm Friday following the US decision to order Daiwa Bank Ltd. to close its US operations.**
L=6	BoJ calls for calm . (2,5 4 6 3,1)
L=7	BoJ calls for market calm . (3,7 4 5 6 2,1)
L=8	BoJ urges markets to remain calm . (5,7 6 4 3 1 8,2)
L=9	BoJ urges financial markets to remain calm . (6,2 7 4 5 9 1 8,3)
L=10	BoJ calls for calm after Daiwa closure . (1,2 6 7 5 8 10,4 9,3)
L=11	BoJ calls for calm after Daiwa Bank closure . (1,2 6 7 5 8 11,4 3 10,9)
L=12	BoJ calls for calm after Daiwa Bank closure order . (2,3 5 6 1 11 8,7 9 12 10,4)
L=13	BoJ urges markets to remain calm after Daiwa Bank closure . (6,13 8 7 9 11 4 10 5,2 1 12,3)
L=14	BoJ calls for calm after Daiwa Bank 's US closure . (3,4 7 8 2 14 13,6 5 10,9 11 12,1)
L=15	BoJ calls for calm after US order for Daiwa Bank to close . (10,3 4 5 2 15 8 13 14 9,6 7 11 12,1)
L=16	BoJ calls for calm after US order on Daiwa 's US operations . (3,5 4 7 2 16 13 12 14 8,6 11,10 9 15,1)

Table 5.4: A positive instance for the selector consists of a ground-truth summary (marked by ✔) and its source text. A negative instance consists of a corrupted summary (✘) and its source text. *Entity Replacement*: replacing a named entity of the ground-truth summary with a random entity. *Negation*: negating a ground-truth summary sentence. *Incomplete Summary*: replacing the ground-truth summary with one of its sentence constituents to produce a corrupted summary that contains 5 words or less. *Search and Replace*: swapping the ground-truth summary with a similar summary in the training set that have 4 or more common bigrams. *Swap Segments*: splitting the ground-truth into two parts of similar length, the parts are swapped to produce an ungrammatical summary.

Entity Replacement
• German art experts have authenticated a painting believed to be the last portrait ever made of the composer Wolfgang Amadeus Mozart, the body which runs Berlin's museums said on Thursday.
✔ German experts identify last known portrait of **Mozart**
✘ German experts identify last known portrait of **Mount Mayon's**

Negation
• US Secretary of State Condoleezza Rice suggested Tuesday that International Atomic Energy Agency chief Mohamed ElBaradei should not interfere in diplomatic issues after he warned against the hasty use of force in the Iranian nuclear dispute.
✔ Rice suggests IAEA chief **should** stay clear of diplomacy
✘ Rice suggests IAEA chief **shouldn't** stay clear of diplomacy

Incomplete Summary
• Total Hong Kong dollar deposits grew 2.2 percent in March, compared to 2.1 percent in February, according to the Hong Kong Monetary Authority.
✔ HK Bank Deposits **Increase in March**
✘ **Increase in March**

Search and Replace
• Israel is on course to complete the main tranche of its controversial West Bank security barrier in 2004 and wrap up the project in the following year, the defence ministry said Wednesday
✔ **Israel surges ahead with West Bank barrier construction**
✘ **Soul-searching in Israel over shooting of West Bank barrier protestor**

Swap Segments
• The Security Council on Thursday voted unanimously to extend the mandate of the UN mission in Georgia for four months ahead of next week's international talks on the fallout of the recent Caucasus conflict.
✔ **Security Council extends mandate of UN mission in Georgia**
✘ **UN mission in Georgia Security Council extends mandate of**

Table 5.5: Results on the Gigaword test set evaluated by ROUGE [42].[2]

System	R-1	R-2	R-L
lvt2k-1sent [28]	32.67	15.59	30.64
SEASS [60]	36.15	17.54	33.63
DRGD [84]	36.27	17.57	33.62
Pointer-Gen [29]	34.19	16.92	31.81
R3Sum [142]	37.04	19.03	34.46
EntailGen [124]	35.98	17.76	33.63
BiSET [143]	38.45	19.53	36.04
MASS [36]	38.73	19.71	35.96
UniLM [35]	38.90	20.05	36.00
PEGASUS [38]	39.12	19.86	36.24
Ours (Average)	35.51	16.33	32.75
Ours (Best Quality)	36.71	17.27	33.63
Ours (Best Summary Length)	**39.27**	**20.40**	**36.76**

Table 5.6: Results on Gigaword and Newsroom datasets where the generator produces summaries of varying lengths.

