SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents

Ramesh Nallapati, Feifei Zhai,* Bowen Zhou
nallapati@us.ibm.com, ffzhai2012@gmail.com, zhou@us.ibm.com
IBM Watson
1011 Kitchawan Road, Yorktown Heights, NY 10598

Abstract
We present SummaRuNNer, a Recurrent Neural Network (RNN) based sequence model for extractive summarization of documents and show that it achieves performance better than or comparable to state-of-the-art. Our model has the additional advantage of being very interpretable, since it allows visualization of its predictions broken up by abstract features such as information content, salience and novelty. Another novel contribution of our work is abstractive training of our extractive model that can train on human generated reference summaries alone, eliminating the need for sentence-level extractive labels.

1 Introduction
Document summarization is an important problem that has many applications in information retrieval and natural language understanding. Summarization techniques are mainly classified into two categories: extractive and abstractive. Extractive methods aim to select salient snippets, sentences or passages from documents, while abstractive summarization techniques aim to concisely paraphrase the information content in the documents.

A vast majority of the literature on document summarization is devoted to extractive summarization. Traditional methods for extractive summarization can be broadly classified into greedy approaches (e.g., (Carbonell and Goldstein 1998)), graph based approaches (e.g., (Radev and Erkan 2004)) and constraint optimization based approaches (e.g., (McDonald 2007)).

Recently, neural network based approaches have become popular for extractive summarization. For example, (Kageback et al. 2014) employed the recursive autoencoder (Socher et al. 2011) to summarize documents, producing best performance on the Opinosis dataset (Ganesan, Zhai, and Han 2010). (Yin and Pei 2015) applied Convolutional Neural Networks (CNN) to project sentences to continuous vector space and then select sentences by minimizing the cost based on their ‘prestige’ and ‘diverseness’, on the task of multi-document extractive summarization. Another related work is that of (Cao et al. 2016), who address the problem of query-focused multi-document summarization using CNNs, where they use weighted-sum pooling over sentence representations to represent documents. The weights are learned from attention over sentence representations based on the query.

Recently, with the emergence of strong generative neural models for text (Bahdanau, Cho, and Bengio 2014), abstractive techniques are also becoming increasingly popular. For example, (Rush, Chopra, and Weston 2015) proposed an attentional feed-forward network for abstractive summarization of sentences into short headlines. Further developing on their work, (Nallapati, Zhou, and Xiang 2016) propose a set of recurrent neural network based encoder-decoder models that focus on various aspects of summarization like handling out-of-vocabulary words and modeling syntactic features of words in the sentence. In a follow-up work (Nallapati et al. 2016), they also propose abstractive techniques for summarization of large documents into multi-sentence summaries, using the CNN/DailyMail corpus.

Despite the emergence of abstractive techniques, extractive techniques are still attractive as they are less complex, less expensive, and generate grammatically and semantically correct summaries most of the time. In a very recent work, Cheng and Lapata (2016) proposed an attentional encoder-decoder for extractive single-document summarization and applied to the CNN/Daily Mail corpus.

Like (Cheng and Lapata 2016), our work also focuses only on sentential extractive summarization of single documents using neural networks. We use the same corpus used by (Nallapati et al. 2016) and (Cheng and Lapata 2016) for our experiments, since its large size makes it attractive for training deep neural networks such as ours, with several thousands of parameters.

Our main contributions are as follows: (a) we propose SummaRuNNer, a simple recurrent network based sequence classifier that outperforms or matches state-of-the-art models for extractive summarization; (b) the simple formulation of our model facilitates interpretable visualization of its decisions; and (c) we present a novel training mechanism that allows our extractive model to be trained end-to-end using abstractive summaries.

*Work was done while the author was an employee at IBM.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

1https://github.com/deepmind/rc-data
2 SummaRuNNer

In this work, we treat extractive summarization as a sequence classification problem wherein, each sentence is visited sequentially in the original document order and a binary decision is made (taking into account previous decisions made) in terms of whether or not it should be included in the summary. We use a GRU based Recurrent Neural Network (Chung et al. 2014) as the basic building block of our sequence classifier. A GRU-RNN is a recurrent network with two gates, \( u \) called the update gate and \( r \), the reset gate, and can be described by the following equations:

\[
\begin{align*}
    u_j &= \sigma(W_{ux}x_j + W_{ah}h_{j-1} + b_u) \\
    r_j &= \sigma(W_{rx}x_j + W_{rh}h_{j-1} + b_r) \\
    h'_j &= \tanh(W_{hx}x_j + W_{hh}(r_j \odot h_{j-1}) + b_h) \\
    h_j &= (1 - u_j) \odot h'_j + u_j \odot h_{j-1}
\end{align*}
\]

where the \( W \)'s and \( b \)'s are the parameters of the GRU-RNN and \( h_j \) is the real-valued hidden-state vector at timestep \( j \) and \( x_j \) is the corresponding input vector, and \( \odot \) represents the Hadamard product.

