

# Neural Extractive Text Summarization with Syntactic Compression

Jiacheng Xu and Greg Durrett

Department of Computer Science

The University of Texas at Austin

{jcxu, gdurrett}@cs.utexas.edu

## Abstract

Recent neural network approaches to summarization are largely either sentence-extractive, choosing a set of sentences as the summary, or abstractive, generating the summary from a seq2seq model. In this work, we present a neural model for single-document summarization based on joint extraction and compression. Following recent successful extractive models, we frame the summarization problem as a series of local decisions. Our model chooses sentences from the document and then decides which of a set of compression options to apply to each selected sentence. We compute this set of options using discrete compression rules based on syntactic constituency parses; however, our approach is modular and can flexibly use any available source of compressions. For learning, we construct oracle extractive-compressive summaries that reflect uncertainty over our model’s decision sequence, then learn both of our components jointly with this supervision. Experimental results on the CNN/Daily Mail and New York Times datasets show that our model achieves the state-of-the-art performance on content selection evaluated by ROUGE. Moreover, human and manual evaluation show that our model’s output generally remains grammatical.

## 1 Introduction

Neural network approaches to document summarization have ranged from purely extractive (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018) to abstractive (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; See et al., 2017; Tan et al., 2017; Gehrmann et al., 2018). Extractive systems are robust and straightforward to use. Abstractive systems are more flexible for varied summarization situations (Grusky et al., 2018), but can make factual errors (Cao

et al., 2018; Li et al., 2018) or fall back on extraction in practice (See et al., 2017). Extractive and compressive systems (Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2011; Qian and Liu, 2013; Durrett et al., 2016) combine the strengths of both approaches. However, there has been little work studying neural network models for these approaches; past work in the neural domain has largely used abstractive-style sentence compression (Chen and Bansal, 2018).

In this work, we propose a model that can combine the high performance of neural extractive systems, additional flexibility from compression, and interpretability given by having discrete compression options. Our model first encodes the source document and its sentences. It then sequentially selects a set of sentences to further compress. Each sentence has a set of compression options available that are selected to preserve meaning and grammaticality; these are derived from syntactic constituency parses and represent an expanded set of discrete options from prior work (Berg-Kirkpatrick et al., 2011; Wang et al., 2013; Durrett et al., 2016). The neural model additionally scores and chooses which compressions to apply given the context of the document, the sentence, and the decoder model’s recurrent state.

A principal challenge of dealing with extractive models is training, as gold standard reference summaries are unreachable. Past approaches have used reinforcement learning in combination with oracle pre-extraction (Narayan et al., 2018; Chen and Bansal, 2018). We focus on identifying a set of high-quality oracle extractive-compressive summaries for each document, rather than using high-variance policy gradient training. We use beam search over oracle-compressed sentences to identify good sentences to extract; our decoder is allowed to select these in any order. We then derive oracle compression decisions for each sen-

tence in the document through an additional refinement process. Our model’s training objective combines these extractive and compressive components and learns them jointly.

We conduct experiments on a few single document news summarization datasets: CNN, Daily Mail (Hermann et al., 2015), and the New York Times Annotated Corpus (Sandhaus, 2008). Our model matches or exceeds the state-of-the-art on all of these datasets, achieving 41.7 ROUGE-1  $F_1$  on CNN/DM. Analyzing these results more deeply, we see the largest improvement on CNN (+2.4 ROUGE- $F_1$  over our extractive baseline) due to the more compressed nature of CNN summaries. We show that our model’s compression threshold is robust across a range of settings yet tunable to give different-length summaries. Finally, we investigate the fluency and grammaticality of our compressed sentences. The human evaluation shows that our system yields generally grammatical output, with many remaining errors being attributed to the parser.<sup>1</sup>

## 2 Background: Compression

Sentence compression is a long-studied problem dealing with how to delete the least critical information in a sentence to make it shorter (Knight and Marcu, 2000, 2002; Martins and Smith, 2009; Cohn and Lapata, 2009; Wang et al., 2013; Li et al., 2014). Many of these approaches are syntax-driven, though end-to-end neural models have been proposed as well (Filippova et al., 2015; Wang et al., 2017).

Our syntax-driven approach follows a line of successful prior compressive summarization work with non-neural models (Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2011). Approaches based on linguistic preprocessing, including this and RST-based approaches (Carlson et al., 2001; Hirao et al., 2013; Li et al., 2016), have several advantages. First, our model can learn which of the available compression options are most appropriate for the summarization task at hand, giving us more flexibility.<sup>2</sup> Second, the model’s output is more interpretable and controllable, as we have an

<sup>1</sup>The code, full model output, and the pre-trained model are available at <https://github.com/jiacheng-xu/neu-compression-sum>

<sup>2</sup>Directly learning to compress in an extractive summarization framework using gold summaries as supervision is challenging, as optimizing for ROUGE does not guarantee grammaticality. One generally would need to somehow optimize for grammaticality as well (Pasunuru and Bansal, 2018).

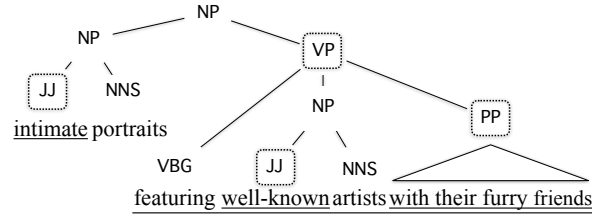


Figure 1: Text compression example. In this case, “intimate”, “well-known”, “with their furry friends” and “featuring ... friends” are deletable given compression rules.

explicit compression threshold parameter and can curate the set of compressions we use. This is in contrast to abstractive summarization, where the decoder model is trained to exactly produce the target summary word by word. Such models are highly flexible, but factuality and correctness of the generated output are sometimes compromised.

