Neural Abstractive Summarization with Structural Attention

Tanya Chowdhury\textsuperscript{1}, Sachin Kumar\textsuperscript{2} and Tanmoy Chakraborty\textsuperscript{1}
\textsuperscript{1}IIIT-Delhi, India
\textsuperscript{2}Carnegie Mellon University, USA
\{tanya14109,tanmoy\}@iiitd.ac.in, sachink@andrew.cmu.edu

Abstract

Attentional, RNN-based encoder-decoder architectures have achieved impressive performance on abstractive summarization of news articles. However, these methods fail to account for long term dependencies within sentences of a document. This problem is exacerbated in multi-document summarization tasks such as summarizing the popular opinion in threads present in community question answering (CQA) websites such as Yahoo! Answers and Quora. These threads contain answers which often overlap or contradict each other. In this work, we present a hierarchical encoder based on structural attention to model such inter-sentence and inter-document dependencies. We set the popular pointer-generator architecture and some of the architectures derived from it as our baselines and show that they fail to generate good summaries in a multi-document setting. We further illustrate that our proposed model achieves significant improvement over the baselines in both single and multi-document summarization settings – in the former setting, it beats the best baseline by 1.31 and 7.8 ROUGE-1 points on CNN and CQA datasets, respectively; in the latter setting, the performance is further improved by 1.6 ROUGE-1 points on the CQA dataset.

1 Introduction

Sequence-to-sequence (seq2seq) architectures with attention have led to tremendous success in many conditional language generation tasks such as machine translation [Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015], dialog generation [Vinyals and Le, 2015], abstractive summarization [Rush et al., 2015; See et al., 2017; Paulus et al., 2018] and question answering [Yin et al., 2016]. Attention models that compute a context vector based on the entire input text at every decoder step have especially benefited abstractive summarization. This is because – (1) LSTM encoders struggle to encode long documents into just one vector from which one can decode the summary, and (2) generating summary involves a lot of copying from the source text which attention models easily facilitate that can also be made part of the training process [See et al., 2017; Nallapati et al., 2016].

Studies in abstractive summarization majorly focus on generating news summaries, and the CNN/DailyMail dataset [Hermann et al., 2015; Nallapati et al., 2016] is largely used as a benchmark. This is a very clean dataset with fluent and properly structured text collected from a restricted domain, and it hardly features any repetitive information. Therefore, the current models which have been built on this dataset do not need to account too much for repeated or contradicting information. Additionally, these models are trained on truncated documents (up to 400 words), and increasing the word length leads to a drop in performance of these models [See et al., 2017]. This is because either LSTM is unable to encode longer documents accurately or in this dataset, the first few sentences contain most of the summary information. However, real-world documents are usually much longer, and their summary information is spread across the entire document rather than in the first few sentences. In this paper, we explore one such dataset dealing with community question answering (CQA) summarization.

Multi-document summarization (MDS) is a well-studied problem with possible applications in summarizing tweets [Cao et al., 2016], news from varying sources [Yasunaga et al., 2017], reviews in e-commerce services [Ganesan et al., 2010], and so on. CQA services such as Quora, Yahoo! Answers, Stack Overflow help in curating information which may often be difficult to obtain directly from the existing resources on the web. However, a majority of question threads on these services harbor unstructured, often repeating or contradictory answers. Additionally, the answers vary in length from being a few words to a thousand words long. A large number of users with very different writing styles contribute to form these knowledge bases, which results in a diverse and challenging corpora. We define the task of CQA summarization as follows – given a list of such question threads, summarize the popular opinion reflected in them. This can be viewed as a multi-document summarization task [Chowdhury and Chakraborty, 2019].

We envision a solution to this problem by incorporating structural information present in language in the summarization models. Recent studies have shown promise in this direction. For example, [Fernandes et al., 2018] add graph neural networks (GNNs) [Li et al., 2015] to the seq2se en-
encoder model for neural abstractive summarization. [Liu et al., 2019] model documents as multi-root dependency trees, each node representing a sentence; and pick the induced tree roots to be summary sentences in an extractive setting. Recently, [Song et al., 2018] attempt to preserve important structural dependency relations obtained from syntactic parse trees of the input text in the summary by using a structure infused copy mechanism. However, obtaining labeled data to explicitly model such dependency relations is expensive. Here, we attempt to mitigate this issue by proposing a structural encoder for summarization based on the prior work on structure attention networks [Liu and Lapata, 2018] which implicitly incorporates structural information within end-to-end training.

