# Distraction-Based Neural Networks for Modeling Documents

**Qian Chen,**[1] **Xiaodan Zhu,**[2] **Zhenhua Ling,**[1] **Si Wei,**[3] **Hui Jiang**[2]
[1]University of Science and Technology of China, Hefei, China
[2]York University, Canada
[3]iFLYTEK Research, Hefei, China
cq1231@mail.ustc.edu.cn, xiaodan@cse.yorku.ca
zhling@ustc.edu.cn, siwei@iflytek.com, hj@cse.yorku.ca

## Abstract

Distributed representation learned with neural networks has recently shown to be effective in modeling natural languages at fine granularities such as words, phrases, and even sentences. Whether and how such an approach can be extended to help model larger spans of text, e.g., documents, is intriguing, and further investigation would still be desirable. This paper aims to enhance neural network models for such a purpose. A typical problem of document-level modeling is automatic summarization, which aims to model documents in order to generate summaries. In this paper, we propose neural models to train computers not just to pay attention to specific regions and content of input documents with attention models, but also distract them to traverse between different content of a document so as to better grasp the overall meaning for summarization. Without engineering any features, we train the models on two large datasets. The models achieve the state-of-the-art performance, and they significantly benefit from the distraction modeling, particularly when input documents are long.

## 1 Introduction

Modeling the meaning of text lies in center of natural language understanding. Distributed representation learned with neural networks has recently shown to be effective in modeling fine granularities of text, including words [Collobert *et al.*, 2011; Mikolov *et al.*, 2013; Chen *et al.*, 2015], phrases [Yin and Schütze, 2014], and arguably sentences [Socher *et al.*, 2012; Irsoy and Cardie, 2014; Kalchbrenner *et al.*, 2014; Tai *et al.*, 2015; Zhu *et al.*, 2015a; 2015b].

Whether and how such an approach can be extended to help model larger spans of text, e.g., documents, is intriguing, and further investigation would still be desirable, although there has been interesting research conducted recently along this line [Li *et al.*, 2015; Hu *et al.*, 2015; Wang and Cho, 2015; Hermann *et al.*, 2015]. A typical problem of document-level modeling is automatic summarization [Mani, 2001; Das and Martins, 2007; Nenkova and McKeown, 2013], in which computers generate summaries for documents, based on their shallow or deep understanding of the documents.

If one regards the process of representing input documents, generating summaries, and the interaction between them to be a (complicated) function, fitting such a function could expect to have a large-scale annotated dataset to estimate a large set of parameters, while on the other hand, hard-coding summarization knowledge (in different forms) or limiting the number of model parameters (e.g., as in many extractive summarization models) are often adopted when there are no enough training data. This work explores the former direction and utilizes relatively large datasets [Hu *et al.*, 2015; Hermann *et al.*, 2015] to train neural summarization models. In general, neural networks, as universal approximators, can fit very complicated functions and have shown to be very effective on many problems recently.

Understanding the input documents and generating summaries are both challenging. On the understanding side, much recent work seems to have suggested that distributed representations (often vectors) by themselves may not be adequate for representing sentences, let along with longer documents. Additional modeling such as soft or hard *attention* has been applied to retrospect subsequences or even words in input text to remedy the limits, which has shown to improve performances of different tasks such as those discussed in [Bahdanau *et al.*, 2014; Luong *et al.*, 2015; Rush *et al.*, 2015] among others. We regard this to be a mechanism that provides a connection between input document modeling (encoding) and summary generating (decoding), which could model a level of cognitive controls—human summarizers themselves often move between the input documents and summaries when they summarize a document.

We consider this control layer to be important, and in this paper we focus on better designing this control layer for summarization. We propose neural models to train computers not just to pay attention to specific regions and content of input documents with attention models, but also distract them to traverse between different content of a document so as to better grasp the overall meaning for summarization.

Without engineering any features, we train the models with two large datasets. The models achieve the state-of-the-art performance and they significantly benefit from the distraction modeling, particularly when input documents are long. We also explore several technologies that have been applied to sentence-level tasks and extend them to document summarization, and we present in this paper the technologies that showed to help improve the summarization performance. Even when it is applied onto the models that have leveraged these technologies, the distraction models can further improve the performance significantly. In general, our models here aim to perform *abstractive* summarization.