**Compression Rules** We refer to the rules derived in Li et al. (2014); Wang et al. (2013); Durrett et al. (2016) and design a concise set of syntactic rules including the removal of:

- Appositive noun phrases;
- Relative clauses and adverbial clauses;
- Adjective phrases in noun phrases, and adverbial phrases (see Figure 1);
- Gerundive verb phrases as part of noun phrases (see Figure 1);
- Prepositional phrases in certain configurations like *on Monday*;
- Content within parentheses and other parentheticals;

Each rule is formulated as a criterion matching a constituent based on its immediate syntactic environment and lexical environment. Figure 1 shows examples of several compression rules applied to a short snippet: we can delete adjectives, a prepositional phrase, and a gerundive verb phrase. All combinations of compressions maintain grammaticality, though some content is likely fairly important in this context (the VP and PP) and should not be deleted. Our model must learn not to delete these elements.

While our rules aim to be effective and applicable to a wide range of corpora, it is difficult to achieve full cross-domain generality: for example, punctuation conventions for appositive noun phrases can alternate between “-”, “-”, and “,”.

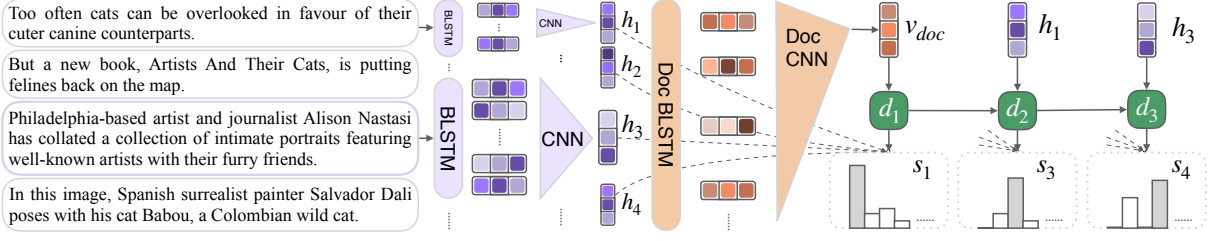


Figure 2: Sentence extraction module of JECS. Words in input document sentences are encoded with BiLSTMs. Two layers of CNNs aggregate these into sentence representations  $h_i$  and then the document representation  $v_{doc}$ . This is fed into an attentive LSTM decoder which selects sentences based on the decoder state  $d$  and the representations  $h_i$ , similar to a pointer network.

For this work, we consider our rules a proof-of-concept solution; for specific domains or summarization applications, more finely-tuned sets of rules could lead to higher performance and be used in our framework seamlessly. This configurability is one advantage compared to a pre-trained compression model.

### 3 Model

Our model is a neural network model that encodes a source document, chooses sentences from that document, and selects discrete compression options. The model architecture of sentence extraction module and text compression module are shown in Figure 2 and 3. The training and the oracle construction of the model will be discussed in Section 4.

#### 3.1 Extractive Sentence Selection

A single document consists of  $n$  sentences  $D = \{s_1, s_2, \dots, s_n\}$ . The  $i$ -th sentence is denoted as  $s_i = \{w_{i1}, w_{i2}, \dots, w_{im}\}$  where  $w_{ij}$  is the  $j$ -th word in  $s_i$ . The content selection module learns to pick up a subset of  $D$  denoted as  $\hat{D} = \{\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k, | \hat{s}_i \in D\}$  where  $k$  sentences are selected.

**Sentence & Document Encoder** We first use a bidirectional LSTM to encode words in each sentence in the document separately and then we apply multiple convolution layers and max pooling layers to extract the representation of every sentence. More specifically,

$$[h_{i1}, \dots, h_{im}] = \text{BiLSTM}([w_{i1}, \dots, w_{im}])$$

$$h_i = \text{CNN}([h_{i1}, \dots, h_{im}])$$

where  $h_i$  is the representation of the  $i$ -th sentence in the document. This process is shown in the left

side of Figure 2 illustrated in purple blocks. Previous work (Martins and Smith, 2009; Li et al., 2014; Wang et al., 2013; Durrett et al., 2016) used surface features including both word level (POS tags, stop words, position in the sentence, etc.) and sentence level (length, location, etc.) features while we only use neural networks to capture contextualized information.

After sentence encoding, we have the hidden representations  $h_{sent} = \{h_1, h_2, \dots, h_n\}$  for sentences  $D = \{s_1, s_2, \dots, s_n\}$ . We then aggregate these sentence representations into a document representation  $v_{doc}$  with a similar BiLSTM and CNN combination, shown in Figure 2 with orange blocks.

**Decoding** The decoding stage selects a number of sentences given the document representation  $v_{doc}$  and sentences' representations  $h_i$ . This process is depicted in the right half of Figure 2. We use a sequential LSTM decoder where, at each time step, we take the representation  $h$  of the last selected sentence, the overall document vector  $v_{doc}$ , and the recurrent state  $d_{t-1}$ , and produce a distribution over all of the remaining sentence representations excluding those already selected. This approach resembles pointer network-style approaches used in past work (Zhou et al., 2018).