	Summary Length (L)											Best	Best
Gigaword	7	8	9	10	11	12	13	14	15	16	**Avg.**	**Quality**	**Length**
R-1 F_1 (%)	32.01	35.42	37.05	37.95	**38.05**	37.79	37.27	36.66	35.75	35.13	36.31	36.71	**39.27**
R-2 F_1 (%)	13.47	15.68	17.39	18.31	**18.24**	18.22	17.85	17.19	16.63	16.00	16.90	17.27	**20.40**
R-L F_1 (%)	29.76	32.85	34.46	**35.31**	35.10	34.87	34.21	33.53	32.71	32.02	33.48	33.63	**36.76**
	Summary Length (L)											**Best**	**Best**
Newsroom	7	8	9	10	11	12	13	14	15	16	**Avg.**	**Quality**	**Length**
R-1 F_1 (%)	40.99	43.38	44.94	46.06	46.57	**46.77**	46.53	46.25	45.76	45.21	45.25	45.77	46.60
R-2 F_1 (%)	19.15	20.99	22.11	23.02	23.47	**23.59**	23.38	23.15	22.79	22.33	22.40	22.58	**23.85**
R-L F_1 (%)	38.24	40.34	41.56	42.36	**42.69**	42.68	42.31	41.88	41.29	40.63	41.40	41.48	**43.07**

Table 5.7: Example annotation interface. A human annotator is instructed to read over the summaries before seeing the source text to effectively recognize any hallucinated content that is not found in the source text. A native English speaker creates annotations for multiple instances, which are shared with all annotators to provide guidance.

Candidate Summary	**Contains the main idea?**	**Is true-to-original?**	**Is grammatical?**
(1) Izetbegovic blasts Karadzic	☐ Yes ☒ No	☒ Yes ☐ No	☒ Yes ☐ No
(2) Karadzic accused of swaying US Congress	☒ Yes ☐ No	☒ Yes ☐ No	☒ Yes ☐ No
(3) Karadzic seeks to sway US Congress	☒ Yes ☐ No	☒ Yes ☐ No	☒ Yes ☐ No
(4) Karadzic seeks to sway Congress	☒ Yes ☐ No	☒ Yes ☐ No	☒ Yes ☐ No
(5) Karadzic misleading US Congress	☐ Yes ☒ No	☐ Yes ☒ No	☐ Yes ☒ No
(6) Monday's international soccer scores	☐ Yes ☒ No	☐ Yes ☒ No	☐ Yes ☒ No
Source Text: Bosnian President Alija Izetbegovic on Monday accused Bosnian Serb leader Radovan Karadzic of seeking to sway the US Congress against approving US troops to help enforce peace in the former Yugoslavia.			

Table 5.8: Results of human assessment. BEST-QUALITY summaries have a higher likelihood of being admissible according to the criteria, suggesting the effectiveness of the method.

	Content	**Truthful**	**Grammatical**	**Overall**
Average	80.7	82.6	96.5	74.2
Best Length	82.8	86.0	**97.4**	77.8
Best Quality	**93.0**	**90.8**	97.0	**88.2**

CHAPTER 6
CONCLUSION

6.1 More Challenges and Future Works

Current summarization systems remain imperfect, and they are still suffering from hallucination and lack of flexibility. Moreover, they may miss the most important information. On the one hand, current abstractive methods tend to generate some ”Jack of all trades” that are commonly seen in the training data. These methods will miss important but rare content. This conflicts the goal of figuring out the key messages. On the other hand, retrieving those important fragments using extractive method is not easy either, especially in long documents. A potential way is to incorporate extractive and abstractive methods within an affordable length. For longer documents, we may also need to study how different genres of languages are structured and the ways of modeling them. This will help summarization task in some specific domains.

In addition, different people have different flavors of summary. Some people would like to read the summary without thorough understanding while others may require more details. Moreover, people may have their own perspectives and questions before reading the summary. All of the above makes it harder to evaluate a summary. It will get better after new evaluation metrics and new dataset that support diverse demands are purposed.

6.2 Future Expectation

In the next 3 to 5 years, there will be multiple customized applications for domain specific scenarios, e.g. sport games reporting, news feeding, meeting highlights. It will save human efforts in those scenarios.

