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 selection-based extraction or generation-based abstraction. In this work, we present a neural model for single-document summarization based on joint extraction and syntactic compression. Our model chooses sentences from the document, identifies possible compressions based on constituency parses, and scores those compressions with a neural model to produce the final summary. For learning, we construct oracle extractive-compressive summaries, 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 strong performance (comparable to state-of-the-art systems) as evaluated by ROUGE. Moreover, our approach outperforms an off-the-shelf compression module, and human and manual evaluation shows 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; 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; Qian and Liu, 2013; Durrett et al., 2016) combine the strengths of both approaches; however, there has been little work studying neural network models in this vein, and the approaches that have been employed typically use seq2seq-based sentence compression (Chen and Bansal, 2018).

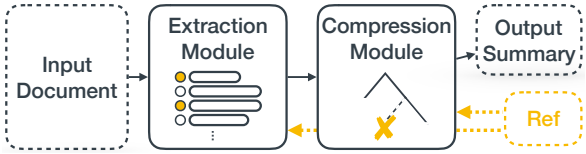

Figure 1: Diagram of the proposed model. Extraction and compression are modularized but jointly trained with supervision derived from the reference summary.

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 and 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). 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 training an extractive and compressive model is constructing the oracle summary for supervision. We identify a set of high-quality sentences from the document with beam search and derive oracle compression labels in each sentence through an additional refinement process. Our model's training objective combines these extractive and compressive components and learns them jointly.

We conduct experiments on standard single document news summarization datasets: CNN, Daily Mail (Hermann et al., 2015), and the New

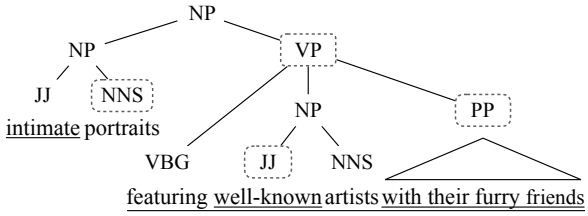

Figure 2: Text compression example. In this case, "intimate", "well-known", "with their furry friends" and "featuring ... friends" are deletable given compression rules.

York Times Annotated Corpus (Sandhaus, 2008). Our model matches or exceeds the state-of-the-art on all of these datasets and achieves 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.[1]

2 Compression in Summarization

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., 2017a). Past non-neural work on summarization has used both syntax-based (Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2011) and discourse-based (Carlson et al., 2001; Hirao et al., 2013; Li et al., 2016) compressions. Our approach follows in the syntax-driven vein.

Our high-level approach to summarization is shown in Figure 1. In Section 3, we describe the models for extraction and compression. Our compression depends on having a discrete set of valid compression options that maintain the grammaticality of the underlying sentence, which we now proceed to describe.

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

Compression Rules We refer to the rules derived in Li et al. (2014), Wang et al. (2013), and Durrett et al. (2016) and design a concise set of syntactic rules including the removal of: 1. Appositive noun phrases; 2. Relative clauses and adverbial clauses; 3. Adjective phrases in noun phrases, and adverbial phrases (see Figure 2); 4. Gerundive verb phrases as part of noun phrases (see Figure 2); 5. Prepositional phrases in certain configurations like *on Monday*; 6. Content within parentheses and other parentheticals.

Figure 2 shows examples of several compression rules applied to a short snippet. All combinations of compressions maintain grammaticality, though some content is fairly important in this context (the VP and PP) and should not be deleted. Our model must learn not to delete these elements.

Compressability Summaries from different sources may feature various levels of compression. At one extreme, a summary could be fully sentence-extractive; at another extreme, the editor may have compressed a lot of content in a sentence. In Section 4, we examine this question on our summarization datasets and use it to motivate our choice of evaluation datasets.

Universal Compression with ROUGE While we use syntax as a source of compression options, we note that other ways of generating compression options are possible, including using labeled compression data. However, supervising compression with ROUGE is critical to learn what information is important for this particular source, and in any case, labeled compression data is unavailable in many domains. In Section 5, we compare our model to off-the-shelf sentence compression module and find that it substantially underperforms our approach.