More formally, the recurrence is defined as

$$d_t = \text{LSTM}(d_{t-1}, h_k, v_{doc})$$

$$\text{score}_{t,i} = W_m \tanh(W_d d_t + W_h h_i)$$

$$\hat{s}_t = \arg\max(\text{score}_{t,i})$$

$$p(\hat{s}_t = s_i | d_t, h_k, v_{doc}, h_i) = \text{softmax}(\text{score}_{t,i})$$

where  $h_k$  is the representation of the sentence selected at time step  $t - 1$ .  $d_{t-1}$  is the decoding hidden state from last time step.  $W_d$ ,  $W_h$ ,  $W_m$  and parameters in LSTM are learned. Once a sentence

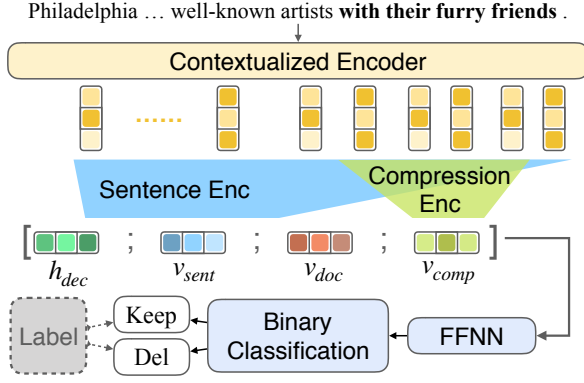


Figure 3: Text compression module. A neural classifier scores the compression option (*with their furry friends*) in the sentence and broader document context and decides whether or not to delete it.

is selected, it cannot be selected again.<sup>3</sup>

### 3.2 Text Compression

After selecting the sentences, the text compression module evaluates our discrete compression options and decides whether to remove certain phrases or words in the selected sentences. Figure 3 shows an example of this process for deciding whether or not to delete a PP in this sentence. This PP was marked as deletable based on rules described in Section 2. Our network then encodes this sentence and the compression, combines this information with the document context  $v_{doc}$  and decoding context  $h_{dec}$ , and uses a feedforward network to decide whether or not to delete the span.

Let  $C_i = \{c_{i1}, \dots, c_{il}\}$  denote the possible compression spans derived from the rules described in Section 2. Let  $y_{i,c}$  be a binary variable equal to 1 if we are deleting the  $c$ th option of the  $i$ th sentence. Our text compression module models  $p(y_{i,c}|D, \hat{s})$  as described in the following section.

**Compression Encoder** We use a contextualized encoder to encode the whole sentence. In this paper, we use ELMo (Peters et al., 2018) as a black box to compute contextualized word representations. We then use convolutional neural networks with max pooling to encode the sentence (shown in blue in Figure 3) and the candidate compression (shown in light green in Figure 3). The sentence

representation  $v_{sent}$  and the compression span representation  $v_{comp}$  are concatenated with the hidden state in sentence decoder  $h_{dec}$  and the document representation  $v_{doc}$ .<sup>4</sup>

**Compression Classifier** We feed the concatenated representation to a feedforward neural network to predict whether the compression span should be deleted or kept, which is formulated as a binary classification problem. This classifier computes the final probability  $p(y_{i,c}|h_{dec}, v_{doc}, c, s_i)$

The overall probability of a summary  $(\hat{s}, \hat{y})$  is then:

$$p(\hat{s}, \hat{y}|D) = \prod_{t=1}^T \left[ p(\hat{s}_t|\hat{s}_{<t}, D) \prod_{c \in C_i} p(\hat{y}_{t,c}|\hat{s}, D) \right]$$

**Heuristic Deduplication** Inspired by the trigram avoidance trick proposed in Paulus et al. (2018) to reduce redundancy, we take full advantage of our linguistically motivated compression rules and the constituent parse tree and allow our model to compress deletable chunks with redundant information. In our example in Figure 1, the word *intimate* actually appears several times throughout the document and in the reference summary, the phrase *intimate portraits* is intentionally abbreviated as *portraits*. Typically, readers can understand what is meant in cases like this as long as the possibly compressed words appear somewhere. We, therefore, take our model’s output and apply a postprocessing stage where we remove any compression option whose unigrams are completely covered elsewhere in the summary. We perform this compression after the model prediction and compression.

## 4 Training

Our model makes a series of sentence extraction decisions  $\hat{s}$  and then compression decisions  $\hat{y}$ . To supervise it, we need to derive gold-standard labels for these decisions. Depending on which sentences are extracted, different compression decisions may be optimal; however, re-deriving these with a dynamic oracle (Goldberg and Nivre, 2012) is prohibitively expensive during training. We therefore find a “compromise” set of compression decisions for each sentence in the document, so we

<sup>3</sup>For our experiments, we decode for a fixed number of sentences, tuned for each dataset, as in prior extractive work (Narayan et al., 2018). We experimented with dynamically choosing a number of sentences and found this to make little difference.

<sup>4</sup>In initial experiments, we also found ELMo to be useful on the encoder side as well. However, to simplify comparisons with past work and due to scaling issues, we utilize it in the compression stage only.



**Reference:** Artist and journalist Alison Nastasi put together the portrait collection. Also features images of Picasso, Frida Kahlo, and John Lennon. Reveals quaint personality traits shared between artists and their felines.

**Document:** ... **Philadelphia-based** artist and journalist Alison Nastasi has collated a collection of **intimate** portraits **featuring well-known artists with their furry friends**. ...

Compression $R_{bf} = 19.4$	$R_{af}$	Ratio	Label
Philadelphia-based	19.8	1.02	DEL
intimate	19.8	1.02	DEL
well known	20.4	1.05	DEL
featuring ... their furry friends	18.1	0.93	KEEP

Table 1: Oracle label computation for the text compression module.  $R_{bf}$  and  $R_{af}$  are the ROUGE scores before and after compression. The ratio is defined as  $\frac{R_{af}}{R_{bf}}$ . ROUGE increases when words not appearing in the reference are deleted. ROUGE can decrease when terms appearing in the reference summary, like *featuring*, are deleted.

can appropriately supervise the compression layer no matter what sentences are extracted.

