Joint Parsing and Generation for Abstractive Summarization

Kaiqiang Song,[1] Logan Lebanoff,[1] Qipeng Guo[2]
Xipeng Qiu,[2] Xiangyang Xue,[2] Chen Li,[3] Dong Yu,[3] Fei Liu[1]
[1]Computer Science Department, University of Central Florida
[2]School of Computer Science, Fudan University, [3]Tencent AI Lab, Bellevue, WA
{kqsong, loganlebanoff}@knights.ucf.edu, {qpguo16, xpqiu, xyxue}@fudan.edu.cn,
{ailabchenli, dyu}@tencent.com, feiliu@cs.ucf.edu

Abstract

Sentences produced by abstractive summarization systems can be ungrammatical and fail to preserve the original meanings, despite being locally fluent. In this paper we propose to remedy this problem by jointly generating a sentence and its syntactic dependency parse while performing abstraction. If generating a word can introduce an erroneous relation to the summary, the behavior must be discouraged. The proposed method thus holds promise for producing grammatical sentences and encouraging the summary to stay true-to-original. Our contributions of this work are twofold. First, we present a novel neural architecture for abstractive summarization that combines a sequential decoder with a tree-based decoder in a synchronized manner to generate a summary sentence and its syntactic parse. Secondly, we describe a novel human evaluation protocol to assess if, and to what extent, a summary remains true to its original meanings. We evaluate our method on a number of summarization datasets and demonstrate competitive results against strong baselines.

Introduction

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 (See, Liu, and Manning 2017; Chen and Bansal 2018; Kryscinski et al. 2018). 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 (Song, Zhao, and Liu 2018; Lebanoff et al. 2019a). In Table 1, we show example abstractive summaries produced by neural abstractive summarizers. The first summary has failed to conform 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.

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***

Table 1: Example summaries generated by neural abstractive summarizers. They are manually re-cased for readability.

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]

Related Work

Recent years have seen increasing interest in summarization using encoder-decoder models (Rush, Chopra, and Weston 2015; Nallapati et al. 2016; See, Liu, and Manning 2017; Celikyilmaz et al. 2018; Lebanoff et al. 2019b). 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 (Chen et al. 2016; Tan, Wan, and Xiao 2017; Cohan et al. 2018; Lebanoff, Song, and Liu 2018; Gehrmann, Deng, and Rush 2018), convolutional networks (Chopra, Auli, and Rush 2016; Narayan, Cohen, and Lapata 2018), or Transformer (Devlin et al. 2018; Liu et al. 2018; Song et al. 2020). 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 (1997; 1998) 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 (Daumé III and Marcu 2002; Clarke and Lapata 2008; Martins and Smith 2009; Filippova 2010; Berg-Kirkpatrick, Gillick, and Klein 2011; Wang et al. 2013; Durrett, Berg-Kirkpatrick, and Klein 2016), rhetorical structure (Christensen et al. 2013; Yoshida et al. 2014; Li, Thadani, and Stent 2016), and abstract meaning representation (Liu et al. 2015; Liao, Lebanoff, and Liu 2018; Hardy and Vlachos 2018) to generate compressive and abstractive summaries. In this paper we emphasize that *target-side* syntactic analysis is especially important to ensure the well-formedness of abstractive summaries, because generating summary words and predicting relations between words are interleaved operations.

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

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 (Dyer et al. 2015; Kiperwasser and Goldberg 2016; Dozat and Manning 2017; Ma et al. 2018). Our method, inspired by the recurrent neural network grammar (RNNG; Dyer et al., 2016) that describes a generative probabilistic model for parsing and language modeling (Kuncoro et al. 2017), offers a way to perform summary generation and parsing in a synchronized manner. Incorporating syntax is found to improve translation (Li et al. 2017a; Eriguchi, Tsuruoka, and Cho 2017; Wu et al. 2017; Wang et al. 2018). 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.