In the next 6 to 10 years, summarization technologies will become an important component of cloud services provided to business users even for open-domain scenarios, just like speech recognition and translation services. Everyone can get access to it and can benefit from it.

6.3 Conclusion

In my five years of research on summarization, I find summarization is an important technology for information processing. It helps people in fast browsing the key messages they need to know and saves their time. Though, it is still a challenging task, I feel much more confident that with more advanced technologies being discovered, and they will become robust and flexible for use. More people will benefit from this technology.

LIST OF REFERENCES

[1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[2] H. P. Edmundson, "New methods in automatic extracting," *Journal of the ACM (JACM)*, vol. 16, no. 2, pp. 264–285, 1969.

[3] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[4] I. Mani and E. Bloedorn, "Multi-document summarization by graph search and matching," in *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, 1997, pp. 622–628.

[5] R. Barzilay and M. Elhadad, "Using lexical chains for text summarization," *Advances in automatic text summarization*, pp. 111–121, 1999.

[6] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of artificial intelligence research*, vol. 22, pp. 457–479, 2004.

[7] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, 1995, pp. 68–73.

[8] C.-Y. Lin, "Training a selection function for extraction," in *Proceedings of the eighth international conference on Information and knowledge management*, 1999, pp. 55–62.

[9] M. Osborne, "Using maximum entropy for sentence extraction," in *Proceedings of the ACL-02 Workshop on Automatic Summarization*, 2002, pp. 1–8.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[16] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 484–494.

[17] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: a recurrent neural network based sequence model for extractive summarization of documents," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 3075–3081.

[18] K. Arumae and F. Liu, "Guiding extractive summarization with question-answering rewards," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 2566–2577.

[19] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3721–3731.

[20] F. Liu and Y. Liu, "From extractive to abstractive meeting summaries: Can it be done by sentence compression?" in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, 2009, pp. 261–264.

[21] T. Cohn and M. Lapata, "Sentence compression beyond word deletion," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008, pp. 137–144.

[22] C. Li, F. Liu, F. Weng, and Y. Liu, "Document summarization via guided sentence compression," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 490–500.

[23] F. Liu and Y. Liu, "Towards abstractive speech summarization: Exploring unsupervised and supervised approaches for spoken utterance compression," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1469–1480, 2013.

[24] K. Ganesan, C. X. Zhai, and J. Han, "Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions," in *23rd International Conference on Computational Linguistics, Coling 2010*, 2010.

[25] F. Liu, J. Flanigan, S. Thomson, N. Sadeh, and N. A. Smith, "Toward abstractive summarization using semantic representations," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1077–1086.

[26] K. Liao, L. Lebanoff, and F. Liu, "Abstract meaning representation for multi-document summarization," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1178–1190.

[27] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 379–389. [Online]. Available: https://www.aclweb.org/anthology/D15-1044

[28] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 280–290.

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

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[32] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training."

[33] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[34] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan,

R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[35] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/c20bb2d9a50d5ac1f713f8b34d9aac5a-Paper.pdf

[36] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MASS: Masked sequence to sequence pre-training for language generation," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 5926–5936. [Online]. Available: http://proceedings.mlr.press/v97/song19d.html

[37] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.703

[38] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.

[39] W. Qi, Y. Yan, Y. Gong, D. Liu, N. Duan, J. Chen, R. Zhang, and M. Zhou, "Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 2401–2410.

[40] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020.

[41] A. Nenkova and R. Passonneau, "Evaluating content selection in summarization: The pyramid method," in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*. Boston, Massachusetts, USA: Association for Computational

Linguistics, May 2 - May 7 2004, pp. 145–152. [Online]. Available: https://www.aclweb.org/anthology/N04-1019

[42] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013

[43] A. Nenkova and K. McKeown, "Automatic summarization," *Foundations and Trends in Information Retrieval*, 2011.

[44] H. Jing and K. McKeown, "The decomposition of human-written summary sentences," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[45] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. [Online]. Available: https://arxiv.org/abs/1705.04304

[46] J. Tan, X. Wan, and J. Xiao, "Abstractive document summarization with a graph-based attentional neural model," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017. [Online]. Available: https://www.aclweb.org/anthology/P17-1108

[47] H. Daumé III and D. Marcu, "A noisy-channel model for document compression," in *Proc. of ACL*, 2002. [Online]. Available: https://www.aclweb.org/anthology/P02-1057

[48] T. Berg-Kirkpatrick, D. Gillick, and D. Klein, "Jointly learning to extract and compress," in *Proc. of ACL*, 2011. [Online]. Available: https://dl.acm.org/citation.cfm?id=2002534

[49] K. Thadani and K. McKeown, "Sentence compression with joint structural inference," in *Proceedings of CoNLL*, 2013.