#### 4.1 Oracle Construction

**Extractive Oracle** An oracle for purely extractive summarization consists of a set of sentences with the highest possible ROUGE score. ILP-based approaches can exactly optimize for ROUGE recall given a reference summary (Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011); however, exactly doing this optimization for ROUGE  $F_1$  is intractable<sup>5</sup> and generally takes time  $O(n^k)$  to select  $k$  out of  $n$  sentences.

We therefore identify an oracle with a beam search procedure inspired by Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998). For each additional sentence we propose to add, we compute a heuristic cost equal to the ROUGE score of a given sentence with respect to the reference summary. When pruning states, we calculate the ROUGE score of the combination of sentences currently selected and sort in descending order. Let the beam width be  $\beta$ . The time complexity of the approximate approach is  $O(nk\beta)$  where in practice  $k \ll n$  and  $\beta \ll n$ . We set  $\beta = 8$  and  $n = 30$  which means we only consider the first 30 sentences in the document.

**Joint Extractive & Compressive Oracle** The extractive oracle we just described does not take compression into account. One can imagine that

<sup>5</sup>Computing precision requires dividing by the number of selected words, making the objective no longer linear.

Category	CNN	DM	NYT
Bad	27%	48%	49%
Weak Positive	58%	43%	47%
Strong Positive	15%	10%	4%

Table 2: The oracle label distribution over three datasets. Compressions in the “Bad” category decrease ROUGE and are labeled as negative (do not delete), while weak positive (less than 5% ROUGE improvement) and strong positive (greater than 5%) both represent ROUGE improvements. CNN features much more compression than the other datasets.

knowing a particular sentence will be compressed changes the set of other sentences that should be selected alongside it. To optimize over extraction and compression jointly, one method is to do a beam search over a larger state space where each action consists of either selecting a new sentence or compressing the most recent sentence. However, this space is too large to search effectively with beam search.

Our approach factors these decisions in a heuristic but effective way. Let  $s_i^0$  denote the original  $i$ th sentence. Let  $s_i^k$  denote the  $i$ th sentence with the optimal set of  $k$  compression options applied, as measured by ROUGE gain on that sentence with that compression option applied. To compute our sentence extractive oracle, we change our candidate sentences pool to  $\{s_i^1\}$ , the set of sentences with one compression being applied to each.<sup>6</sup> We then follow the same beam search procedure that was used to choose the extractive oracle.

**Compression Label** With a set of “compression-aware” sentence labels  $s^*$ , we now need to compute the truly optimal compression decisions. For each compression option, we calculate the ROUGE score of the sentence that contains it before and after the compression is applied. Any option that increases ROUGE is treated as a compression that should be applied. When calculating this ROUGE value, we remove stop words, stem, and handle redundant words in the sentence. Empirically, this approach led to a good balance of positive- and negative-labeled compressions. Having a somewhat balanced

<sup>6</sup>Searching over  $k$  and choosing the optimal number of compressions often leads to very little compression being applied. Sentences evaluated in isolation usually attain high ROUGE by maximizing selected content, even if compressing content is ultimately the right decision when multiple sentences are selected.

set is critical to making the compression labels learnable so that the model can apply compression appropriately.

The fraction of positive and negative labels assigned to compression options is shown for each of the three datasets in Table 2. CNN appears to feature more compression than the other datasets. However, the compression decisions are reasonably balanced between positive and negative in all three settings.

## 4.2 Learning Objective

Often, many oracle summaries achieve similar ROUGE values. We therefore want to avoid committing to a single oracle summary for the learning process. As a result, we can generate  $m$  extractive oracles  $s_i^*$ ; let  $s_{i,t}^*$  denote the gold sentence for the  $i$ -th oracle at timestep  $t$ . Past work (Narayan et al., 2018; Chen and Bansal, 2018) has employed policy gradient in this setting to optimize directly for ROUGE. However, because oracle summaries usually have very similar ROUGE scores, we choose to simplify this objective as follows:

$$\mathcal{L}_{\text{sent}} = -\frac{1}{m} \sum_{i=1}^m \sum_{t=1}^T \log p(s_{i,t}^* | D, s_{i,<t}^*)$$

Put another way, we optimize the log likelihood averaged across  $m$  different oracles to ensure that each has high likelihood. We use  $m = 5$  oracles during training.

We also don’t require our model to necessarily produce the oracle sentences in the correct order. At each step of training, we maximize the marginal log likelihood of extracting any correct future sentence:

$$\mathcal{L}_{\text{sent}} = -\frac{1}{m} \sum_{i=1}^m \sum_{t=1}^T \log \sum_{t'=t}^T p(s_{i,t'}^* | D, s_{i,<t}^*)$$

We found that this more flexible objective improved our model’s learnability.

The learning of the compression module is a binary classification problem.

$$\mathcal{L}_{\text{comp}} = -\sum_{i=1}^m \sum_{c=1}^C \log p(y_{i,c}^* | D, \hat{s})$$

where  $p(y_{i,c}^*)$  is the probability of the target decision for the  $c$ -th compression options of the  $i$ -th sentence.

The joint loss function is  $\mathcal{L} = \mathcal{L}_{\text{sent}} + \alpha \mathcal{L}_{\text{comp}}$ . We set  $\alpha = 1$  in practice.

Dataset	#Train	#Dev	#Test	Ref len	Doc len
CNN	90266	1220	1093	37	540
DM	196961	12148	10397	61	593
NYT50	137778	17222	17223	88	727

Table 3: Statistics of the CNN, Daily Mail, and NYT50 (see text) datasets. CNN features the shortest reference summaries overall, and this is where we find compression is most effective.