## 2 Related work

**Distributed representation** Distributed representation has shown to be effective in modeling fine granularities of text as discussed above. Much recent work has also attempted to model longer spans of text with neural networks [Li *et al.*, 2015; Hu *et al.*, 2015; Lin *et al.*, 2015; Wang and Cho, 2015; Hermann *et al.*, 2015]. This includes research that incorporates document-level information for language modeling [Wang and Cho, 2015; Lin *et al.*, 2015] and that answers questions [Hermann *et al.*, 2015] by comprehending input documents with attention-based models. More relevant to ours, the work of [Li *et al.*, 2015] learned distributed representation for short documents with the averaged length of about a hundred word tokens, although the objective is not summarization. Summarization typically faces documents longer than those, and summarization may be more necessary when documents are long. In this paper, we propose neural models for summarizing typical news articles with up to thousands of word tokens. We find it is necessary to enable computers not just to pay attention to specific content of input documents with attention models, but also distract them to traverse between different content so as to better grasp the overall meaning for summarization, particularly when documents are long.

**Neural summarization models** Automatic summarization has been intensively studied for both text [Mani, 2001; Das and Martins, 2007; Nenkova and McKeown, 2013] and speech [Zhu and Penn, 2006; Zhu *et al.*, 2010]. Most state-of-the-art summarization models have focused on *extractive* summarization, although some efforts have also been exerted on *abstractive* summarization. Recent neural summarization models include the recent efforts of [Rush *et al.*, 2015; Lopyrev, 2015; Hu *et al.*, 2015]. The research performed in [Rush *et al.*, 2015] focuses on neural models for sentence compression and rewriting, but not full document summarization. The work of [Lopyrev, 2015] leverages neural networks to generate news headline, where input documents are limited to 50 word tokens, and the work of [Hu *et al.*, 2015] also deals with short texts (up to dozens of word tokens), in which summarization problems such as content redundancy is less prominent and attention-based models seem to be sufficient. However, summarization typically faces documents longer than that and summarization is often more needed when documents are long. In this work we attempt to explore neural summarization technologies for news articles with up to thousands of word tokens, in which we find distraction-based summarization models help improve performance. Note that our improvement is achieved over the model that has already outperformed the attention-based model reported in [Hu *et al.*, 2015] on short documents.

## 3 Our approach

### 3.1 Overview

We base our model on the general encoder-decoder framework [Sutskever *et al.*, 2011; 2014; Cho *et al.*, 2014] that has shown to be effective recently on different tasks. This is a general sequence-to-sequence modeling framework in which the encoding part can be devoted to model the input documents and the decoder to generate output.

We believe the control layer that helps navigate the input documents to optimize the generation objectives would be of importance, and we will focus on the control layer in this paper and enrich its expressiveness. Specifically for summarization, unlike much recent work that focuses more on *attention* in order to grasp local context or correspondence (e.g, in machine translation and sentence compression), we force our models to traverse between different content of a document to avoid focusing on a region or same content, to better grasp the overall meaning for the summarization objective.

We also explore several popular technologies that have been applied to sentence-level tasks and extend them to document summarization, and we present those that help improve the summarization performance.

### 3.2 GRU-based encoding and decoding

**Encoding** The general document modeling and summarizing framework takes in an input document $\mathbf{x} = x_1, \cdots, x_{T_x}$ and write the summary of the document as the output $\mathbf{y} = y_1, \cdots, y_{T_y}$. The summarization process is modeled as finding the output text $\mathbf{y}^*$ that maximizes the conditional probability $\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$, given gold summary sequences. As discussed above, such a model has been found to be very effective in modeling sentences or sub-sentential text spans. We will address the challenges faced at the document level.

On encoding we do not restrict the encoders' architectures as if it is a recurrent neural network (RNN). The recent literature shows long-short term memory (LSTM) [Hochreiter and Schmidhuber, 1997; Sutskever *et al.*, 2014] and gated recurrent units (GRU) [Bahdanau *et al.*, 2014] are both good architectures. In developing our systems we empirically found GRU achieved similar performance as LSTM but it is fast to train; we will therefore describe the GRU implement of our neural summarization models.