Our major contributions in this work are three-fold:

1. We enhance the pointer-generator architecture by adding a structural attention based encoder to implicitly capture long term dependency relations in summarization of lengthy documents.

2. We further propose a hierarchical encoder with multi-level structural attention to capture document-level discourse information in the multi-document summarization task.

3. We introduce multi-level contextual attention in the structural attention setting to enable word level copying and to generate more abstractive summaries, compared to similar architectures.

We compare our proposed solution against the popular pointer-generator model [See et al., 2017] and a few other summarization models derived from it in both single and multi-document settings. We show that our structural encoder architecture beats the strong pointer-generator baseline by 1.31 ROUGE-1 points on the CNN/Dailymail dataset and by 7.8 ROUGE-1 points on the concatenated CQA dataset for single document summarization (SDS). Our hierarchical structural encoder architecture further beats the concatenated approach by another 1.6 ROUGE-1 points on the CQA dataset. A qualitative analysis of the generated summaries observe considerable qualitative gains after inclusion of structural attention. Our structural attention based summarization model is one of the few abstractive approaches to beat extractive baselines in MDS. The code is public at https://bit.ly/35f7q93.

2 Models

In this section, we first describe the seq2seq based pointer-generator architecture proposed by [See et al., 2017; Nallapati et al., 2016]. We then elaborate on how we incorporate structural attention into the mechanism and generate non-projective dependency trees to capture long term dependencies within documents. Following this, we describe our hierarchical encoder architecture and propose multi-level contextual attention to better model discourse for MDS.

2.1 Pointer-Generator Model

The pointer-generator architecture (PG) [See et al., 2017] serves as the starting point for our approach. PG is inspired from the LSTM encoder-decoder architecture [Nallapati et al., 2016] for the task of abstractive summarization. Tokens are fed to a bidirectional encoder to generate hidden representations $h$, and a unidirectional decoder is used to generate the summary. At the decoding step $t$, the decoder uses an attention mechanism [Bahdanau et al., 2015] to compute an attention vector $a^t$ for every encoder hidden state. These attention vectors are pooled to compute a context vector $c_t$ for every decoder step. The context vector $c_t$ at that step along with the hidden representations $h$ are passed through a feedforward network followed by softmax to give a multinomial distribution over the vocabulary $p_{vocab}(w)$.

Additionally, [See et al., 2017] introduce a copy mechanism [Vinyals et al., 2015] in this model. At each decoding step $t$, it predicts whether to generate a new word or copy the word based on the attention probabilities $a^t$ using a probability value $p_{gen}$. This probability is a function on the decoder input $x_t$, context vector $c_t$, and decoder state $s_t$. $p_{gen}$ is used as a soft switch to choose between copying and generation as shown below:

$$p(w) = p_{gen}p_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i = w} a^t_i$$

This model is trained by minimizing the negative log-likelihood of the predicted sequence normalized by the sequence length: $L = -\frac{1}{T} \sum_{t=0}^{T} \log p(w^*_t)$, where $T$ is the length of the generated summary. Once $L$ gets converged, an additional coverage loss is used to train the model further, which aims to minimize repetitions in the generated summaries. We refer the readers to [See et al., 2017] for more details.

2.2 Structural Attention

We observe that while the pointer-generator architecture performs well on the news corpora, it fails to generate meaningful summaries on complex datasets. We hypothesize that taking into account intra-document discourse information will help in generating better summaries. We propose an architecture that implicitly incorporates such information using document structural representation [Liu and Lapata, 2018; Kim et al., 2017] to include richer structural dependencies in the end-to-end training.

Proposed Model

We model our document as a non-projective dependency parse tree by constraining inter-token attention as weights of the dependency tree. We use the Matrix tree theorem [Koo et al., 2007; Tutte, 1984] to carry out the same.