## 5 Experimental Setup

### 5.1 Datasets

We evaluate the proposed method on three popular news summarization datasets: the New York Times corpus (Sandhaus, 2008), CNN and Daily-mail (DM) (Hermann et al., 2015).

We preprocess the datasets with the scripts provided by See et al. (2017), which uses Stanford CoreNLP tokenization (Manning et al. (2014)). We use the non-anonymized version of the CNN/DM as in previous summarization work. For the New York Times Corpus, we filter out the examples with abstracts shorter than 50 words following the criteria in (Durrett et al., 2016), yielding the NYT50 dataset. The statistics of the datasets are listed in Table 3. During sentence selection, we always select 3 sentences for CNN/DM and 5 sentences for NYT, which gave the best performance. For our syntactic analysis, all datasets are parsed with the constituency parser in Stanford CoreNLP (Manning et al., 2014).

### 5.2 Models

We present several variants of our model to show how extraction and compression work jointly. In extractive summarization, the LEAD baseline (first  $k$  sentences) is a strong baseline due to how newswire articles are written. LEADDEDUP is a non-learned baseline that uses our heuristic deduplication technique on the lead sentences. LEAD-COMP is a compression only model where compression is performed on the lead sentences. This shows the effectiveness of the compression module in isolation rather than in the context of abstraction. EXTRACTION is the extraction only model. JECS is the full Joint Extractive and Compressive Summarizer.

We compare our model with various abstractive and extractive summarization models. NeuSum (Zhou et al., 2018) uses a seq2seq model to predict a sequence of sentences indices to be picked

Model	CNNDM			CNN			DM		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Lead PointGenCov (See et al., 2017)	40.3	17.7	36.6	—	—	—	—	—	—
Lead Neusum (Zhou et al., 2018)	40.2	17.7	36.5	—	—	—	—	—	—
Lead BanditSum (Dong et al., 2018)	40.0	17.5	36.2	28.8	11.0	25.5	41.2	18.2	37.3
Lead (Ours)	40.3	17.6	36.4	29.1	11.1	25.8	41.3	18.2	37.4
Refresh* (Narayan et al., 2018)	40.0	18.1	36.6	30.3	<b>11.6</b>	26.9	41.0	18.8	37.7
NeuSum	<b>41.6</b>	<b>19.0</b>	38.0	—	—	—	—	—	—
LatSum* (Zhang et al., 2018)	41.0	18.8	37.4	28.8	11.5	25.4	<b>42.3</b>	<b>19.5</b>	<b>38.6</b>
LatSum w/ Compression	36.7	15.4	34.3	—	—	—	—	—	—
BanditSum	41.5	18.7	37.6	<b>30.7</b>	<b>11.6</b>	<b>27.4</b>	42.1	18.9	38.3
PointGenCov	39.5	17.3	36.4	—	—	—	—	—	—
CBDec (Jiang and Bansal, 2018)	40.7	17.9	37.1	—	—	—	—	—	—
FARS (Chen and Bansal, 2018)	40.9	17.8	<b>38.5</b>	—	—	—	—	—	—
LEADEDUP	40.5	17.4	36.5	29.7	10.9	26.2	41.7	18.1	37.7
LEADCOMP	40.8	17.4	36.8	30.6	10.8	27.2	42.0	18.0	37.9
EXTRACTION	40.7	18.0	36.8	30.3	11.0	26.5	41.7	18.9	38.0
JECS	<b>41.7</b>	<b>18.5</b>	<b>37.9</b>	<b>32.7</b>	<b>12.2</b>	<b>29.0</b>	<b>42.7</b>	<b>19.2</b>	<b>38.8</b>

Table 4: Experimental results on the test sets of CNN and DM. We report ROUGE F1 of models trained and tested on CNN/DM together as well as separately. \* indicates models evaluates with our own ROUGE metrics. Our model outperforms our extractive model and lead-based baselines, as well as prior work. Gains are more pronounced on CNN because this dataset features shorter, more compressed reference summaries.

up from the document. Our extractive approach is most similar to this model. **Refresh** (Narayan et al., 2018) is a sentence ranking based extractive summarization model. **BanditSum** (Dong et al., 2018) treats extractive summarization as a contextual bandit problem. **LatSum** (Zhang et al., 2018) proposed a latent variable extractive model with an additional abstractive compression module.

We also compare with some abstractive models. **PointGenCov** (See et al., 2017) is an abstractive model with copy and coverage mechanisms. **FARS** (Chen and Bansal, 2018) proposed an abstractive rewriting model on top of extractive sentence selection. **CBDec** (Jiang and Bansal, 2018) is an abstractive seq2seq model with some improvement on the decoder.

### 5.3 Implementation Details

We use the same pretrained word embeddings used in (Narayan et al., 2018). The size of the sentence and document representation vectors is 200. For the compression module, we use ELMo as the contextualized encoder without fine-tuning the parameter and project the vectors back to 200 dimensions after the ELMo layer. Dropout is applied after word embedding layers and LSTM layers at a rate of 0.2. We use the Adam optimizer (Kingma and Ba, 2014) with the initial learning rate at 0.001. The model converges after 2 epochs of training.

**Evaluation Metrics** We use ROUGE (Lin, 2004) for evaluation.<sup>7</sup> During oracle construction, we use simplified unigram and bigram  $F_1$  scores as a faster approximation to the full ROUGE.

## 6 Results

We evaluate our model on two axes. First, for content selection, we use ROUGE as is standard; this allows for extensive comparison with prior work. Second, we evaluate the grammaticality of our model to ensure that it is not substantially damaged by compression. Following past work, we use human evaluators and have them focus on this task rather than assessing overall summary quality (Gillick and Liu, 2010).