In the simplest uni-directional setting, when reading input symbols from left to right, a GRU learns the hidden annotations $h_i$ at time $i$ with

$$h_i = \text{GRU}(h_{i-1}, e(x_i)) \tag{1}$$

where the $h_i \in \mathbb{R}^n$ encodes all content seen so far at time $i$ which is computed from $h_{i-1}$ and $e(x_i)$, where $e(x_i) \in \mathbb{R}^m$ is the $m$-dimensional embedding of the current word $x_i$. The forward propagation of GRU is computed as follows.

$$h_i = (1 - u_i) \odot h_{i-1} + u_i \odot \tilde{h_i} \tag{2}$$

$$\tilde{h_i} = \tanh(We(x_i) + U(r_i \odot h_{i-1})) \tag{3}$$

$$r_i = \text{sigmoid}(W_r e(x_i) + U_r h_{i-1}) \tag{4}$$

$$u_i = \text{sigmoid}(W_u e(x_i) + U_u h_{i-1}) \tag{5}$$

where $W_u$, $W_r$, $W \in \mathbb{R}^{n\times m}$ and $U_u$, $U_r$, $U \in \mathbb{R}^{n\times n}$ are weight matrices, $n$ is the number of hidden units, and $\odot$ is element-wise multiplication.

In our work, we actually applied bi-directional GRUs (bi-GRUs), which we found achieving better results than single directional GRUs consistently. As its name suggests, in a bi-GRU unit, the annotation vector $h_t$ encodes the sequence from two directions, modeling both the left and right context. The bottom part of Figure 1 shows the encoder intuitively, while for more details, readers can refer to [Bahdanau *et al.*, 2014] for further discussion.

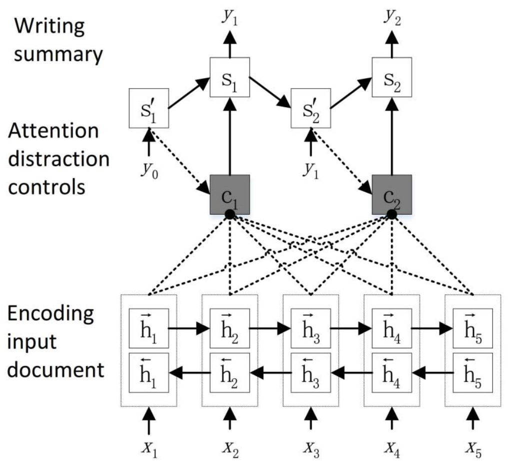
Writing summary
Attention distraction controls
Encoding input document
$y_1$
$y_2$
$s_1$
$s_2$
$s'_1$
$s'_2$
$y_0$
$y_1$
$c_1$
$c_2$
$\vec{h}_1$ $\vec{h}_2$ $\vec{h}_3$ $\vec{h}_4$ $\vec{h}_5$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_4$ $\overleftarrow{h}_5$
$x_1$ $x_2$ $x_3$ $x_4$ $x_5$

Figure 1: A high-level view of the summarization model.

**Generation** When generating summaries, the decoder predicts the next word $y_t$ given all annotations obtained in encoding $\mathbf{h} = h_1, \cdots, h_{T_x}$, as well as all previously predicted words $y_1, \cdots, y_{t-1}$. The objective is a probability over the summary $\mathbf{y}$ with decomposition into the ordered conditionals:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} \prod_{t=1}^{T_y} p(y_t | y_1, \cdots, y_{t-1}, \mathbf{h}) \quad (6)$$

$$= \operatorname*{argmax}_{\mathbf{y}} \prod_{t=1}^{T_y} g(y_{t-1}, s_t, c_t)|_{y_t} \quad (7)$$

where Equation (6) depicts a high level abstraction of generating a summary word $y_t$ over previous output as well as input annotations $\mathbf{h} = h_1, \cdots, h_{T_x}$, and $y_t$ is a legal output word at time $t$, while $\mathbf{y}^*$ is the optimal summary found by the model.

The conditional probability is further rewritten as Equation (7) to factorize the model according to the structures of neural networks. The function $g(y_{t-1}, s_t, c_t)$ is a nonlinear function that computes the probability vector for all legal output words at output time $t$, and $g(y_{t-1}, s_t, c_t)|_{y_t}$ takes the element of the resulting vector corresponding to word $y_t$, i.e., the predicated probability for word $y_t$.

The vector $s_t$ and $c_t$ are the control layers that connect output $\mathbf{y}$ and input $\mathbf{h}$, which we will discuss in details in Section 3.3. For completeness, function $g(.)$ is computed with:

$$g(y_{t-1}, s_t, c_t) = \sigma(W_o \tanh(V_o e(y_{t-1}) + U_o s_t + C_o c_t)) \quad (8)$$

where $\sigma$ is a softmax function; $W_o \in \mathbb{R}^{K\times n}$, $U_o \in \mathbb{R}^{n\times n}$, $V_o \in \mathbb{R}^{n\times m}$ and $C_o \in \mathbb{R}^{n\times 2n}$ are weight matrices; $K$ is the vocabulary size; $e(y_{t-1}) \in \mathbb{R}^m$ is the $m$-dimensional embedding of the previously predicted word $y_{t-1}$.