[50] G. Durrett, T. Berg-Kirkpatrick, and D. Klein, "Learning-based single-document summarization with compression and anaphoricity constraints," in *Proc. of ACL*, 2016. [Online]. Available: https://arxiv.org/abs/1603.08887

[51] R. McDonald, "Discriminative sentence compression with soft syntactic evidence," in *Proceedings of EACL*, 2006.

[52] J. Clarke and M. Lapata, "Global inference for sentence compression: An integer linear programming approach," *JAIR*, 2008. [Online]. Available: https://www.aaai.org/Papers/JAIR/Vol31/JAIR-3112.pdf

[53] K. Filippova, E. Alfonseca, C. Colmenares, L. Kaiser, and O. Vinyals, "Sentence compression by deletion with lstms," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[54] D. Zajic, B. J. Dorr, J. Lin, and R. Schwartz, "Multi-candidate reduction: Sentence compression as a tool for document summarization tasks," *Information Processing and Management*, 2007. [Online]. Available: http://users.umiacs.umd.edu/~jimmylin/publications/Zajic_etal_IPM2007.pdf

[55] D. Galanis and I. Androutsopoulos, "An extractive supervised two-stage method for sentence compression," in *Proceedings of NAACL-HLT*, 2010.

[56] L. Wang, H. Raghavan, V. Castelli, R. Florian, and C. Cardie, "A sentence compression based framework to query-focused multi-document summarization," in *Proceedings of ACL*, 2013. [Online]. Available: https://arxiv.org/abs/1606.07548

[57] C. Li, Y. Liu, F. Liu, L. Zhao, and F. Weng, "Improving multi-documents summarization by sentence compression based on expanded constituent parse tree," in *Proceedings of EMNLP*, 2014.

[58] Y. Kikuchi, G. Neubig, R. Sasano, H. Takamura, and M. Okumura, "Controlling output length in neural encoder-decoders," in *Proceedings of EMNLP*, 2016.

[59] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, and H. Jiang, "Distraction-based neural nets for doc. summarization," in *IJCAI*, 2016. [Online]. Available: https://arxiv.org/abs/1610.08462

[60] Q. Zhou, N. Yang, F. Wei, and M. Zhou, "Selective encoding for abstractive sentence summarization," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017. [Online]. Available: https://arxiv.org/abs/1704.07073

[61] G. Carenini and J. C. K. Cheung, "Extractive vs. NLG-based abstractive summarization of evaluative text: The effect of corpus controversiality," in *Proceedings of the Fifth International Natural Language Generation Conference (INLG)*, 2008.

[62] S. Gerani, Y. Mehdad, G. Carenini, R. T. Ng, and B. Nejat, "Abstractive summarization of product reviews using discourse structure," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[63] G. D. Fabbrizio, A. J. Stent, and R. Gaizauskas, "A hybrid approach to multi-document summarization of opinions in reviews," *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, 2014.

[64] F. Liu, J. Flanigan, S. Thomson, N. Sadeh, and N. A. Smith, "Toward abstractive summarization using semantic representations," in *Proc. of NAACL*, 2015. [Online]. Available: https://www.aclweb.org/anthology/N15-1114

[65] S. Takase, J. Suzuki, N. Okazaki, T. Hirao, and M. Nagata, "Neural headline generation on abstract meaning representation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[66] D. Pighin, M. Cornolti, E. Alfonseca, and K. Filippova, "Modelling events through memory-based, open-ie patterns for abstractive summarization," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

[67] Z. Cao, F. Wei, W. Li, and S. Li, "Faithful to the original: Fact aware neural abstractive summarization," in *Proc. of AAAI*, 2018. [Online]. Available: https://arxiv.org/abs/1711.04434

[68] J. Li, D. Xiong, Z. Tu, M. Zhu, M. Zhang, and G. Zhou, "Modeling source syntax for neural machine translation," in *ACL*, 2017. [Online]. Available: https://arxiv.org/abs/1705.01020

[69] H. Chen, S. Huang, D. Chiang, and J. Chen, "Improved neural machine translation with a syntax-aware encoder and decoder," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[70] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the Conference Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[71] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[73] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

[74] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proceedings of ACL*, 2016.