### 6.1 Content Selection on CNN/DM

Table 4 shows the experimental results on CNN/DM. First, we list LEAD baselines reported in each paper as a point of reference for comparison, since different authors use slightly different tokenization or ROUGE arguments. In the following rows, we list the performance of competitor models on these datasets. Starred models are evaluated according to our ROUGE metrics; numbers very closely match the originally reported results.

Our models yield strong performance compared to these approaches. Our basic EXTRACTION achieves comparable results to past successful ex-

<sup>7</sup>Command line parameters: “-c 95 -m -n 2”

Model	R-1	R-2	R-L
Lead	41.8	22.6	35.0
LEADDEDUP	42.0	22.8	35.0
LEADCOMP	42.4	22.7	35.4
EXTRACTION	44.3	25.5	37.1
JECS	45.5	25.3	38.2

Table 5: Experimental results on the NYT50 dataset. ROUGE-1, -2 and -L  $F_1$  is reported. JECS substantially outperforms our Lead-based systems and our extractive model.

tractive approaches. JECS improves on this across the datasets. However, we note a striking difference between our performance on CNN and on Daily Mail, with all compressive models giving bigger benefits on CNN compared to Daily Mail (+2.4 ROUGE-1  $F_1$  from compression vs. +1.0 ROUGE-1  $F_1$ ). A possible reason is suggested by Table 3, where CNN has significantly shorter summaries overall. This suggests that compression, while useful, will have its benefits more clearly apparent in evaluation on certain datasets.

Second, we note that compression is somewhat effective in isolation, as shown by the performance of LEADDEDUP and LEADCOMP. But we also note that these techniques in isolation give less benefit (on top of LEAD) than when combined with the extractive model (JECS) in our joint framework.

Our model slightly underperforms on ROUGE-2 in some cases. One possible reason is that we remove stop words when constructing our oracles, which could underestimate the importance of bigrams containing stopwords for evaluation.

Finally, we note that our compressive approach substantially outperforms the compression-augmented LatSum model. That model used a separate seq2seq model for rewriting, which is potentially harder to learn than our compression model and which is not learned jointly with the extraction.

## 6.2 Content Selection on NYT

Table 5 shows the experimental results on the NYT50 dataset. We see again that the inclusion of compression leads to improvements in both the LEAD setting as well as for our full JECS model.<sup>8</sup>

<sup>8</sup>Paulus et al. (2018) do not use the NYT50 dataset, so our results are not directly comparable to theirs. Durrett et al. (2016) use a different evaluation setup with a hard limit on the summary length of 50 words and evaluating on recall only.

Model	Mean $\pm$ CI
EXTRACTIONDROPOUT	6.22 $\pm$ 0.126
JECS	6.36 $\pm$ 0.124
EXTRACTION	6.39 $\pm$ 0.123
POINTGENCOV	6.44 $\pm$ 0.122

Table 6: Human evaluation results from Mechanical Turk. Turkers rated summaries on a scale from 1 to 10. We report the mean and the 95% confidence interval of the ratings.

## 6.3 Grammaticality

We evaluate grammaticality in two ways: using Amazon Mechanical Turk following prior work (Gillick and Liu, 2010; Durrett et al., 2016) and with manual analysis.

**Human Evaluation** We first conduct the human evaluation on Amazon Mechanical Turk platform. Our prompt is shown in the Appendix. We ask Turkers to rate grammaticality on a scale from 1 to 10. We compare four models: our EXTRACTIVE model, our full JECS model, and the PointGenCov of See et al. (2017). We also propose another baseline, EXTRACTDROPOUT, which randomly drops 10% of the words in a sentence rounded down (so one token will be dropped if the length of the sentence ranges from 10 to 19). We sampled 400 sentences from each of the four models; each sentence of every model is rated by 3 Turkers *in isolation*, not evaluated in the context of the whole summary.

The results are shown in Table 6. EXTRACTIONDROPOUT has the lowest performance, and all other systems are very close. It appears that Turkers are sensitive enough to pick up on randomly dropped words, and our system performs better than that. One possible reason for overall low scores is that we lowercased all words to be comparable to the output of See et al. (2017), which may have thrown off Turkers despite the provided instructions.

**Manual Error Analysis** Our model’s errors may be sub-threshold for Turkers. To get a better sense of our model’s output from an expert standpoint, we conduct a manual analysis of our applied compressions to get a sense of how many are valid. We manually examined 40 model summaries, comparing the output with the raw sentences before compression, and identified the following errors:



1. 8 bad deletions due to parsing errors like *a UK [JJ national] from London*.
2. 8 inappropriate adjective deletions causing correctness issues with respect to the reference document. Examples include *[former] president* and *[nuclear] weapon*.
3. 3 other errors including the partial deletion of slang, inappropriate PP attachment deletion, and miscellaneous mistakes like *students [first], athletes [second]*.

Improving the parser could therefore improve grammaticality, along with a strengthened, more semantically-aware set of compression rules. One advantage of our approach is that compression is a modular part of the summarization system, so we believe these components could be improved to address these errors without disrupting the effectiveness of the rest of the system.

## 7 Compression Analysis

### 7.1 Compression Threshold

Compression in our model is an imbalanced binary classification problem. The natural classification threshold (probability of DEL > 0.5) may not be optimal, as the model’s behavior depends on the length of the reference summaries, how many sentences are extracted, and how the oracle is constructed. We experiment with varying the classification threshold from 0 (no deletion, only heuristic deduplication) to 1 (all compressible pieces removed). The results on CNN are shown in Figure 4. We present the average of ROUGE-1, -2 and -L of the outcome sentences as an aggregate metric. The dotted horizontal line shows the ROUGE value of the extractive baseline. The model achieves the best performance at 0.45 but performs well in a wide range from 0.3 to 0.55. Our compression is therefore robust yet also provides a controllable parameter to change the amount of compression in produced summaries.