3 Model

Our model is a neural network model that encodes a source document, chooses sentences from that document, and selects discrete compression options to apply. The model architecture of sentence extraction module and text compression module are shown in Figure 3 and 4.

3.1 Extractive Sentence Selection

A single document consists of n sentences $D = \{s_1, s_2, \cdots, s_n\}$. The i-th sentence is denoted as $s_i = \{w_{i1}, w_{i2}, \cdots, w_{im}\}$ where w_{ij} is the

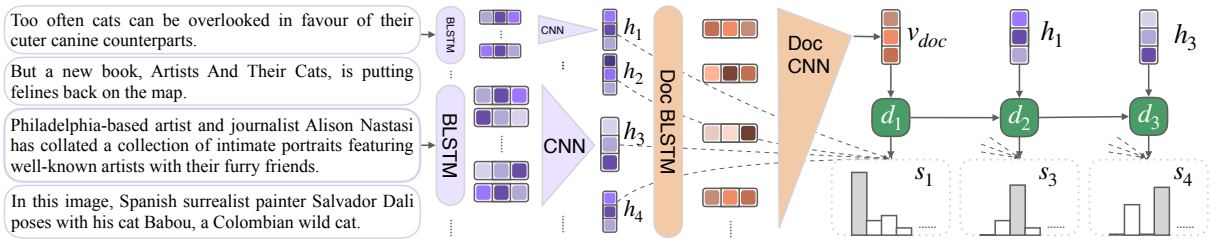

Figure 3: Sentence extraction module of JECS. Words in input document sentences are encoded with BiLSTMs. Two layers of CNNs aggregate these into sentence representations h_i and then the document representation v_{doc}. This is fed into an attentive LSTM decoder which selects sentences based on the decoder state d and the representations h_i, similar to a pointer network.

j-th word in s_i. The content selection module learns to pick up a subset of D denoted as $\hat{D} = \{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_k, |\hat{s}_i \in D\}$ 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. Specifically, $[h_{i1}, \cdots, h_{im}] = \text{BiLSTM}([w_{i1}, \cdots, w_{im}])$ and $h_i = \text{CNN}([h_{i1}, \cdots, h_{im}])$ where h_i is a representation of the i-th sentence in the document. This process is shown in the left side of Figure 3 illustrated in purple blocks. We then aggregate these sentence representations into a document representation v_{doc} with a similar BiLSTM and CNN combination, shown in Figure 3 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 3. 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 sentences excluding those already selected. This approach resembles pointer network-style approaches used in past work (Zhou et al., 2018). Formally, we write this as:

$$d_t = \text{LSTM}(d_{t-1}, h_k, v_{doc})$$
$$\text{score}_{t,i} = W_m \tanh(W_d d_t + W_h h_i)$$
$$p(\hat{s}_t = s_i | d_t, h_k, v_{doc}, h_i) = \text{softmax}(\text{score}_{t,i})$$

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 is selected, it cannot be selected again. At test time, we use greedy decoding to identify the most likely sequence of sentences under our model.[2]

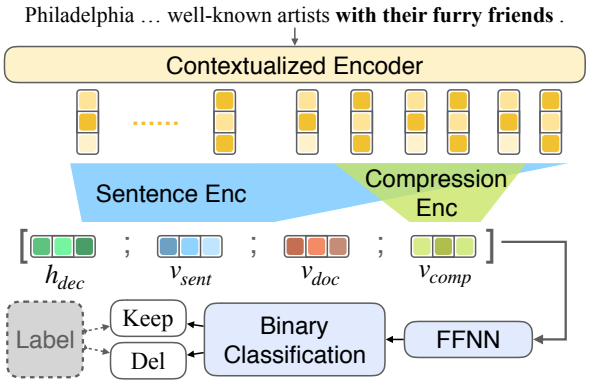

Figure 4: Text compression module. A neural classifier scores the compression option (*with their furry friends*) in the sentence and broader document context and decides whether or not to delete it.