As shown in Figure 1, we feed our input tokens $(w_i)$ to a bi-LSTM encoder to obtain hidden state representations $h_i$. We decompose the hidden state vector into two parts: $d_i$ and $e_i$, which we call the structural part and the semantic part, respectively:

$$[e_i, d_i] = h_i$$

For every pair of two input tokens, we transform their structural parts $d_i$ and try to compute the probability of a parent-child relationship edge between them in the dependency tree. For tokens $j$ and $k$, this is done as:

$$u_j = \tanh(W_p d_j); \quad u_k = \tanh(W_c d_k)$$
The structural attention model, while efficient, requires 2.3 Hierarchical Model

When a document with K tokens, f is a K × K matrix representing inter-token attention. We model each token as a node in the dependency tree and define the probability of an edge between tokens at positions j and k, \( P(z_{jk} = 1) \), which is given as,

\[
A_{jk} = \begin{cases} 
0 & \text{if } j = k \\
\exp(f_{jk}) & \text{otherwise}
\end{cases}
\]

\[
L_{jk} = \begin{cases} 
\sum_{j'=1}^{K} A_{j'k} & \text{if } j = k \\
-A_{jk} & \text{otherwise}
\end{cases}
\]

\[
f_{jk} = W_{r}d_{j}
\]

\[
\bar{L}_{jk} = \begin{cases} 
\exp(f_{jk}) & j = 1 \\
L_{jk} & j > 1
\end{cases}
\]

\[
P(z_{jk} = 1) = (1 - \delta(j, k))A_{jk}L_{k-1}^{-1} - (1 - \delta(j, 1))A_{jk}L_{k-1}^{-1}
\]

where \( \delta(x, y) = 1 \) when \( x = y \). We denote \( P(z_{jk} = 1) \) by \( a_{jk} \) (structural attention). Let \( a_{j}^{t} \) be the probability of the \( j \)th token to be the root:

\[
a_{j}^{t} = \exp(W_{r}d_{j})\bar{L}_{j}^{-1}
\]

We use this (soft) dependency tree formulation to compute a structural representation \( r \) for each encoder token as,

\[
s_{i} = a_{i}^{t}e_{\text{root}} + \sum_{k=1}^{n} a_{ik}e_{k}; c_{i} = \sum_{k=1}^{n} a_{ik}e_{i}
\]

\[
r_{i} = \tanh(W_{r}[e_{i}, s_{i}, c_{i}])
\]

Thus, for encoder step \( i \), we now obtain a structure infused hidden representation \( r_{i} \). We then compute the contextual attention for each decoding time step \( t \) as,

\[
e_{\text{struct}}^{t} = v^{t}\tanh(W_{r}r_{i} + W_{s}s_{i} + b_{\text{attn}})
\]

\[
a_{\text{struct}}^{t} = \text{softmax}(e_{\text{struct}}^{t})
\]

Now, using \( a_{\text{struct}}^{t} \), we can compute a context vector similar to standard attentional model by weighted sum of the hidden state vectors as \( c_{\text{struct}}^{t} = \sum_{i=1}^{t} a_{\text{struct}}^{t}h_{i} \). At every decoder time step, we also compute the basic contextual attention vector \( a^{t} \) (without structure incorporation), as discussed previously. We use \( c_{\text{struct}}^{t} \) to compute \( P_{\text{vocab}} \) and \( p_{\text{gen}} \). We, however, use the initial attention distribution \( a_{i} \) to compute \( P(w) \) in order to facilitate token level pointing.