Our model consists of a two-layer bi-directional GRU-RNN, whose graphical representation is presented in Figure 1. The first layer of the RNN runs at the word level, and computes hidden state representations at each word position sequentially, based on the current word embeddings and the previous hidden state. We also use another RNN at the word level that runs backwards from the last word to the first, and we refer to the pair of forward and backward RNNs as a bi-directional RNN. The model also consists of a second layer of bi-directional RNN that runs at the sentence-level and accepts the average-pooled, concatenated hidden states of the bi-directional word-level RNNs as input. The hidden states of the second layer RNN encode the representations of the sentences in the document. The representation of the entire document is then modeled as a non-linear transformation of the average pooling of the concatenated hidden states of the bi-directional sentence-level RNN, as shown below.

\[
d = \tanh(W_d \frac{1}{N_d} \sum_{j=1}^{N_d} [h^f_j, h^b_j] + b),
\]

where \( h^f_j \) and \( h^b_j \) are the hidden states corresponding to the \( j \)th sentence of the forward and backward sentence-level RNNs respectively, \( N_d \) is the number of sentences in the document and \( [\cdot, \cdot] \) represents vector concatenation.

For classification, each sentence is revisited sequentially in a second pass, where a logistic layer makes a binary decision as to whether that sentence belongs to the summary, as shown below.

\[
P(y_j = 1|h_j, s_j, d) = \sigma(W_c h_j + h^f_d W_d d) # (content) \\
- h^b_d W_r \tanh(s_j) # (novelty) \\
+ W_{ap} p^a_j # (abs. pos. imp.) \\
+ W_{rp} p^r_j # (rel. pos. imp.) \\
+ b), # (bias term)
\]

where \( y_j \) is a binary variable indicating whether the \( j \)th sentence is part of the summary, \( h_j \), the representation of the sentence is given by a non-linear transformation of the concatenated hidden states at the \( j \)th time step of the bidirectional sentence-level RNN, and \( s_j \) is the dynamic representation of the summary at the \( j \)th sentence position, given by:

\[
s_j = \sum_{i=1}^{j-1} h_i P(y_i = 1|h_i, s_i, d).
\]

In other words, the summary representation is simply a running weighted summation of all the sentence-level hidden states visited till sentence \( j \), where the weights are given by their respective probabilities of summary membership.

In Eqn. (6), the term \( W_r h_j \) represents the information content of the \( j \)th sentence, \( h^f_d W_d d \) denotes the salience of the sentence with respect to the document, \( h^b_d W_r \tanh(s_j) \) captures the redundancy of the sentence with respect to the current state of the summary\(^2\), while the next two terms model the notion of the importance of the absolute and relative position of the sentence with respect to the document.\(^3\) We consider \( p^a \) and \( p^r \), the absolute and relative positional embeddings respectively, as model parameters as well.

\(^2\)We squash the summary representation using the tanh operation so that the magnitude of summary remains the same for all time-steps.

\(^3\)The absolute position denotes the actual sentence number, whereas the relative position refers to a quantized representation that divides each document into a fixed number of segments and computes the segment ID of a given sentence.
We minimize the negative log-likelihood of the observed labels at training time.

\[
l(W, b) = - \sum_{d=1}^{N_d} \sum_{j=1}^{N} (y_j^d \log P(y_j^d = 1|\mathbf{s}_j^d, \mathbf{d}_d) + (1 - y_j^d) \log(1 - P(y_j^d = 1|\mathbf{s}_j^d, \mathbf{d}_d)))
\]

where \(x\) is the document representation and \(y\) is the vector of its binary summary labels. At test time, the model emits probability of summary membership \(P(y_j)\) at each sentence sequentially, which is used as the model’s soft prediction of the extractive summary.