## 3.3 The control layers

The document modeling and summary generation are described above as two components: input document encoding and summary generation. A core problem is how these two components are associated. In sentence-level modeling such as machine translation and speech recognition, attention model is often applied to grasp local context and correspondence between input and output texts. For example, in translation attention is shown to be very useful for aligning the words of the target language (the language being translated to) to the corresponding source words and their context.

Attention can be regarded as a type of cognitive controls. In modeling documents, we take a general viewpoint on this control layer and propose distraction modeling to enable the model to traverse over different content of a long document, and we will show it improves the summarization performance significantly. In general, the control layer allows a complicated examination over the input. In this section we describe the controls that consider both attention and distraction to navigate input documents and to generate summaries.

**Two-layer hidden output** We first extended the recent two-level hidden output model [Luong *et al.*, 2015] to our summarization models. As presented later in the experiments, the two-level hidden output model consistently improves the summarization performance on different datasets. More specifically, the updating of $s_t$ follows a two-layer GRU architecture shown in the top part of Figure 1.

$$s_t = \text{GRU}_1(s'_t, c_t) \quad (9)$$

$$s'_t = \text{GRU}_2(s_{t-1}, e(y_{t-1})) \quad (10)$$

The forward propagation of $\text{GRU}_1$ and $\text{GRU}_2$ are computed similar to Equation (1) above. $\text{GRU}_1$ and $\text{GRU}_2$ use untied parameter matrices. The two-layer model allows for capturing a direct interaction between $s'_t$ and $c_t$, with the former encoding the current and previous output information and the latter encoding the current input content that is primed with distraction and attention. We will discuss how these vectors are computed below.

**Distraction in training** We propose to enforce distraction from two perspectives: adding the distraction constraints in training as well as in decoding. We first discuss the distraction in training.

*Distraction over input content vectors* In training we force the model not to pay attention to the same content or same part of the input documents too much. We accumulate the previously viewed content vector as a *history* content vector $\sum_{j=1}^{t-1} c_j$ and incorporate it into the currently computed $c'_t$. We refer to this model as **M1**.

$$c_t = \tanh(W_c c'_t - U_c \sum_{j=1}^{t-1} c_j) \quad (11)$$

where $W_c \in \mathbb{R}^{2n\times 2n}$ and $U_a \in \mathbb{R}^{2n\times 2n}$ are diagonal matrices. And $c'_t$ is input content vectors that have not been directly penalized with history yet; $c'_t$ is directly computed with conventional equation as follows:

$$c'_t = \sum_{i=1}^{T_x} \alpha_{t,i} h_i \tag{12}$$

where $h_i$ are annotation vectors that encode the current input word and its context with the input GRU described above in Equation (1). And $\alpha_{t,i}$ is the attention weight put on $h_i$ at the current output time $t$. The distraction-based $c_t$ computed in Equation (11) can then be incorporated in Equation (8).

*Distraction over attention weight vectors* We also propose to add distraction directly on the attention weight vectors. Similarly as above, we accumulate the past attention weights as a history attention weight $\sum_{j=1}^{t-1} \alpha_{j,i}$ and use it to prime the current attention weights. The model in [Tu *et al.*, 2016] also uses history attention weights, but we use history here to force distraction in order to avoid redundancy, which is not a concern in the machine translation task. We refer to the model as **M2**.

$$\alpha'_{t,i} = v_a^T \tanh(W_a s'_t + U_a h_i - b_a \sum_{j=1}^{t-1} \alpha_{j,i}) \tag{13}$$

where $W_a \in \mathbb{R}^{l \times n}$, $U_a \in \mathbb{R}^{l \times 2n}$, $b_a \in \mathbb{R}^l$, and $v_a \in \mathbb{R}^l$ are the weight matrices, and $l$ is the number of hidden units. Note that $W_a s'_t + U_a h_i$ in the equation computes the conventional attention without penalizing attention history. $\alpha'$ is often normalized with a softmax to generate attention weights $\alpha_{t,i}$ below, which is in turn used in Equation (12).

$$\alpha_{t,i} = \frac{\exp(\alpha'_{t,i})}{\sum_{j=1}^{T_x} \exp(\alpha'_{t,j})} \tag{14}$$

**Distraction in decoding** In the decoding process, we also enforced different types of distraction, one by computing the difference between the distribution of the current attention weight $\alpha_t$ and that of all previous attention weights $\alpha_1, \cdots, \alpha_{t-1}$. Since $\alpha$ can be seen as a proper probabilistic distribution, normalized in Equation (14), we used Kullback-Leibler (KL) divergence to measure their difference with Equation (15), which was found to be consistently better than several other distance metrics we tried on the held-out data.