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 $\mathbf{y}^{\mathcal{T}}$. We expect a summary to contain a single sentence, as our focus is to improve sentence grammaticality.[2] We use dependency 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 $\mathbf{y}^{\mathcal{T}}$ consists of interleaved GEN(w) and REDUCE-L/R operations that incrementally build a dependency parse tree. Table 2 shows an example. The second column contains $\mathbf{y}^{\mathcal{T}}$ 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 $\mathbf{y}^{\mathcal{T}}$ by retrieving all GEN operations.

We aim to predict the target sequence $\mathbf{y}^{\mathcal{T}}$ conditioned on the source $\mathbf{x}$. The process proceeds incrementally. As illustrated in Eq. (1), $P(\mathbf{y}^{\mathcal{T}}|\mathbf{x})$ is factorized over time steps.

[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 (Chen and Bansal 2018; Gehrmann, Deng, and Rush 2018).

[3]Our method is also general enough to allow other syntactic/semantic formalisms such as the constituency grammar or abstract meaning representation (Banarescu et al. 2013; Konstas et al. 2017) to be exploited in future work.

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

Table 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 $\mathbf{y}^{\mathcal{T}}$ and the third column contains partial parse trees stored in a *stack*.

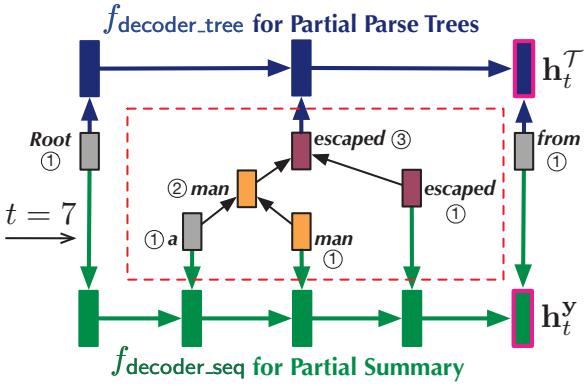

Figure 1: $f_{\text{decoder\_tree}}$ (top) consumes the *partial tree* representations of time t one by one to build hidden representation $\mathbf{h}_t^{\mathcal{T}}$; $f_{\text{decoder\_seq}}$ (bottom) consumes the embeddings of summary words to build *partial summary* representation $\mathbf{h}_t^{\mathbf{y}}$.

$P(\mathbf{y}_t^{\mathcal{T}} = o|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x})$ denotes the probability of a parsing operation, where $o \in \{\text{REDUCE-L}, \text{REDUCE-R}, \text{GEN}\}$ and GEN is unlexicalized. $P(\mathbf{y}_t^{\mathcal{T}} = w|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x})$ 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.

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

At training time, the ground-truth sequence $\hat{\mathbf{y}}^{\mathcal{T}}$ is available, $P(\hat{\mathbf{y}}_t^{\mathcal{T}} = w|\hat{\mathbf{y}}_{<t}^{\mathcal{T}}, \mathbf{x})$ 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. (2)). During inference, we calculate $P(\mathbf{y}_t^{\mathcal{T}}|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x})$ as a joint distribution over parsing and summarization operations, where $\mathbf{y}_t^{\mathcal{T}} \in \{\text{REDUCE-L}, \text{REDUCE-R}, \text{GEN}(w)\}$.

$$\log P(\hat{\mathbf{y}}^{\mathcal{T}}|\mathbf{x}) = \Big[\sum_t \log P(\hat{\mathbf{y}}_t^{\mathcal{T}} = o|\hat{\mathbf{y}}_{<t}^{\mathcal{T}}, \mathbf{x}) \Big] + \Big[\sum_{t:o_t=\text{GEN}} \log P(\hat{\mathbf{y}}_t^{\mathcal{T}} = w|\hat{\mathbf{y}}_{<t}^{\mathcal{T}}, \mathbf{x}) \Big] \quad (2)$$

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 (Dyer et al. 2015; 2016). Our stack maintains a set of partial trees at any time t; they are shown in the t-th row of Table 2. For each partial tree, we build a vector representation for it by recursively applying a syntactic composition function (Eq. (3)). The representation is built from bottom up, shown in the dotted circle of Figure 1. A left arc (REDUCE-L) pops two elements from the stack. It then applies the composition function to create a new representation $\mathbf{g}_{\text{new\_head}}$ 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. (2017) report that the composition function learns to compute a tree representation by preserving the semantics of the head word, which fits our task.