[75] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proceedings of ACL*, 2016.

[76] W. Zeng, W. Luo, S. Fidler, and R. Urtasun, "Efficient summarization with read-again and copy mechanism," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[77] R. Parker, "English Gigaword fifth edition LDC2011T07," *Philadelphia: Linguistic Data Consortium*, 2011. [Online]. Available: https://catalog.ldc.upenn.edu/LDC2011T07

[78] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. of EMNLP*, 2014. [Online]. Available: https://nlp.stanford.edu/pubs/emnlp2014-depparser.pdf

[79] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[80] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

[81] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural nets," in *NAACL*, 2016. [Online]. Available: https://www.aclweb.org/anthology/N16-1012

[82] Y. Miao and P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression," in *EMNLP*, 2016. [Online]. Available: https://arxiv.org/abs/1609.07317

[83] R. Pasunuru, H. Guo, and M. Bansal, "Towards improving abstractive summarization via entailment generation," in *Proceedings of the Workshop on New Frontiers in Summarization*, 2017. [Online]. Available: https://www.aclweb.org/anthology/W17-4504

[84] P. Li, W. Lam, L. Bing, and Z. Wang, "Deep recurrent generative decoder for abstractive text summarization," in *EMNLP*, 2017. [Online]. Available: https://www.aclweb.org/anthology/D17-1222

[85] X. Zhang and M. Lapata, "Sentence simplification with deep reinforcement learning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[86] Y.-C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," in *Proc. of ACL*, 2018. [Online]. Available: https://arxiv.org/abs/1805.11080

[87] W. Kryscinski, R. Paulus, C. Xiong, and R. Socher, "Improving abstraction in text summarization," in *EMNLP*, 2018. [Online]. Available: https://arxiv.org/abs/1808.07913

[88] K. Song, L. Zhao, and F. Liu, "Structure-infused copy mechanisms for abstractive summarization," in *Proc. of COLING*, 2018. [Online]. Available: https://aclweb.org/anthology/C18-1146

[89] L. Lebanoff, J. Muchovej, F. Dernoncourt, D. S. Kim, S. Kim, W. Chang, and F. Liu, "Analyzing sentence fusion in abstractive summarization," in *Wksp. on New Frontiers in Summ.*, 2019.

[90] A. Celikyilmaz, A. Bosselut, X. He, and Y. Choi, "Deep communicating agents for abstractive summarization," in *NAACL*, 2018. [Online]. Available: https://arxiv.org/pdf/1803.10357.pdf

[91] L. Lebanoff, K. Song, F. Dernoncourt, D. S. Kim, S. Kim, W. Chang, and F. Liu, "Scoring sentence singletons and pairs for abstractive summarization," in *ACL*, 2019.

[92] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian, "A discourse-aware attention model for abstractive summarization of long documents," in *NAACL*, 2018. [Online]. Available: https://arxiv.org/abs/1804.05685

[93] L. Lebanoff, K. Song, and F. Liu, "Adapting the neural encoder-decoder framework from single to multi-document summarization," in *EMNLP*, 2018.

[94] S. Gehrmann, Y. Deng, and A. M. Rush, "Bottom-up abstractive summarization," in *Proc. of EMNLP*, 2018. [Online]. Available: https://arxiv.org/abs/1808.10792

[95] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization," in *Proc. of EMNLP*, 2018. [Online]. Available: https://arxiv.org/abs/1808.08745

[96] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1801.10198

[97] K. Song, B. Wang, Z. Feng, L. Ren, and F. Liu, "Controlling the amount of verbatim copying in abstractive summarization," in *AAAI*, 2020.

[98] D. Marcu, "From discourse structures to text summaries," in *Intelligent Scalable Text Summarization*, 1997. [Online]. Available: https://www.aclweb.org/anthology/W97-0713

[99] ——, "Improving summarization through rhetorical parsing tuning," in *Sixth Workshop on Very Large Corpora*, 1998. [Online]. Available: https://www.aclweb.org/anthology/W98-1124

[100] A. F. T. Martins and N. A. Smith, "Summarization with a joint model for sentence extraction and compression," in *Proc. of the ACL Workshop on ILP for Natural Language Processing*, 2009. [Online]. Available: https://www.aclweb.org/anthology/W09-1801