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 4 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.

[2]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.

Let $C_i = \{c_{i1}, \cdots, c_{il}\}$ 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 cth option of the ith sentence. Our text compression module models $p(y_{i,c}|D, \hat{s}_t = s_i)$ as described in the following section.

Compression Encoder We use a contextualized encoder, ELMo (Peters et al., 2018) to compute contextualized word representations. We then use CNNs with max pooling to encode the sentence (shown in blue in Figure 4) and the candidate compression (shown in light green in Figure 4). 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}.

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 $p(y_{i,c}|D, \hat{s}_t = s_i) = p(y_{i,c}|h_{dec}, v_{doc}, v_{comp}, s_i)$. The overall probability of a summary $(\hat{s}, \hat{y})$, where $\hat{s}$ is the sentence oracle and $\hat{y}$ is the compression label, is the product of extraction and compression models: $p(\hat{s}, \hat{y}|D) = \prod_{t=1}^{T} \left[p(\hat{s}_t|D, \hat{s}_{<t}) \prod_{c \in C_i} p(\hat{y}_{t,c}|D, \hat{s})\right]$.

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. 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. Our oracle identification approach relies on first identifying an oracle set of sentences and then the oracle compression options.[3]

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.

4.1 Oracle Construction

Sentence Extractive Oracle We first identify an oracle set of sentences to extract using a beam search procedure similar to 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 β. 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.

The beam search procedure returns a beam of β different sentence combinations in the final beam. We use the sentence extractive oracle for both the extraction-only model and the joint extraction-compression model.

Oracle Compression Labels To form our joint extractive and compressive oracle, we need to give the compression decisions binary labels $y_{i,c}$ in each set of extracted sentences. For simplicity and computational efficiency, we assign each sentence

[3]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.

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

Table 2: Compressibility: 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.

a single $y_{i,c}$ independent of the context it occurs in. For each compression option, we assess the value of it by comparing the ROUGE score of the sentence with and without this phrase. Any option that increases ROUGE is treated as a compression that should be applied. When calculating this ROUGE value, we remove stop words include stemming.

We run this procedure on each of our oracle extractive sentences. The fraction of positive and negative labels assigned to compression options is shown for each of the three datasets in Table 2. CNN is the most compressable dataset among CNN, DM and NYT.

ILP-based oracle construction Past work has derived oracles for extractive and compressive systems using integer linear programming (ILP) (Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011). Following their approach, we can directly optimize for ROUGE recall of an extractive or compressive summary in our framework if we specify a length limit. However, we evaluate on ROUGE F_1 as is standard when comparing to neural models that don't produce fixed-length summaries. Optimizing for ROUGE F_1 cannot be formulated as an ILP, since computing precision requires dividing by the number of selected words, making the objective no longer linear. We experimented with optimizing for ROUGE F_1 indirectly by finding optimal ROUGE recall summaries at various settings of maximum summary length. However, these summaries frequently contained short sentences to fill up the budget, and the collection of summaries returned tended to be less diverse than those found by beam search.

4.2 Learning Objective

Often, many oracle summaries achieve very similar ROUGE values. We therefore want to avoid committing to a single oracle summary for the learning process. Our procedure from Section 4.1 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 $\mathcal{L}_{\text{sent}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\log p(s_{i,t}^*|D, s_{i,<t}^*)$. 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. The oracle sentence indices are sorted according to the individual salience (ROUGE score) rather than document order.

The objective of the compression module is defined as $\mathcal{L}_{\text{comp}} = -\sum_{i=1}^{m}\sum_{c=1}^{C}\log p(y_{i,c}^*|D, \hat{s})$ 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.

5 Experiments

We evaluate our model on two axes. First, for content selection, we use ROUGE as is standard. Second, we evaluate the grammaticality of our model to ensure that it is not substantially damaged by compression.