2.1 Extractive Training

In order to train our extractive model, we need ground truth in the form of sentence-level binary labels for each document, representing their membership in the summary. However, most summarization corpora only contain human written abstractive summaries as ground truth. To solve this problem, we use an unsupervised approach to convert the abstractive summaries to extractive labels. Our approach is based on the idea that the selected sentences from the document should be the ones that maximize the Rouge score with respect to gold summaries. Since it is computationally expensive to find a globally optimal subset of sentences that maximizes the Rouge score, we employ a greedy approach, where we add one sentence at a time incrementally to the summary, such that the Rouge score of the current set of selected sentences is maximized with respect to the entire gold summary. We stop when none of the remaining candidate sentences improves the Rouge score upon addition to the current summary set. We return this subset of sentences as the extractive ground-truth, which is used to train our RNN based sequence classifier.

2.2 Abstractive Training

In this section, we propose a novel training technique to train SummaRuNNer abstractively, thus eliminating the need to generate approximate extractive labels. To train SummaRuNNer using reference summaries, we couple it with an RNN decoder that models the generation of abstractive summaries at training time only. The RNN decoder uses the summary representation at the last time-step of SummaRuNNer as context, which modifies Eqs. 1 through 3 as follows:

\[
\begin{align*}
u_k &= \sigma(W'_{ux}x_k + W'_{uh}h_{k-1} + W'_{us}s_{-1} + b'_u) \\
r_k &= \sigma(W'_{rx}x_k + W'_{rh}h_{k-1} + W'_{rs}s_{-1} + b'_r) \\
h'_k &= \tanh(W'_{hx}x_k + W'_{hh}(r_k \odot h_{k-1}) + W'_{hs}h_{k-1} + b'_h) \\
\end{align*}
\]

where \(s_{-1}\) is the summary representation as computed at the last sentence of the sentence-level bidirectional RNN of SummaRuNNer as shown in Eq. 7. The parameters of the decoder are distinguished from those of SummaRuNNer using the ‘prime’ notation, and the time-steps of the decoder use index \(k\) to distinguish word positions in the summary from sentence indices \(j\) in the original document. For each time-step of the decoder, the embedding of the word from the previous time-step is treated as its input \(x_k\).

Further, the decoder is equipped with a soft-max layer to emit a word at each time-step. The emission at each time-step is determined by a feed-forward layer \(f\) followed by a softmax layer that assigns \(p_s\), probabilities over the entire vocabulary at each time-step, as shown below.

\[
f_k = \text{tanh}(W'_f h_k + W'_f x_k + W'_f s_{-1} + b'_f) \\
P_\psi(w_k) = \text{softmax}(W'_v f_k + b'_v)
\]

Instead of optimizing the log-likelihood of the extractive ground truth as shown in Eq. 8, we minimize the negative log-likelihood of the words in the reference summary as follows.

\[
l(W, b, W', b') = - \sum_{k=1}^{N_s} \log(P_\psi(w_k))
\]

where \(N_s\) is the number of words in the reference summary. At test time, we uncouple the decoder from SummaRuNNer and emit only the sentence-level extractive probabilities \(p_s(y_j)\) of Eq. 6.

Intuitively, since the summary representation \(s_{-1}\) acts as the only information channel between the SummaRuNNer model and the decoder, maximizing the probability of abstractive summary words as computed by the decoder will require the model to learn a good summary representation which in turn depends on accurate estimates of extractive probabilities \(p_s(y_j)\).

3 Related Work

Treating document summarization as a sequence classification model has been considered by earlier researchers. For example, (Shen et al. 2007) used Conditional Random Fields to binary-classify sentences sequentially. Our approach is different from theirs in the sense that we use RNNS in our model that do not require any handcrafted features for representing sentences and documents.

Since the sequence classifier requires sentence-level summary membership labels to train on, we used a simple greedy approach to convert the abstractive summaries to extractive labels. Similar approaches have been employed by other researchers such as (Svore, Vanderwende, and Burges 2007). Further, recently (Cao et al. 2015) propose an ILP based approach to solve this problem optimally.