$$\mathbf{g}_{\text{new\_head}} = \tanh(\mathbf{W}^g[\mathbf{g}_{\text{head}}||\mathbf{g}_{\text{dependent}}] + \mathbf{b}^g) \quad (3)$$

We introduce an LSTM, denoted by $f_{\text{decoder\_tree}}$, to consume the *partial tree* representations of time t one by one to build the hidden representation $\mathbf{h}_t^{\mathcal{T}}$. An illustration is presented in Figure 1. E.g., when t=7, the stack contains 3 partial trees and we build a vector representation for each. $f_{\text{decoder\_tree}}$ is unrolled 3 steps and its last hidden state is $\mathbf{h}_t^{\mathcal{T}}$. Similarly, we build the *partial summary* representation $\mathbf{h}_t^{\mathbf{y}}$ using an LSTM denoted by $f_{\text{decoder\_seq}}$, which consumes the embeddings of summary words. For example, when t=7, there are 5 words in the partial summary. $f_{\text{decoder\_seq}}$ 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 $\mathbf{h}_{t-1}^{\mathbf{y}}$.

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.

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

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. (4)). We concatenate the partial tree and partial summary representations $[\mathbf{h}_t^{\mathcal{T}}||\mathbf{h}_t^{\mathbf{y}}]$ 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. (5)).

$$S_{t,i} = \mathbf{w}^\top \tanh(\mathbf{W}^d[\mathbf{h}_t^{\mathcal{T}}||\mathbf{h}_t^{\mathbf{y}}] + \mathbf{W}^e\mathbf{h}_i^{\mathbf{x}}) \quad (4)$$

$$\mathbf{c}_t^{\mathbf{x}} = \text{softmax}(\mathbf{S}_t)\mathbf{h}^{\mathbf{x}} \quad (5)$$

Prediction We predict summary words $P(\mathbf{y}_t^{\mathcal{T}} = w|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x})$ and parsing operations $P(\mathbf{y}_t^{\mathcal{T}} = o|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x})$ 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 2. We thus use an LSTM to encode the sequence of past operations and its last hidden state is denoted by $\mathbf{h}_t^{\mathcal{O}}$. A parsing operation is predicted based on $[\mathbf{h}_t^{\mathcal{T}}||\mathbf{h}_t^{\mathcal{O}}||\mathbf{c}_t^{\mathbf{x}}]$, and we apply the softmax to obtain a distribution over parsing operations (Eq. (7)).

$$\widetilde{\mathbf{h}}_t^{\mathcal{T}} = \tanh(\mathbf{W}^a[\mathbf{h}_t^{\mathcal{T}}||\mathbf{h}_t^{\mathcal{O}}||\mathbf{c}_t^{\mathbf{x}}] + \mathbf{b}^a) \quad (6)$$

$$P(\mathbf{y}_t^{\mathcal{T}} = o|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x}) = \text{softmax}(\mathbf{W}^o\widetilde{\mathbf{h}}_t^{\mathcal{T}}) \quad (7)$$

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. (2017), where $\lambda = \sigma(\mathbf{W}^z[\mathbf{h}_t^{\mathbf{y}}||\mathbf{h}_t^{\mathcal{T}}||\mathbf{c}_t^{\mathbf{x}}]) + b^z)$ is the likelihood of generating a summary word from the vocabulary. The generation probability is defined in Eqs. (8-9). 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, $P(\mathbf{y}_t^{\mathcal{T}} = w|\cdot)$ is a weighted sum of the generation and copy probabilities.

$$\widetilde{\mathbf{h}}_t^{\mathbf{y}} = \tanh(\mathbf{W}^c[\mathbf{h}_t^{\mathbf{y}}||\mathbf{h}_t^{\mathcal{T}}||\mathbf{c}_t^{\mathbf{x}}] + \mathbf{b}^c) \quad (8)$$

$$\widetilde{P}(\mathbf{y}_t^{\mathcal{T}} = w|\mathbf{y}_{<t}^{\mathcal{T}}, \mathbf{x}) = \text{softmax}(\mathbf{W}^w\widetilde{\mathbf{h}}_t^{\mathbf{y}}) \quad (9)$$

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

Acceleration Obtaining partial tree representations ($\mathbf{h}_t^{\mathcal{T}}$) can be computationally expensive, because $\mathbf{h}_t^{\mathcal{T}}$ 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 (Neubig and et al. 2017) 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 ($\mathbf{h}_t^{\mathcal{T}}$) are computed in parallel. For example, in Figure 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.