[101] K. Filippova, "Multi-sentence compression: Finding shortest paths in word graphs," in *Proc. of COLING*, 2010. [Online]. Available: https://www.aclweb.org/anthology/C10-1037

[102] J. Christensen, Mausam, S. Soderland, and O. Etzioni, "Towards coherent multi-document summarization," in *NAACL*, 2013. [Online]. Available: https://aclweb.org/anthology/N13-1136

[103] Y. Yoshida, J. Suzuki, T. Hirao, and M. Nagata, "Dependency-based discourse parser for single-document summarization," in *Proc. of EMNLP*, 2014. [Online]. Available: https://pdfs.semanticscholar.org/3eb3/43d0b8160fb406da59855118a49c53e1136e.pdf

[104] J. J. Li, K. Thadani, and A. Stent, "The role of discourse units in near-extractive summarization," in *Proc. of SIGDIAL*, 2016.

[105] K. Liao, L. Lebanoff, and F. Liu, "Abstract meaning representation for multi-document summarization," in *COLING*, 2018.

[106] H. Hardy and A. Vlachos, "Guided neural language generation for abstractive summarization using abstract meaning representation," in *Proc. of EMNLP*, 2018. [Online]. Available: https://arxiv.org/abs/1808.09160

[107] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," in *Proc. of ACL*, 2015. [Online]. Available: https://arxiv.org/abs/1505.08075

[108] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional LSTM feature representations," *Trans. of the Association for Computational Linguistics*, 2016. [Online]. Available: https://aclweb.org/anthology/Q16-1023

[109] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," in *Proc. of ICLR*, 2017. [Online]. Available: https://arxiv.org/abs/1611.01734

[110] X. Ma, Z. Hu, J. Liu, N. Peng, G. Neubig, and E. Hovy, "Stack-pointer networks for dependency parsing," in *ACL*, 2018. [Online]. Available: https://arxiv.org/abs/1805.01087

[111] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, "Recurrent neural network grammars," in *Proc. of NAACL*, 2016. [Online]. Available: https://arxiv.org/abs/1602.07776

[112] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith, "What do recurrent neural network grammars learn about syntax?" in *Proc. of EACL*, 2017. [Online]. Available: https://aclweb.org/anthology/E17-1117

[113] A. Eriguchi, Y. Tsuruoka, and K. Cho, "Learning to parse and translate improves neural machine translation," in *ACL*, 2017.

[114] S. Wu, D. Zhang, N. Yang, M. Li, and M. Zhou, "Sequence-to-dependency neural machine translation," in *ACL*, 2017. [Online]. Available: https://www.aclweb.org/anthology/P17-1065

[115] X. Wang, H. Pham, P. Yin, and G. Neubig, "A tree-based decoder for neural machine translation," in *Proc. of EMNLP*, 2018. [Online]. Available: https://arxiv.org/abs/1808.09374

[116] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract meaning representation for sembanking," in *Proceedings of Linguistic Annotation Workshop*, 2013. [Online]. Available: https://www.aclweb.org/anthology/W13-2322

[117] I. Konstas, S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer, "Neural AMR: Sequence-to-sequence models for parsing and generation," in *Proc. of ACL*, 2017. [Online]. Available: https://arxiv.org/abs/1704.08381

[118] G. Neubig and et al., "DyNet: The dynamic neural network toolkit," *arXiv preprint arXiv:1701.03980*, 2017. [Online]. Available: https://arxiv.org/abs/1701.03980

[119] M. Grusky, M. Naaman, and Y. Artzi, "NEWSROOM: A dataset of 1.3 million summaries with diverse extractive strategies," in *Proc. of NAACL-HLT*, 2018. [Online]. Available: https://arxiv.org/abs/1804.11283

[120] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Proc. of NIPS*, 2015. [Online]. Available: https://arxiv.org/abs/1506.03340

[121] K. Thadani and K. McKeown, "Supervised sentence fusion with single-stage inference," in *Proc. of IJCNLP*, 2013. [Online]. Available: https://www.aclweb.org/anthology/I13-1198

[122] S. Narayan, C. Gardent, S. B. Cohen, and A. Shimorina, "Split and rephrase," in *Proc. of EMNLP*, 2017. [Online]. Available: https://arxiv.org/pdf/1707.06971.pdf

[123] R. Pasunuru and M. Bansal, "Multi-reward reinforced summarization with saliency and entailment," in *Proc. of NAACL*, 2018. [Online]. Available: https://arxiv.org/abs/1804.06451

[124] H. Guo, R. Pasunuru, and M. Bansal, "Soft, layer-specific multi-task summarization with entailment and question generation," in *Proc. of ACL*, 2018. [Online]. Available: https://arxiv.org/abs/1805.11004

[125] R. E. de Castilho, G. Dore, T. Margoni, P. Labropoulou, and I. Gurevych, "A legal perspective on training models for natural language processing," in *Proc. of LREC*, 2019.