### 7.2 Compression Type Analysis

We further break down the types of compressions used in the model. There are two pertinent questions: first, what compressions are available to the model? Second, what compressions actually get chosen? Table 7 answers this first question. We see that a variety of syntactic constituents are available for compression with high frequency.

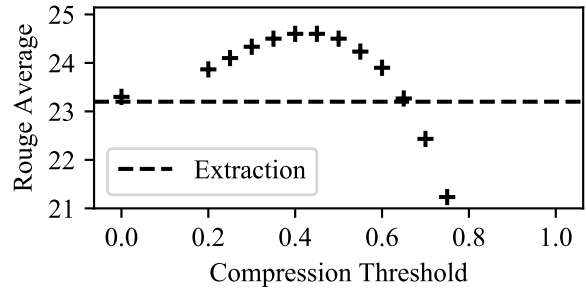


Figure 4: Effect of changing the compression threshold on CNN. The y-axis shows the average of the F1 of ROUGE-1,-2 and -L. The dotted line is the extractive baseline. The model outperforms the extractive model and achieves nearly optimal performance across a range of threshold values.

Node Type	Len	% of comps	Oracle comp %
PP	5.7	39%	67%
JJ	1.0	19%	84%
SBAR	12.1	11%	59%
ADVP	1.3	7%	91%

Table 7: Statistics of compression options on CNN. We show the top four constituency types that are compressible, along with the average length, the fraction of available compressions it accounts for, and how frequently the oracle says to compress these constituents.

Table 7 then shows the compressions that our model ends up choosing. This distribution looks different from that in Table 7, with more JJs getting picked up relative to PPs. PPs are often compressed by the deduplication mechanism because the compressible PPs tend to be temporal and location adjuncts, which may be redundant across sentences. Without the manual deduplication mechanism, our model matches the ground truth around 80% of the time. However, a low accuracy here may not actually cause a low final ROUGE score, as many compression choices only affect the final ROUGE score by a small amount.

## 8 Related Work

**Neural Extractive Summarization** Neural networks have shown to be effective in extractive summarization. Past approaches have structured the decision either as binary classification over sentences (Cheng and Lapata, 2016; Nallapati et al., 2017) or classification followed by ranking (Narayan et al., 2018). Zhou et al. (2018) used a seq-to-seq decoder instead. For our model, text compression forms a module largely orthogonal to the extraction module, although the joint ora-

Node Type	Len	% of comps	Comp Acc	Dedup
JJ	1.0	34%	72%	30%
PP	3.4	26%	47%	72%
ADVP	1.4	17%	79%	17%
PRN	2.2	6%	80%	5%

Table 8: The compressions actually used by our model on CNN; average lengths and the fraction of that constituency type among compressions taken by our model. Comp Acc indicates how frequently that compression was taken by the oracle; note that error, especially keeping constituents that we shouldn’t, may have minimal impact on summary quality. Dedup indicates the percentage of chosen compressions which arise from deduplication as opposed to model prediction. Many PPs are removed in this process contrary to what the oracle states.

cle and joint learning are used in our best model. Additional improvements to extractive modeling might therefore be expected to stack with our approach.

**Syntactic Compression** Prior to the explosion of neural models for summarization, syntactic compression was relatively more common. [Martins and Smith \(2009\)](#) cast joint extraction and compression as an ILP and used dependency parsing information in their model. [Woodsend and Lapata \(2011\)](#) induced a quasi-synchronous grammar from Wikipedia for compression. Several systems explored the usage of constituency parses ([Berg-Kirkpatrick et al., 2011](#); [Wang et al., 2013](#); [Li et al., 2014](#)) as well as RST-based approaches ([Hirao et al., 2013](#); [Durrett et al., 2016](#)). Our approach follows in this vein but could be combined with more sophisticated neural text compression methods as well.

**Neural Text Compression** [Filippova et al. \(2015\)](#) presented an LSTM approach to deletion-based sentence compression. [Miao and Blunsom \(2016\)](#) proposed a deep generative model for text compression. [Zhang et al. \(2018\)](#) explored the compression module after the extraction model but the separation of these two modules hurt the performance.

## 9 Conclusion

In this work, we presented a neural network model for extractive and compressive summarization. Our model consists of a sentence extraction model joined with a compression classifier that decides whether or not to delete syntax-derived com-

pression options for each sentence. Training the model involves finding an oracle set of extraction and compression decision with high score, which we do through a combination of a beam search procedure and heuristics. Our model outperforms past work on the CNN/Daily Mail corpus in terms of ROUGE, achieves substantial gains over the extractive model, and appears to have acceptable grammaticality according to human evaluations.

## Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, a Bloomberg Data Science Grant, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. Thanks as well to Shashi Narayan, Qingyu Zhou, and Xingxing Zhang for providing their model outputs, and Yasumasa Onoe for the help with visualization.