### 2.3 Hierarchical Model

The structural attention model, while efficient, requires \( O(K^{2}) \) memory to compute attention, where \( K \) is the length of the input sequence making it memory intensive for long documents. Moreover, in CQA, answers reflect different opinions of individuals. In such a case, concatenating answers results in user specific information loss. To model discourse structure better in case of conflicting (where one answer contradicts other), overlapping or varying opinion, we introduce a hierarchical encoder model based on structural attention (Figure 2). We feed each answer independently to a bi-LSTM encoder and obtain token level hidden representations \( h_{idx, idx} \) where \( h_{idx} \) is the document (or answer) index and \( idx \) is the token index. We then transform these representations into structure infused vectors \( r_{idx, idx} \) as described previously. For each answer in the CQA question thread, we pool the token representations to obtain a composite answer vector \( r_{idx} \). We consider three types of pooling: average, max and sum, out of which sum pooling performs the best after initial experiments. We feed these structure infused answer embeddings to an answer-level bi-LSTM encoder and obtain higher-level encoder hidden states \( h_{idx} \) from which we calculate structure infused embeddings \( g_{idx} \). At the decoder time step \( t \), we calculate contextual attention at answer as well as token level as follows:

\[
e_{\text{ans}}^{t}_{idx} = v^{t}\tanh(W_{g}g_{idx} + W_{s}s_{t} + b_{\text{attn}})
\]

\[
a_{\text{ans}}^{t} = \text{softmax}(e_{\text{ans}}^{t})
\]

\[
e_{\text{token}}^{t}_{idx, idx} = v^{t}\tanh(W_{h}h_{idx, idx} + W_{s}s_{t} + b_{\text{attn}})
\]

\[
a_{\text{token}}^{t} = \text{softmax}(e_{\text{token}}^{t})
\]

We use the answer-level attention distribution \( a_{\text{ans}}^{t} \) to compute the context vector at every decoder time step which we use to calculate \( p_{\text{vocab}} \) and \( p_{\text{gen}} \) as described before.

To enable copying, we use \( a_{\text{token}}^{t} \). The final probability of predicting word \( w \) is given by,

\[
p(w) = p_{\text{gen}}p_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{t: w_{t} = w} a_{\text{token}}^{t}
\]

### 3 Datasets

We primarily use two datasets to evaluate the performance:
datasets. (ii) We also use the CQA dataset\cite{Chowdhury and Chakraborty, 2019} which is generated by filtering the Yahoo! Answers L6 corpora to find question threads where the best answer can serve as a summary for the remaining answers. Since Yahoo! Answers is an unstructured and unmoderated question-answer repository, this has turned out to be a challenging summarization dataset\cite{Chowdhury and Chakraborty, 2019}. Additionally, we include analysis on MultiNews\cite{Fabri et al., 2019}, a news based MSD corpora to aid similar studies. It is the first large scale MSD news dataset consisting of 56,216 article summary pairs, crowd-sourced from various different news websites.

4 Competing Methods

We compare the performance of the following SDS and MDS models.

- **Lead3**: It is an extractive baseline where the first 100 tokens of the document (in case of SDS datasets) and concatenated ranked documents (in case of MDS datasets) are picked to form the summary.
- **KL-Summ**: It is an extractive summarization method introduced by\cite{Haghighi and Vanderwende, 2009} that attempts to minimize KL-Divergence between candidate documents and generated summary.
- **LexRank**: It is an unsupervised extractive summarization method\cite{Erkan and Radev, 2004}. A graph is built with sentences as vertices, and edge weights are assigned based on sentence similarity.
- **TextRank**: It is an unsupervised extractive summarization method which selects sentences such that the information being disseminated by the summary is as close as possible to the original documents\cite{Mihalcea and Tarau, 2004}.
- **Pointer-Generator (PG)**: It is a supervised abstractive summarization model\cite{See et al., 2017}, as discussed earlier. It is a strong and popularly used baseline for summarization.
- **Pointer-Generator + Structure Infused Copy (PG+SC)**: Our implementation is similar to one of the methods proposed by\cite{Song et al., 2018}. We explicitly compute the dependency tree of sentences and encode a structure vector based on features like POS tag, number of incoming edges, depth of the tree, etc. We then concatenate this structural vector for every token to its hidden state representation in pointer-generator networks.
- **Pointer-Generator+MMR (PG+MMR)**: It is an abstractive MSD model\cite{Lebanoff et al., 2018} trained on the CNN/Dailymail dataset. It combines Maximal Marginal Relevance (MMR) method with pointer-
Table 1: Parameters common to all PG-based models.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocabulary size</td>
<td>50,000</td>
</tr>
<tr>
<td>Input embedding dim</td>
<td>128</td>
</tr>
<tr>
<td>Training decoder steps</td>
<td>100</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.15</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adagrad</td>
</tr>
<tr>
<td>Adagrad init accumulator</td>
<td>0.1</td>
</tr>
<tr>
<td>Max gradient norm (for clipping)</td>
<td>2.0</td>
</tr>
<tr>
<td>Max decoding steps (for BS decoding)</td>
<td>120</td>
</tr>
<tr>
<td>Min decoding steps (for BS decoding)</td>
<td>35</td>
</tr>
<tr>
<td>Beam search width</td>
<td>1</td>
</tr>
<tr>
<td>Weight of coverage loss</td>
<td>R-L</td>
</tr>
<tr>
<td>GPU</td>
<td>GeForce 2080 Ti</td>
</tr>
</tbody>
</table>