5.1 Experimental Setup

Datasets We evaluate the proposed method on three popular news summarization datasets: the New York Times corpus (Sandhaus, 2008), CNN and Dailymail (DM) (Hermann et al., 2015).[4]

As discussed in Section 2, compression will give different results on different datasets depending on how much compression is optimal from the standpoint of reproducing the reference summaries, which changes how measurable the impact of compression is. In Table 2, we show the "compressability" of these three datasets: how valuable various compression options seem to be from the standpoint of improving ROUGE. We found that CNN has significantly more positive compression options than the other two. Critically, CNN also has the shortest references (37 words on average,

[4]More details about the experimental setup, implementation details, and human evaluation are provided in the Appendix.

Model	CNN		
	R-1	R-2	R-L
Lead (Ours)	29.1	11.1	25.8
Refresh* (Narayan et al., 2018)	30.3	**11.6**	26.9
LatSum* (Zhang et al., 2018)	28.8	11.5	25.4
BanditSum (Dong et al., 2018)	**30.7**	**11.6**	**27.4**
LEADDEDUP	29.7	10.9	26.2
LEADCOMP	30.6	10.8	27.2
EXTRACTION	30.3	11.0	26.5
EXTLSTMDEL	30.6	11.9	27.1
JECS	**32.7**	**12.2**	**29.0**

Table 3: Experimental results on the test sets of CNN. * indicates models evaluates with our own ROUGE metrics. Our model outperforms our extractive model and lead-based baselines, as well as prior work.

Model	CNNDM		
	R-1	R-2	R-L
Lead (Ours)	40.3	17.6	36.4
Refresh* (Narayan et al., 2018)	40.0	18.1	36.6
NeuSum	**41.6**	**19.0**	38.0
LatSum* (Zhang et al., 2018)	41.0	18.8	37.4
LatSum w/ Compression	36.7	15.4	34.3
BanditSum	41.5	18.7	37.6
CBDec (Jiang and Bansal, 2018)	40.7	17.9	37.1
FARS (Chen and Bansal, 2018)	40.9	17.8	**38.5**
LEADDEDUP	40.5	17.4	36.5
LEADCOMP	40.8	17.4	36.8
EXTRACTION	40.7	18.0	36.8
JECS	**41.7**	**18.5**	**37.9**

Table 4: Experimental results on the test sets of CNNDM. The portion of CNN is roughly one of tenth of DM. Gains are more pronounced on CNN because this dataset features shorter, more compressed reference summaries.

compared to 61 for Daily Mail; see Appendix). In our experiments, we first focus on CNN and then evaluate on the other datasets.

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. LEADCOMP 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 up from the document. Our extractive approach is most similar to this model. Refresh (Narayan et al., 2018), BanditSum (Dong et al., 2018) and LatSum (Zhang et al., 2018) are extractive summarization models for comparison. We also compare with some abstractive models including PointGen-Cov (See et al., 2017), FARS (Chen and Bansal, 2018) and CBDec (Jiang and Bansal, 2018).

We also compare our joint model with a pipeline model with an off-the-shelf compression module. We implement a deletion-based BiLSTM model for sentence compression (Wang et al., 2017b) and run the model on top of our extraction output.[5] The pipeline model is denoted as EXTLSTMDEL.

5.2 Results on CNN

Table 3 shows experiments results on CNN. We list performance of the LEAD baseline and 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 model achieves substantially higher performance than all baselines and past systems (+2 ROUGE F1 compared to any of these). On this dataset, compression is substantially useful. Compression is somewhat effective in isolation, as shown by the performance of LEADDEDUP and LEADCOMP. But compression in isolation still gives less benefit (on top of LEAD) than when combined with the extractive model (JECS) in the joint framework. Furthermore, our model beats the pipeline model EXTLSTMDEL which shows the necessity of training a joint model with ROUGE supervision.

5.3 Results on Combined CNNDM and NYT

We also report the results on the full CNNDM and NYT although they are less compressable. Table 4 and Table 5 shows the experimental results on these datasets.

Our models still yield strong performance compared to baselines and past work on the CNNDM

[5] We reimplemented the authors' model following their specification and matched their accuracy. For fair comparison, we tuned the deletion threshold to match the compression rate of our model; other choices did not lead to better ROUGE scores.