	$\lvert\mathbf{y}\rvert$	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

Table 3: Statistics of our datasets. $\lvert\mathbf{y}\rvert$ is number of words.

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.

Data and Hyperparameters

We conduct experiments on a variety of datasets to gauge the effectiveness of our proposed method. We experiment with GIGAWORD (Parker 2011) and NEWSROOM (Grusky, Naaman, and Artzi 2018). 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. (2015) 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 (Hermann et al. 2015) has been extensively studied. We use the version provided by See et al. (2017) 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 (Lin 2004). 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.

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 (Thadani and McKeown 2013). In this work we present a novel use of a newly released dataset—WebSplit (Narayan et al. 2017). 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 (§). In Table 3, we provide statistics of all datasets used in this study.

System	Gigaword Test Set 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 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.

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 (Chen and Manning 2014) to obtain parse trees for reference summaries. During training, we use a batch size of 64 and Adam (Kingma and Ba 2015) 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 (Tan, Wan, and Xiao 2017) to generate summary sequences. K=10 is the beam size.

Experimental Results

Summarization We present summarization results on all datasets. Evaluation is performed using the automatic metric of ROUGE (Lin 2004), 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 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 $\mathbf{h}_t^{\mathcal{T}}$ and $\mathbf{h}_t^{\mathcal{O}}$ in all equations. All other components remain the same. As shown in Table 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+* (Rush, Chopra, and Weston 2015) are the first work using an encoder-decoder architecture for summarization;
- *Luong-NMT* (Chopra, Auli, and Rush 2016) re-implements the attentive encoder-decoder of Luong et al. (2015);
- *RAS-LSTM* and *RAS-Elman* (Chopra, Auli, and Rush 2016) describe a convolutional attentive encoder that ensures the decoder focuses on appropriate words at each step of generation;
- *ASC+FSC1* (Miao and Blunsom 2016) presents a generative auto-encoding sentence compression model jointly trained on labelled/unlabelled data;
- *lvt2k-1sent* and *lvt5k-1sent* (Nallapati et al. 2016) address issues in the encoder-decoder model, including modeling keywords, capturing sentence-to-word structure, and handling rare words;
- *Multi-Task w/ Entailment* (Pasunuru and Bansal 2018) combines entailment with summarization in a multi-task setting;
- *DRGD* (Li et al. 2017b) describes a deep recurrent generative decoder learning latent structure of summary sequences via variational inference;
- *Struct+2Way+Word* (Song, Zhao, and Liu 2018) describes a structure infused copy mechanism for sentence summarization;
- *EntailGen+QuesGen* (Guo, Pasunuru, and Bansal 2018) is a multi-task architecture to perform summarization with question generation and entailment generation in one framework.

In Table 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 5 should not be directly compared with other published results. Instead, we train the pointer-generator networks with coverage mechanism (PointerGen; See et al. 2017), 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 2, generating the word *escaped*

	System	ROUGE-1 P	ROUGE-1 R	ROUGE-1 F	ROUGE-2 P	ROUGE-2 R	ROUGE-2 F	ROUGE-L P	ROUGE-L R	ROUGE-L F
NEWSROOM	PointerGen (See, Liu, and Manning 2017)	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 (See, Liu, and Manning 2017)	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 (See, Liu, and Manning 2017)	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 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.