Most single-document summarization datasets available for research such as DUC corpora are not large enough to train deep learning models. Two recent papers ((Nallapati et al. 2016) and (Cheng and Lapata 2016)) solve this problem by proposing a new corpus based on news stories from CNN and Daily Mail that consist of around 280,000 documents and human generated summaries. Of these, the work of (Cheng and Lapata 2016) is the closest to our work since they also employ an extractive approach for summarization. Their model is based on an encoder-decoder approach where the encoder learns the representation of sentences and documents while the decoder classifies each sentence based on encoder’s representations using an attention
mechanism. Our model, when extractively trained, employs a single sequence model with no decoder, and therefore may have fewer parameters. Our abstractively trained model has a decoder too, but it is different from that of (Cheng and Lapata 2016) since our decoder is used to model the likelihood of abstractive gold summaries at training time, so as to eliminate the need for extractive labels. Their model, on the other hand, requires extractive labels even with the decoder. In fact, unlike our unsupervised greedy approach to convert abstractive summaries to extractive labels, (Cheng and Lapata 2016) chose to train a separate supervised classifier using manually created labels on a subset of the data. This may yield more accurate gold extractive labels, but incurs additional annotation costs.

The work of (Nallapati et al. 2016) also uses an encoder-decoder approach, but is fully abstractive in the sense that it generates its own summaries at test time. Our abstractive trainer comes close to their work, but only generates sentence-extraction probabilities at test time. We include comparison numbers with this work too, in the following section.

4 Experiments and Results
4.1 Corpora
For our experiments, we used the CNN/DailyMail corpus originally constructed by (Hermann et al. 2015) for the task of passage-based question answering, and re-purposed for the task of document summarization as proposed in (Cheng and Lapata 2016) for extractive summarization and (Nallapati et al. 2016) for abstractive summarization. In order to make a fair comparison with the former, we left out the CNN subset of the corpus, as done by them. To compare with the latter, we used the joint CNN/Daily Mail corpora. Overall, we have 196,557 training documents, 12,147 validation documents and 10,396 test documents from the Daily Mail corpus. If we also include the CNN subset, we have 286,722 training documents, 13,362 validation documents and 11,480 test documents. On average, there are about 28 sentences per document in the training set, and an average of 3-4 sentences in the reference summaries. The average word count per document in the training set is 802.

We also used the DUC 2002 single-document summarization dataset4 consisting of 567 documents as an additional out-of-domain test set to evaluate our models.

4.2 Evaluation
In our experiments below, we evaluate the performance of SummaRuNNer using different variants of the Rouge metric 5 computed with respect to the gold summaries. To compare with (Cheng and Lapata 2016) on the Daily Mail corpus, we use limited length Rouge recall and 75 bytes and 275 bytes as reported by them. To compare with (Nallapati et al. 2016) on the CNN/Daily Mail corpus, we use the same full-length Rouge F1 metric used by the authors. On DUC 2002 corpus, following the official guidelines, we use the limited length Rouge recall metric at 75 words. We report the scores from Rouge-1, Rouge-2 and Rouge-L, which are computed using the matches of unigrams, bigrams and longest common subsequences respectively, with the ground truth summaries.

4.3 Baselines
On all datasets, we use Lead-3 model, which simply produces the leading three sentences of the document as the summary as a baseline. On the Daily Mail and DUC 2002 corpora, we also report performance of LReg, a feature-rich logistic classifier used as a baseline by (Cheng and Lapata 2016). On DUC 2002 corpus, we report several baselines such as Integer Linear Programming based approach (Woodland and Lapata 2010), and graph based approaches such as TGRAPH (Parveen, Ramsl, and Strube 2015) and URANK (Wan 2010) which achieve very high performance on this corpus. In addition, we also compare with the state-of-the-art deep learning models from (Cheng and Lapata 2016) and (Nallapati et al. 2016).

4.4 SummaRuNNer Settings
We used 100-dimensional word2vec (Mikolov et al. 2013) embeddings trained on the CNN/Daily Mail corpus as our embedding initialization. We limited the vocabulary size to 150K and the maximum number of sentences per document to 100, and the maximum sentence length to 50 words, to speed up computation. We fixed the model hidden state size at 200. We used a batch size of 64 at training time, and adadelta (Zeiler 2012) to train our model. We employed gradient clipping to regularize our model and an early stopping criterion based on validation cost. We trained SummaRuNNer both extractively as well as abstractively. When the model is abstractively trained, we denote it as SummaRuNNer-abs in the results.