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.

Model	Preference (%) ↑	Error ↓	R-1 ↑
EXT LEAD3	–	22	29.1
EXTDROP	12%	161	30.2
EXTLSTMDEL	43%	**24**	30.6
JECS	**45**%	31	**32.7**

Table 6: Human preference, ROUGE and Grammarly grammar checking results. We asked Turkers to rank the models' output based on grammaticality. Error shows the number of grammar errors in 500 sentences reported by Grammarly. Our JECS model achieves the highest ROUGE and is preferred by humans while still making relatively few errors.

dataset. The EXTRACTION model achieves comparable results to past successful extractive approaches on CNNDM and JECS improves on this across the datasets. In some cases, our model slightly underperforms on ROUGE-2. 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.

On NYT, we see again that the inclusion of compression leads to improvements in both the LEAD setting as well as for our full JECS model.[6]

5.4 Grammaticality

We evaluate grammaticality of our compressed summaries in three ways. First, we use Amazon Mechanical Turk to compare different compression techniques. Second, to measure absolute grammaticality, we use an automated out-of-the-box tool Grammarly. Finally, we conduct manual analysis.

Human Evaluation We first conduct a human evaluation on the Amazon Mechanical Turk platform. We ask Turkers to rank different compression versions of a sentence in terms of grammaticality. We compare our full JECS model and the off-the-shelf pipeline model EXTLSTMDEL, which have matched compression ratios. We also propose another baseline, EXTRACTDROPOUT, which randomly drops words in a sentence to match the compression ratio of the other two models. The results are shown in Table 6. Turkers give roughly equal preference to our model and the EXTLSTMDEL model, which was learned from supervised compression data. However, our JECS model achieves substantially higher ROUGE score, indicating that it represents a more effective compression approach.

We found that absolute grammaticality judgments were hard to achieve on Mechanical Turk; Turkers' ratings of grammaticality were very noisy and they did not consistently rate true article sentences above obviously noised variants. Therefore, we turn to other methods as described in the next two paragraphs.

Automatic Grammar Checking We use Grammarly to check 500 sentences sampled from the outputs of the three models mentioned above from CNN. Both EXTLSTMDEL and JECS make a small number of grammar errors, not much higher than the purely extractive LEAD3 baseline. One major source of errors for JECS is having the wrong article after the deletion of an adjective like *an [awesome] style*.

Manual Error Analysis To get a better sense of our model's output, 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. Eight bad deletions due to parsing errors like *a UK [JJ national] from London*. 2. Eight inappropriate adjective deletions causing correctness issues with respect to the reference document like *[former] president* and *[nuclear] weapon*. 3. Three other errors: partial deletion of slang, inappropriate PP attachment deletion, and an unhandled grammatical construction:

[6] 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 and evaluation on recall only.

Reference Summary	Prediction with Compressions
Mullah Omar, the reclusive founder of the Afghan Taliban, is still in charge, a new biography claims. An ex-Taliban insider says there have been rumors that the one-eyed militant is dead.	(CNN) Mullah Mohammed Omar is "still the leader" of the Taliban's self-declared Islamic Emirate of Afghanistan. The Taliban's "Cultural Commission" released the 11-page document in several different translations on the movement's website, ostensibly to commemorate the 19th anniversary of an April 4, 1996, meeting in Afghanistan's Kandahar province when an assembly of Afghans swore allegiance to Omar.
Rebecca Francis' photo with a giraffe was shared by Ricky Gervais. Francis was threatened on Twitter for the picture. Francis, a hunter, said the giraffe was "close to death" and became food for locals.	(CNN) Five years ago, Rebecca Francis posed for a photo while lying next to a dead giraffe. The trouble started Monday, when comedian Ricky Gervais tweeted the photo with a question. Francis, who has appeared on the NBC Sports Network outdoor lifestyle show "Eye of the Hunter" and was the subject of an interview with Hunting Life in late March, responded in a statement to HuntingLife.com on Tuesday, which was posted on its Facebook page.
Frida Ghitis: President Barack Obama is right to want a deal, but this one gives Iran too much. She says the framework agreement starts lifting Iran sanctions much too soon.	(CNN) President Barack Obama tied himself to the mast of a nuclear deal with Iran even before he became the Democratic candidate for president. Reaching a good, solid agreement with Iran is a worthy, desirable goal. But the process has unfolded under the destructive influence of political considerations, weakening America's hand and strengthening Iran.