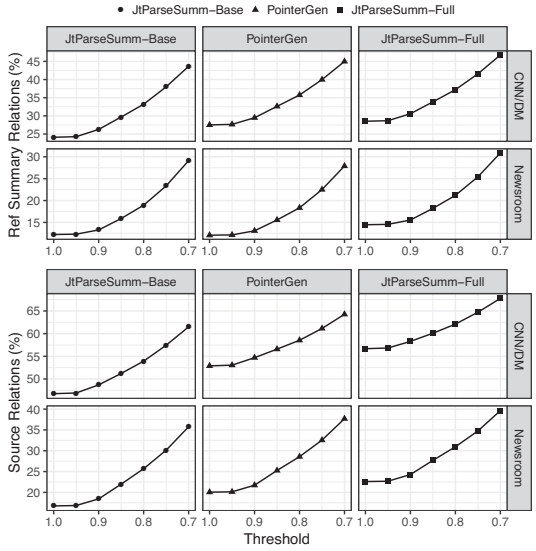

Figure 2: F-scores of systems on preserving relations of reference summaries (*top*) and source texts (*bottom*). We vary the threshold from 1.0 (strict match) to 0.7 in the x-axis to allow for strict and lenient matching of dependency relations.

leads a dependency relation *man*←*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]

In Figure 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 (Chen and Manning 2014). 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 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}←w_{1B} and w_{2A}←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 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 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 ques-

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

	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 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.

tions 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 (Narayan et al. 2017). 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 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 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 meanings (¬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. (2018) and See et al. (2017). 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.

[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 ($p < 0.01$).

	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**

Table 7: Human assessment of grammaticality and semantic accuracy of various summaries. Our GenParse-FULL achieves the best results on both aspects among all systems.

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.

Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions. This research was supported in part by the National Science Foundation grant IIS-1909603.

References

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract meaning representation for sembanking. In *Proceedings of Linguistic Annotation Workshop*.

Berg-Kirkpatrick, T.; Gillick, D.; and Klein, D. 2011. Jointly learning to extract and compress. In *Proc. of ACL*.

Cao, Z.; Wei, F.; Li, W.; and Li, S. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proc. of AAAI*.

Celikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep communicating agents for abstractive summarization. In *NAACL*.

Chen, Y.-C., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proc. of ACL*.

Chen, D., and Manning, C. D. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Chen, Q.; Zhu, X.; Ling, Z.-H.; Wei, S.; and Jiang, H. 2016. Distraction-based neural nets for doc. summarization. In *IJCAI*.

Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural nets. In *NAACL*.

Christensen, J.; Mausam; Soderland, S.; and Etzioni, O. 2013. Towards coherent multi-document summarization. In *NAACL*.

Clarke, J., and Lapata, M. 2008. Global inference for sentence compression: An integer linear programming approach. *JAIR*.

Cohan, A.; Dernoncourt, F.; Kim, D. S.; Bui, T.; Kim, S.; Chang, W.; and Goharian, N. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *NAACL*.

Daumé III, H., and Marcu, D. 2002. A noisy-channel model for document compression. In *Proc. of ACL*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Dozat, T., and Manning, C. D. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Durrett, G.; Berg-Kirkpatrick, T.; and Klein, D. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proc. of ACL*.

Dyer, C.; Ballesteros, M.; Ling, W.; Matthews, A.; and Smith, N. A. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Dyer, C.; Kuncoro, A.; Ballesteros, M.; and Smith, N. A. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Eriguchi, A.; Tsuruoka, Y.; and Cho, K. 2017. Learning to parse and translate improves neural machine translation. In *ACL*.

Filippova, K. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proc. of COLING*.

Gehrmann, S.; Deng, Y.; and Rush, A. M. 2018. Bottom-up abstractive summarization. In *Proc. of EMNLP*.

Grusky, M.; Naaman, M.; and Artzi, Y. 2018. NEWSROOM: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proc. of NAACL-HLT*.

Guo, H.; Pasunuru, R.; and Bansal, M. 2018. Soft, layer-specific multi-task summarization with entailment and question generation. In *Proc. of ACL*.

Hardy, H., and Vlachos, A. 2018. Guided neural language generation for abstractive summarization using abstract meaning representation. In *Proc. of EMNLP*.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Proc. of NIPS*.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Kiperwasser, E., and Goldberg, Y. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Trans. of the Association for Computational Linguistics*.