At test time, picking all sentences with \( P(y = 1) \geq 0.5 \) may not be an optimal strategy since the training data is very imbalanced in terms of summary-membership of sentences. Instead, we pick sentences sorted by the predicted probabilities until we exceed the length limit when limited-length Rouge is used for evaluation. When full-length F1 is used as the metric, we fixed the number of top sentences to be selected based on the validation set.

4.5 Results on Daily Mail corpus
Table 1 shows the performance comparison of SummaRuNNer with state-of-the-art model of (Cheng and Lapata 2016) and other baselines on the DailyMail corpus using Rouge recall with summary length restricted to 75 bytes. While the abstractively trained SummaRuNNer performs on par with the state-of-the-art model, the extractively trained model significantly improves over their model.

In Table 2, we report the performance of our model with respect to Rouge recall at 275 bytes of summary length. In this case, our abstractively trained model underperforms the extractive model of (Cheng and Lapata 2016) while the extractively trained model is statistically indistinguishable from their model. This shows that the SummaRuNNer is better at picking the best sentence for summarization than the subsequent ones.
Figure 2: Visualization of SummaRuNNer output on a representative document. Each row is a sentence in the document, while the shading-color intensity is proportional to its probability of being in the summary, as estimated by the RNN-based sequence classifier. In the columns are the normalized scores from each of the abstract features in Eqn. (6) as well as the final prediction probability (last column). Sentence 2 is estimated to be the most salient, while the longest one, sentence 4, is considered the most content-rich, and not surprisingly, the first sentence the most novel. The third sentence gets the best position based score.

<table>
<thead>
<tr>
<th>Lead-3</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>LReg(500)</td>
<td>18.5</td>
<td>6.9</td>
<td>10.2</td>
</tr>
<tr>
<td>Cheng et al ’16</td>
<td>22.7</td>
<td>8.5</td>
<td>12.5</td>
</tr>
<tr>
<td>SummaRuNner-abs</td>
<td>23.8</td>
<td>9.6</td>
<td>13.3</td>
</tr>
<tr>
<td>SummaRuNNer</td>
<td>26.2±0.4*</td>
<td>10.8±0.3*</td>
<td>14.4±0.3*</td>
</tr>
</tbody>
</table>

Table 1: Performance of various models on the entire Daily Mail test set using the limited length recall variants of Rouge with respect to the abstractive ground truth at 75 bytes. Entries with asterisk are statistically significant using 95% confidence interval with respect to the nearest model, as estimated by the Rouge script.

One potential reason SummaRuNNer does not consistently outperform the extractive model of (Cheng and Lapata 2016) is the additional supervised training they used to create sentence-level extractive labels to train their model. Our model instead uses an unsupervised greedy approximation to create extractive labels from abstractive summaries, and as a result, may be more noisy than their ground truth.

We also notice that the abstractively trained SummaRuNNer underperforms its extractive counterpart. Abstractive training is more difficult since the sequence classifier is trained implicitly through the decoder which in turn depends only on the summary representation. In the future, we will investigate better design and training mechanism for the abstractive version.

4.6 Results on CNN/Daily Mail corpus

We also report the performance of SummaRuNNer on the joint CNN/Daily Mail corpus. The only other work that reports performance on this dataset is the abstractive encoder-decoder based model of (Nallapati et al. 2016), in which they use full-length F1 as the metric since neural abstractive approaches can learn when to stop generating words in the summary. In order to do a fair comparison with their work, we use the same metric as them. On this dataset, SummaRuNNer significantly outperforms their model as shown in Table 3. The superior performance of our model is not entirely surprising since abstractive summarization is a much harder problem, but the table serves to quantify the current performance gap between extractive and abstractive approaches to summarization. The results also demonstrate the difficulty of using the F1 metric for extractive summarization since SummaRuNNer, with its top three sentences with highest prediction probability as the summary, errs on the side of high recall at the expense of precision. Dynamically adjusting the summary length based on predicted probability distribution may help balance precision and recall and may further boost F1 performance, but we have not experimented with it in this work.

Table 2: Performance of various models on the entire Daily Mail test set using the limited length recall variants of Rouge at 275 bytes. SummaRuNNer is statistically indistinguishable from the model of (Cheng and Lapata 2016) at 95% C.I. on Rouge-1 and Rouge-2.
Table 3: Performance comparison of abstractive and extractive models on the entire CNN/Daily Mail test set using full-length F1 variants of Rouge. SummaRuNNer is able to significantly outperform the abstractive state-of-the-art as well as the Lead-3 baseline (on Rouge-1 and Rouge-2).