[126] E. Reiter, "A structured review of the validity of BLEU," *Computational Linguistics*, vol. 44, no. 3, pp. 393–401, 2018.

[127] N. Weber, L. Shekhar, N. Balasubramanian, and K. Cho, "Controlling decoding for more abstractive summaries with copy-based networks," *https://arxiv.org/abs/1803.07038*, 2018.

[128] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of NIPS*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[129] U. Khandelwal, K. Clark, D. Jurafsky, and L. Kaiser, "Sample efficient text summarization using a single pre-trained transformer," *https://arxiv.org/abs/1905.08836*, 2019.

[130] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *https://arxiv.org/abs/1904.09675*, 2019.

[131] P. Over and J. Yen, "An introduction to DUC-2004," *NIST*, 2004.

[132] J. Carletta and et al., "The AMI meeting corpus," in *MLMI*, 2005.

[133] V. Qazvinian, D. R. Radev, S. M. Mohammad, B. Dorr, D. Zajic, M. Whidby, and T. Moon, "Generating extractive summaries of scientific paradigms," *JAIR*, 2013.

[134] J. Ouyang, S. Chang, and K. McKeown, "Crowd-sourced iterative annotation for narrative summarization corpora," in *EACL*, 2017.

[135] J.-P. Ng and V. Abrecht, "Better summarization evaluation with word embeddings for ROUGE," in *EMNLP*, 2015.

[136] R. Barzilay and K. R. McKeown, "Sentence fusion for multidocument news summarization," *Computational Linguistics*, vol. 31, no. 3, 2005.

[137] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[138] R. Knowles and P. Koehn, "Context and copying in neural machine translation," in *Proc. of EMNLP*, 2018.

[139] A. Fan, D. Grangier, and M. Auli, "Controllable abstractive summarization," in *the 2nd Workshop on NMT and Generation*, 2018.

[140] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://www.aclweb.org/anthology/P16-1162

[141] Y. Yang, L. Huang, and M. Ma, "Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation," in *EMNLP*, 2018.

[142] Z. Cao, W. Li, S. Li, and F. Wei, "Retrieve, rerank and rewrite: Soft template based neural summarization," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[143] K. Wang, X. Quan, and R. Wang, "BiSET: bi-directional selective encoding with template for abstractive summarization," in *Proc. of ACL*, 2019.

[144] Y. Liu and M. Lapata, "Hierarchical transformers for multi-document summarization," in *ACL*, 2019.

[145] A. Fabbri, I. Li, T. She, S. Li, and D. Radev, "Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1074–1084. [Online]. Available: https://www.aclweb.org/anthology/P19-1102

[146] A. Bražinskas, M. Lapata, and I. Titov, "Few-shot learning for opinion summarization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 4119–4135. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.337

[147] W. Kryscinski, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 9332–9346. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.750

[148] A. Belz and E. Reiter, "Comparing automatic and human evaluation of NLG systems," in *11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy: Association for Computational Linguistics, Apr. 2006. [Online]. Available: https://www.aclweb.org/anthology/E06-1040

[149] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 1906–1919. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.173

[150] T. Falke, L. F. R. Ribeiro, P. A. Utama, I. Dagan, and I. Gurevych, "Ranking generated summaries by correctness: An interesting but challenging application for natural language inference," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2214–2220. [Online]. Available: https://www.aclweb.org/anthology/P19-1213

[151] E. Durmus, H. He, and M. Diab, "FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5055–5070. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.454

[152] A. Louis and A. Nenkova, "Automatically assessing machine summary content without a gold standard," *Computational Linguistics*, vol. 39, no. 2, pp. 267–300, 2013. [Online]. Available: https://www.aclweb.org/anthology/J13-2002

[153] W. Kryscinski, N. S. Keskar, B. McCann, C. Xiong, and R. Socher, "Neural text summarization: A critical evaluation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 540–551. [Online]. Available: https://www.aclweb.org/anthology/D19-1051