Konstas, I.; Iyer, S.; Yatskar, M.; Choi, Y.; and Zettlemoyer, L. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proc. of ACL*.

Kryscinski, W.; Paulus, R.; Xiong, C.; and Socher, R. 2018. Improving abstraction in text summarization. In *EMNLP*.

Kuncoro, A.; Ballesteros, M.; Kong, L.; Dyer, C.; Neubig, G.; and Smith, N. A. 2017. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*.

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

Lebanoff, L.; Song, K.; Dernoncourt, F.; Kim, D. S.; Kim, S.; Chang, W.; and Liu, F. 2019b. Scoring sentence singletons and pairs for abstractive summarization. In *ACL*.

Lebanoff, L.; Song, K.; and Liu, F. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *EMNLP*.

Li, J.; Xiong, D.; Tu, Z.; Zhu, M.; Zhang, M.; and Zhou, G. 2017a. Modeling source syntax for neural machine translation. In *ACL*.

Li, P.; Lam, W.; Bing, L.; and Wang, Z. 2017b. Deep recurrent generative decoder for abstractive text summarization. In *EMNLP*.

Li, J. J.; Thadani, K.; and Stent, A. 2016. The role of discourse units in near-extractive summarization. In *Proc. of SIGDIAL*.

Liao, K.; Lebanoff, L.; and Liu, F. 2018. Abstract meaning representation for multi-document summarization. In *COLING*.

Lin, C.-Y. 2004. ROUGE: a package for automatic evaluation of summaries. In *ACL Wksp. on Text Summarization Branches Out*.

Liu, F.; Flanigan, J.; Thomson, S.; Sadeh, N.; and Smith, N. A. 2015. Toward abstractive summarization using semantic representations. In *Proc. of NAACL*.

Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R.; Kaiser, L.; and Shazeer, N. 2018. Generating wikipedia by summarizing long sequences. In *Proc. of ICLR*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Ma, X.; Hu, Z.; Liu, J.; Peng, N.; Neubig, G.; and Hovy, E. 2018. Stack-pointer networks for dependency parsing. In *ACL*.

Marcu, D. 1997. From discourse structures to text summaries. In *Intelligent Scalable Text Summarization*.

Marcu, D. 1998. Improving summarization through rhetorical parsing tuning. In *Sixth Workshop on Very Large Corpora*.

Martins, A. F. T., and Smith, N. A. 2009. Summarization with a joint model for sentence extraction and compression. In *Proc. of the ACL Workshop on ILP for Natural Language Processing*.

Miao, Y., and Blunsom, P. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*.

Nallapati, R.; Zhou, B.; dos Santos, C.; Gulcehre, C.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of SIGNLL*.

Narayan, S.; Gardent, C.; Cohen, S. B.; and Shimorina, A. 2017. Split and rephrase. In *Proc. of EMNLP*.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proc. of EMNLP*.

Neubig, G., and et al. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Parker, R. 2011. English Gigaword fifth edition LDC2011T07. *Philadelphia: Linguistic Data Consortium*.

Pasunuru, R., and Bansal, M. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proc. of NAACL*.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for sentence summarization. In *Proceedings of EMNLP*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

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

Song, K.; Zhao, L.; and Liu, F. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proc. of COLING*.

Tan, J.; Wan, X.; and Xiao, J. 2017. Abstractive document summarization with a graph-based attentional neural model. In *ACL*.

Thadani, K., and McKeown, K. 2013. Supervised sentence fusion with single-stage inference. In *Proc. of IJCNLP*.

Wang, L.; Raghavan, H.; Castelli, V.; Florian, R.; and Cardie, C. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

Wang, X.; Pham, H.; Yin, P.; and Neubig, G. 2018. A tree-based decoder for neural machine translation. In *Proc. of EMNLP*.

Wu, S.; Zhang, D.; Yang, N.; Li, M.; and Zhou, M. 2017. Sequence-to-dependency neural machine translation. In *ACL*.

Yoshida, Y.; Suzuki, J.; Hirao, T.; and Nagata, M. 2014. Dependency-based discourse parser for single-document summarization. In *Proc. of EMNLP*.