Table 2: Performance on two versions of the CNN/Dailymail dataset based on ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L) scores.

<table>
<thead>
<tr>
<th>Method</th>
<th>CNN/Dailymail-400</th>
<th>CNN/Dailymail-800</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-1</td>
<td>R-2</td>
</tr>
<tr>
<td>Lead3</td>
<td>40.34</td>
<td>17.70</td>
</tr>
<tr>
<td>KL-Summ</td>
<td>30.50</td>
<td>11.31</td>
</tr>
<tr>
<td>LexRank</td>
<td>34.12</td>
<td>13.31</td>
</tr>
<tr>
<td>TexiRank</td>
<td>31.38</td>
<td>12.29</td>
</tr>
<tr>
<td>PG</td>
<td>39.53</td>
<td>17.28</td>
</tr>
<tr>
<td>PG+SC</td>
<td>39.82</td>
<td>17.68</td>
</tr>
<tr>
<td>PG+Transformers</td>
<td>39.94</td>
<td>36.44</td>
</tr>
<tr>
<td>PG+SA</td>
<td>40.02</td>
<td>17.88</td>
</tr>
</tbody>
</table>

Table 3: Performance on the CQA and MultiNews datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>CQA/Summ</th>
<th>MultiNews</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-1</td>
<td>R-2</td>
</tr>
<tr>
<td>Lead3</td>
<td>8.7</td>
<td>1.0</td>
</tr>
<tr>
<td>KL-Summ</td>
<td>24.4</td>
<td>4.3</td>
</tr>
<tr>
<td>LexRank</td>
<td>28.4</td>
<td>4.7</td>
</tr>
<tr>
<td>TexiRank</td>
<td>27.8</td>
<td>4.8</td>
</tr>
<tr>
<td>PG</td>
<td>23.2</td>
<td>4.0</td>
</tr>
<tr>
<td>PG+SC</td>
<td>25.5</td>
<td>4.2</td>
</tr>
<tr>
<td>PG+MMR</td>
<td>16.2</td>
<td>3.2</td>
</tr>
<tr>
<td>Hi-MAP</td>
<td>30.9</td>
<td>4.69</td>
</tr>
<tr>
<td>PG+SA</td>
<td>29.4</td>
<td>4.9</td>
</tr>
<tr>
<td>PG+HSA</td>
<td>31.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

5 Experimental Results

5.1 Quantitative Analysis

We compare the performance of the models on the basis of ROUGE-1, 2 and L F1-scores [Lin, 2004] on the CNN/Dailymail (Table 2), CQA and Multinews (Table 3) datasets. We observe that infusion of structural information leads to considerable gains over the basic PG architecture on both the datasets. Our approach fares better than explicitly incorporating structural information [Song et al., 2018]. The effects of incorporating structure is more significant in the CQA dataset (+7.8 ROUGE-1 over PG) as compared to the CNN/Dailymail dataset (+0.49 ROUGE-1 over PG). The benefit is comparatively prominent in CNN/Dailymail-800 (+1.31 ROUGE-1 over PG) as compared to CNN/Dailymail-400 (+0.49 ROUGE-1 over PG).