[154] A. Wang, K. Cho, and M. Lewis, "Asking and answering questions to evaluate the factual consistency of summaries," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5008–5020. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.450

[155] Y. Dong, S. Wang, Z. Gan, Y. Cheng, J. C. K. Cheung, and J. Liu, "Multi-fact correction in abstractive text summarization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 9320–9331. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.749

[156] Y. Zhang, D. Merck, E. Tsai, C. D. Manning, and C. Langlotz, "Optimizing the factual correctness of a summary: A study of summarizing radiology reports," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5108–5120. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.458

[157] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Controllable text generation," *CoRR*, vol. abs/1703.00955, 2017. [Online]. Available: http://arxiv.org/abs/1703.00955

[158] N. S. Keskar, B. McCann, L. Varshney, C. Xiong, and R. Socher, "CTRL - A Conditional Transformer Language Model for Controllable Generation," *arXiv preprint arXiv:1909.05858*, 2019.

[159] T. Makino, T. Iwakura, H. Takamura, and M. Okumura, "Global optimization under length constraint for neural text summarization," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1039–1048. [Online]. Available: https://www.aclweb.org/anthology/P19-1099

[160] C. Napoles, B. Van Durme, and C. Callison-Burch, "Evaluating sentence compression: Pitfalls and suggested remedies," in *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Portland, Oregon: Association for Computational Linguistics, Jun. 2011, pp. 91–97. [Online]. Available: https://www.aclweb.org/anthology/W11-1611

[161] O. Shapira, D. Gabay, H. Ronen, J. Bar-Ilan, Y. Amsterdamer, A. Nenkova, and I. Dagan, "Evaluating multiple system summary lengths: A case study," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels,

Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 774–778. [Online]. Available: https://www.aclweb.org/anthology/D18-1087

[162] J. Gu, C. Wang, and J. Zhao, "Levenshtein transformer," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 11 181–11 191. [Online]. Available: http://papers.nips.cc/paper/9297-levenshtein-transformer.pdf

[163] E. Malmi, S. Krause, S. Rothe, D. Mirylenka, and A. Severyn, "Encode, tag, realize: High-precision text editing," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5054–5065. [Online]. Available: https://www.aclweb.org/anthology/D19-1510

[164] Y. Dong, Z. Li, M. Rezagholizadeh, and J. C. K. Cheung, "EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3393–3402. [Online]. Available: https://www.aclweb.org/anthology/P19-1331

[165] G. M. Correia and A. F. T. Martins, "A simple and effective approach to automatic post-editing with transfer learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3050–3056. [Online]. Available: https://www.aclweb.org/anthology/P19-1292

[166] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B1l8BtlCb

[167] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6112–6121. [Online]. Available: https://www.aclweb.org/anthology/D19-1633

[168] D. Ippolito, R. Kriz, J. Sedoc, M. Kustikova, and C. Callison-Burch, "Comparison of diverse decoding methods from conditional language models," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3752–3762. [Online]. Available: https://www.aclweb.org/anthology/P19-1365

[169] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 5753–5763. [Online]. Available: http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

[170] C. Meister, R. Cotterell, and T. Vieira, "If beam search is the answer, what was the question?" in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2173–2185. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.170

[171] S. Sun, O. Shapira, I. Dagan, and A. Nenkova, "How to compare summarizers without target length? pitfalls, solutions and re-examination of the neural summarization literature," in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 21–29. [Online]. Available: https://www.aclweb.org/anthology/W19-2303

[172] L. Huang, K. Zhao, and M. Ma, "When to finish? optimal beam search for neural text generation (modulo beam size)," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2134–2139. [Online]. Available: https://www.aclweb.org/anthology/D17-1227

[173] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[174] A. R. Fabbri, W. Kryściński, B. McCann, C. Xiong, R. Socher, and D. Radev, "Summeval: Re-evaluating summarization evaluation," *arXiv preprint arXiv:2007.12626*, 2020.

[175] Y. Liao, X. Jiang, and Q. Liu, "Probabilistically masked language model capable of autoregressive generation in arbitrary word order," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 263–274. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.24

[176] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? an analysis of BERT's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 276–286. [Online]. Available: https://www.aclweb.org/anthology/W19-4828

[177] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Transactions of the*

Association for Computational Linguistics, vol. 8, pp. 64–77, 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.tacl-1.5