## References

- Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. [Jointly Learning to Extract and Compress](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490. Association for Computational Linguistics.
- Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. [Faithful to the Original: Fact Aware Neural Abstractive Summarization](#). In *AAAI Conference on Artificial Intelligence*.
- Jaime Carbonell and Jade Goldstein. 1998. [The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries](#). In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’98, pages 335–336, New York, NY, USA. ACM.
- Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. 2001. [Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory](#). In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.
- Yen-Chun Chen and Mohit Bansal. 2018. [Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting](#). In *Proceedings of the 56th Annual*

- Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 675–686. Association for Computational Linguistics.
- Jianpeng Cheng and Mirella Lapata. 2016. [Neural Summarization by Extracting Sentences and Words](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.
- Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. [Abstractive Sentence Summarization with Attentive Recurrent Neural Networks](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. Association for Computational Linguistics.
- Trevor Cohn and Mirella Lapata. 2009. [Sentence Compression As Tree Transduction](#). *J. Artif. Int. Res.*, 34(1):637–674.
- Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. [BanditSum: Extractive Summarization as a Contextual Bandit](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748. Association for Computational Linguistics.
- Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. [Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008. Association for Computational Linguistics.
- Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. [Sentence Compression by Deletion with LSTMs](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368. Association for Computational Linguistics.
- Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. [Bottom-Up Abstractive Summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109. Association for Computational Linguistics.
- Dan Gillick and Benoit Favre. 2009. [A Scalable Global Model for Summarization](#). In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18. Association for Computational Linguistics.
- Dan Gillick and Yang Liu. 2010. [Non-expert evaluation of summarization systems is risky](#). In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk*, pages 148–151. Association for Computational Linguistics.
- Yoav Goldberg and Joakim Nivre. 2012. [A Dynamic Oracle for Arc-Eager Dependency Parsing](#). In *Proceedings of COLING 2012*, pages 959–976. The COLING 2012 Organizing Committee.
- Max Grusky, Mor Naaman, and Yoav Artzi. 2018. [Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719. Association for Computational Linguistics.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching Machines to Read and Comprehend](#). In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems* 28, pages 1693–1701. Curran Associates, Inc.
- Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. [Single-Document Summarization as a Tree Knapsack Problem](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520. Association for Computational Linguistics.
- Yichen Jiang and Mohit Bansal. 2018. [Closed-Book Training to Improve Summarization Encoder Memory](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4077. Association for Computational Linguistics.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.
- Kevin Knight and Daniel Marcu. 2000. [Statistics-Based Summarization - Step One: Sentence Compression](#). In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.
- Kevin Knight and Daniel Marcu. 2002. [Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression](#). *Artif. Intell.*, 139(1):91–107.
- Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. [Improving Multi-documents Summarization by Sentence Compression based on Expanded Constituent Parse Trees](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701. Association for Computational Linguistics.
- Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2018. [Ensure the Correctness of the Summary: Incorporate Entailment Knowledge into Abstractive Sentence Summarization](#). In *Proceedings*



- of the 27th International Conference on Computational Linguistics, pages 1430–1441. Association for Computational Linguistics.
- Junyi Jessie Li, Kapil Thadani, and Amanda Stent. 2016. [The role of discourse units in near-extractive summarization](#). In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 137–147. Association for Computational Linguistics.
- Chin-Yew Lin. 2004. [ROUGE: A Package for Automatic Evaluation of Summaries](#). In *Text Summarization Branches Out*.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. [The Stanford CoreNLP Natural Language Processing Toolkit](#). In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.
- Andre Martins and Noah A. Smith. 2009. [Summarization with a joint model for sentence extraction and compression](#). In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.
- Yishu Miao and Phil Blunsom. 2016. [Language as a Latent Variable: Discrete Generative Models for Sentence Compression](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328. Association for Computational Linguistics.
- Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. [SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents](#). In *AAAI Conference on Artificial Intelligence*.
- Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. [Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond](#). In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Ranking sentences for extractive summarization with reinforcement learning](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759. Association for Computational Linguistics.
- Ramakanth Pasunuru and Mohit Bansal. 2018. [Multi-Reward Reinforced Summarization with Saliency and Entailment](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. Association for Computational Linguistics.
- Romain Paulus, Caiming Xiong, and Richard Socher. 2018. [A Deep Reinforced Model for Abstractive Summarization](#). In *International Conference on Learning Representations*.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep Contextualized Word Representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.
- Xian Qian and Yang Liu. 2013. [Fast Joint Compression and Summarization via Graph Cuts](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502. Association for Computational Linguistics.
- Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. [A Neural Attention Model for Abstractive Sentence Summarization](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.
- Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. [Get To The Point: Summarization with Pointer-Generator Networks](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.
- Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. [Abstractive Document Summarization with a Graph-Based Attentional Neural Model](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181. Association for Computational Linguistics.
- Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. [Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1385–1393. Association for Computational Linguistics.
- Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. [A Sentence Compression Based Framework to Query-Focused Multi-Document Summarization](#). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages



1384–1394. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2011. [Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming](#). In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420. Association for Computational Linguistics.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. [Neural Latent Extractive Document Summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. [Neural Document Summarization by Jointly Learning to Score and Select Sentences](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663. Association for Computational Linguistics.

## **A Turk Instructions**

In this task you will be reading text and evaluating it on the basis of how **grammatical** it is. A grammatical sentence should consist of well-formed English writing that flows naturally. It should not be a sentence fragment or have missing components, and it should be understandable in terms of what it is trying to communicate.

Use your best judgment of these factors to give each piece of text a rating from 1 to 10.


- 1 = highly ungrammatical, contains many errors, very hard to understand
- 10 = perfectly grammatical, no errors, looks like professional newspaper-quality writing

These sentences have been intentionally lowercased; please **disregard the punctuation errors and the case of words** when assessing grammaticality.

1. Please rate this piece of text in terms of **grammaticality**.

boko haram still controls swathes of northeastern nigeria.

Not grammatical (1) <===> Grammatical (10)



2. Please rate this piece of text in terms of **grammaticality**

Figure 5: The interface for Amazon turk human evaluation. All of the examples are fully shuffled. Turkers are asked to rate the grammaticality from 1 to 10.