Table 7: Examples of applied compressions. The top two are sampled from among the most compressed examples in the dataset. Our JECS model is able to delete both large chunks (especially temporal PPs giving dates of events) as well as individual modifiers that aren't determined to be relevant to the summary (e.g., the specification of the *19th* anniversary). The last example features more modest compression.

students [first], athletes [second].

Examples of output are shown in Table 7. The first two examples are sampled from the top 25% of the most compressed examples in the corpus. We see a variety of compression options that are used in the first two examples, including removal of temporal PPs, large subordinate clauses, adjectives, and parentheticals. The last example features less compression, only removing a handful of adjectives in a manner which slightly changes the meaning of the summary.

Improving the parser and deriving a more semantically-aware set of compression rules can help achieving better grammaticality and readability. However, we note that such errors are largely orthogonal to the core of our approach; a more refined set of compression options could be dropped into our system and used without changing our fundamental model.

6 Compression Analysis

Compression Threshold Compression in our model is an imbalanced binary classification problem. The trained model's natural classification threshold (probability of DEL > 0.5) may not be optimal for downstream ROUGE. 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 5, where we show the average ROUGE value at different compression thresholds. 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.

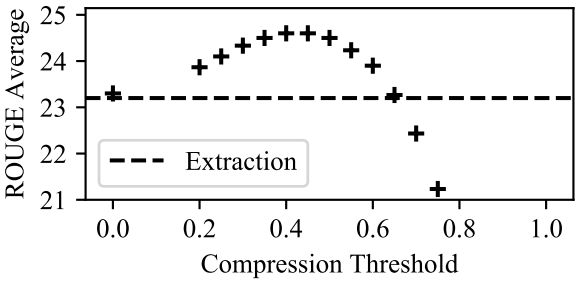

Figure 5: Effect of changing the compression threshold on CNN. The y-axis shows the average of the F1 of ROUGE-1,-2 and -L. The dotted line is the extractive baseline. The model outperforms the extractive model and achieves nearly optimal performance across a range of threshold values.

Compression Type Analysis We further break down the types of compressions used in the model. Table 8 shows the compressions that our model ends up choosing at test time. PPs are often compressed by the deduplication mechanism because the compressible PPs tend to be temporal and location adjuncts, which may be redundant across sen-

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 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.

tences. 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. More details about compression options are in the Supplementary Material.

7 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, so additional improvements to extractive modeling might be expected to stack with our approach.

Syntactic Compression Prior to the explosion of neural models for summarization, syntactic compression (Martins and Smith, 2009; Woodsend and Lapata, 2011) was relatively more common. 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. For this work, we find that relying on syntax gives us more easily understandable and controllable compression options.

Contemporaneously with our work, Mendes et al. (2019) explored an extractive and compressive approach using compresson integrated into the decoder

8 Conclusion

In this work, we presented a neural network framework 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 compression 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 (Keahey et al., 2019). Thanks as well to the anonymous reviewers for their helpful comments.

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 Abstrac-

tive 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.

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.

Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. 2019. Chameleon: a scalable production testbed for computer science research. In *Contemporary High Performance Computing: From Petascale toward Exascale*, 1 edition, volume 3 of *Chapman & Hall/CRC Computational Science*, chapter 5, pages 123–148. CRC Press, Boca Raton, FL.

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 Jessy 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.

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.

Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F. T. Martins, and Shay B. Cohen. 2019. Jointly Extracting and Compressing Documents with Summary State Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

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.

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. 2017a. 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.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017b. 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, Vancouver, Canada. 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.