<table>
<thead>
<tr>
<th>Model</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3</td>
<td>39.2</td>
<td>15.7</td>
<td>35.5</td>
</tr>
<tr>
<td>(Nallapati et al. 2016)</td>
<td>35.4</td>
<td>13.3</td>
<td>32.6</td>
</tr>
<tr>
<td>SummaRuNNer-abs</td>
<td>37.5</td>
<td>14.5</td>
<td>33.4</td>
</tr>
<tr>
<td>SummaRuNNer</td>
<td>39.6±0.2*</td>
<td>16.2±0.2*</td>
<td>35.3±0.2*</td>
</tr>
</tbody>
</table>

Table 4: Performance of various models on the DUC 2002 set using the limited length recall variants of Rouge at 75 words. SummaRuNNer is statistically within the margin of error at 95% C.I. with respect to (Cheng and Lapata 2016), but both are lower than state-of-the-art results.

<table>
<thead>
<tr>
<th>Model</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3</td>
<td>43.6</td>
<td>21.0</td>
<td>40.2</td>
</tr>
<tr>
<td>LReg</td>
<td>43.8</td>
<td>20.7</td>
<td>40.3</td>
</tr>
<tr>
<td>ILP</td>
<td>45.4</td>
<td>21.3</td>
<td>42.8</td>
</tr>
<tr>
<td>TGRAPH</td>
<td>48.1</td>
<td>24.3*</td>
<td>-</td>
</tr>
<tr>
<td>URANK</td>
<td>48.5*</td>
<td>21.5</td>
<td>-</td>
</tr>
<tr>
<td>Cheng et al ’16</td>
<td>47.4</td>
<td>23.0</td>
<td>43.5</td>
</tr>
<tr>
<td>SummaRuNNer-abs</td>
<td>44.8</td>
<td>21.0</td>
<td>41.2</td>
</tr>
<tr>
<td>SummaRuNNer</td>
<td>46.6±0.8</td>
<td>23.1±0.9</td>
<td>43.03±0.8</td>
</tr>
</tbody>
</table>

4.7 Results on the Out-of-Domain DUC 2002 corpus

We also evaluated the models trained on the DailyMail corpus on the out-of-domain DUC 2002 set as shown in Table 4. SummaRuNNer is again statistically on par with the model of (Cheng and Lapata 2016). However, both models perform worse than graph-based TGRAPH (Parveen, Ramsl, and Strube 2015) and URANK (Wan 2010) algorithms, which are the state-of-the-art models on this corpus. Deep learning based supervised models such as SummaRuNNer and that of (Cheng and Lapata 2016) perform very well on the domain they are trained on, but may suffer from domain adaptation issues when tested on a different corpus such as DUC 2002. Graph based unsupervised approaches, on the other hand, may be more robust to domain variations.

5 Qualitative Analysis

In addition to being a state-of-the-art performer, SummaRuNNer has the additional advantage of being very interpretable. The clearly separated terms in the classification layer (see Eqn. 6) allow us to tease out various factors responsible for the classification of each sentence. This is illustrated in Figure 2, where we display a representative document from our validation set along with normalized scores from each abstract feature responsible for its final classification. Such visualization is especially useful in explaining to the end-user the decisions made by the system.

We also display a couple of example documents from the Daily Mail and DUC corpora highlighting the sentences chosen by SummaRuNNer and comparing them with the gold summary in Table 5. The examples demonstrate qualitatively that SummaRuNNer performs a reasonably good job in identifying the key points of the document.

Table 5: Example documents and gold summaries from Daily Mail (top) and DUC 2002 (bottom) corpora. The sentences chosen by SummaRuNNer for extractive summarization are highlighted in bold.

6 Conclusion

In this work, we propose a very interpretable neural sequence model for extractive document summarization that allows intuitive visualization, and show that it is better performing than or is comparable to the state-of-the-art deep learning models.

We also propose a novel abstractive training mechanism to eliminate the need for extractive labels at training time, but this approach is still a couple of Rouge points below our extractive training on most datasets. We plan to further explore combining extractive and abstractive approaches as part of our future work. One simple approach could be to pre-train the extractive model using abstractive training. Further, we plan to construct a joint extractive-abstractive model where the predictions of our extractive component form stochastic intermediate units to be consumed by the abstractive component.
References