5.2 Qualitative Analysis

We ask human evaluators\(^3\) to compare CQASUMM summaries generated by competing models on the ground of content and readability. Here we observe a significant gain with structural attention. PG has difficulty in summarizing articles with repetitive information and tends to assign lower priority to facts repeating across answers. Methods like LexRank, on the other hand, tend to mark these facts the most important. Incorporating structure solves this problem to some extent (due to the pooling operations). We find that PG sometimes picks sentences containing opposing opinions for the same summary. We find this occurrence to be less frequent in the structural attention models. This phenomenon is illustrated with an instance from the CQA dataset in Table 4.

5.3 Model Diagnosis and Discussion

Tree depth. The structural attention mechanism has a tendency to induce shallow parse trees with high tree width. This leads to highly spread out trees especially at the root.

MDS baselines. While supervised abstractive methods like PG significantly outperform unsupervised non-neural methods in SDS, we find them to be inferior or similar in performance in MDS. This has also been reported by recent MDS related studies such as [Lebanoff et al., 2018] where LexRank is shown to significantly beat basic PG on the DUC-04 and TAC-11 datasets. This justifies the choice of largely unsupervised baselines in recent MDS related studies such as [Yasunaga et al., 2017; Lebanoff et al., 2018].

Dataset layout. We observe that while increasing the length of the input documents lowers ROUGE score for the CNN/Dailymail dataset, it boosts the same in the CQA dataset. This might be attributed to the difference in information layout in the two datasets. The CNN/Dailymail has a very high Lead-3 score signifying summary information concentration within the first few lines of the document.

Coverage with structural attention. Training a few thousand iterations with coverage loss is known to significantly reduce repetition. However, in the structural attention models, while training with the CQA dataset we observe that soon

\(^3\)The evaluators were experts in NLP, and their age ranged between 20-30 years.
after coverage loss has converged, on further training the repetition in generated summaries starts increasing again. Future work would include finding a more appropriate coverage function for datasets with repetition.

6 Related Work

Neural Abstractive Summarization

PG-based models. Since the studies conducted by [Nallapati et al., 2016; See et al., 2017], many approaches to neural abstractive summarization have extended them in different ways. [Cohan et al., 2018] extend the approach proposed by [See et al., 2017] with a hierarchical encoder to model discourse between various sections of a scientific paper using PubMed dataset.

MDS. Most studies on MDS were performed during DUC (2001-2007) and TAC (2008-2011) workshops. Opinosis [Ganesan et al., 2010] uses word-level opinion graphs to find cliques and builds them into summaries in highly redundant opinion sentences. Recently, [Lebanoff et al., 2018] adapt the single document architecture to a multi-document setting by using maximal marginal relevance method to select representative sentences from the input and feed to the encoder-decoder architecture. [Nema et al., 2017] propose a query based abstractive summarization approach where they first encode both the query and the document, and at each decoder time step, calculate attention based on both.

Structure Infused Document Modeling

There have been numerous studies to incorporate structure in document representations by adding syntactic parsing to pipeline like architectures. However, external parser computation is costly and does not scale well. [Tang et al., 2015a; Tang et al., 2015b] are one of the first to obtain document representations by first computing sentence representations and hierarchically aggregating them. [Yang et al., 2016] propose a model to implicitly add structural information within end-to-end training by assigning attention scores to each sentence according to the context at that state. [Kim et al., 2017] encoded structural information as graphical models within deep neural networks to facilitate end-to-end training. They use two graphical attention structures – linear chain conditional random fields and graph-based parsing models, and show how both of them can be implemented as layers of a neural network. Recent work by [Balachandran et al., 2020] proposes a similar idea by incorporating both latent and explicit sentence dependencies into single document summarization architecture. They further explicitly induce structure by injecting a coreferencing mention graph across sentences.

7 Conclusion

In this work, we proposed an approach to incorporate structural attention within end-to-end training in summarization networks. We achieved a considerable improvement in terms of ROUGE scores compared to our primary baseline model on both the CNN/Dailymail and CQA datasets. We also introduced multi-level contextual attention which helped in generating more abstract summaries. The analysis hinted that incorporation of some form of structural attention might be the key to achieve significant improvement compared to the extractive counterpart in the complex multi-document summarization task.

References