We also enforced distraction in a similar way on the attention-primed input content vector $c_t$, as well as on the hidden output vector $s_t$. Both $c_t$ and $s_t$ are not normalized but are regular content vectors, where the cosine similarity was found achieving a better performance than several alternatives (e.g., $l_1$ and $l_2$ distances) on the held-out data.

$$d_{\alpha,t} = \min_{i \in \{1, \cdots t-1\}} \mathrm{KL}(\alpha_t, \alpha_i) \tag{15}$$

$$d_{c,t} = \max_{i \in \{1, \cdots t-1\}} \mathrm{cosine}(c_t, c_i) \tag{16}$$

$$d_{s,t} = \max_{i \in \{1, \cdots t-1\}} \mathrm{cosine}(s_t, s_i) \tag{17}$$

The distraction score $d_{*,t}$ was then added into the output probability and the beam search in order to encourage the model to avoid redundant content.

$$\mathrm{score}_t = \sum_{t=1}^{T_y} \{\log(p_{y_t}) + \lambda_1 d_{\alpha,t} + \lambda_2 d_{c,t} + \lambda_3 d_{s,t}\} \tag{18}$$

where $score_t$ was used as follows in the *beam search with distraction* algorithm, and parameter $\lambda_1$, $\lambda_2$, and $\lambda_3$ were determined on the development set. We refer to this model as **M3**.

**Algorithm 1** Beam search with distraction

```
Require: Vocabulary size $K$, beam size $B$, max output length $N$
  ▷ Computed probabilities of all the words in vocabulary
  ▷ Choose the $B$ most likely words and initialize the $B$ hypotheses
  for $i = 1 : N$ do
      ▷ For each hypothesis, compute the next conditional probabilities, then have $B \times K$ candidates with the corresponding probabilities
      ▷ Use the distraction-primed value score to choose $B$ most likely candidates
  end for
```

**Unknown word replacement for summarization** We borrowed the unknown word replacement [Jean *et al.*, 2015] from machine translation to our summarization models and found it improved the performance when summarizing long documents. Specifically, due to the time complexity in handling a larger vocabulary in the softmax layer in summary generation, infrequent words were removed from the vocabulary and were replaced with the symbol $\langle UNK \rangle$. The threshold of vocabulary size is data-dependent and will be detailed later in the experiment set-up section.

After the first-round summary generated for a document, a token labeled as $\langle UNK \rangle$ will be replaced with a word in the input documents. More specifically, we obtained the replacement using Equation (14); i.e., we used the largest element in $\alpha_t$ to find the source location for the current $\langle UNK \rangle$.

## 4 Experiment set-up

### 4.1 Data

We experiment with our summarization models on two publicly available corpora with different document lengths and in different languages: a CNN news collection [Hermann *et al.*, 2015] and a Chinese corpus made available more recently in [Hu *et al.*, 2015]. Both are large datasets appropriate for training neural models, which, as discussed above, employ a large number of parameters to fit the potentially complicated summarization process involving representing input documents, generating summaries, and interacting between them.

**CNN data** The CNN data [Hermann *et al.*, 2015] have a human-generated real-life summary for each news article. The dataset collected in was made available at GitHub.[1] The data was preprocessed with the Stanford CoreNLP tools [Manning *et al.*, 2014] for tokenization and sentence-boundary detection; all capital information is kept. To speed up training, we removed the documents that are too long (over 2,500 word tokens) from the training and validation set, but

[1]https://github.com/deepmind/rc-data

kept all documents in the test set, which does not change the difficulty of the task.

**LCSTS data** The second corpus is LCSTS, which is a Chinese corpus made available more recently in [Hu *et al.*, 2015]. The data is constructed from the Chinese microblogging website, Sina Weibo. We used the original training/testing split mentioned in [Hu *et al.*, 2015], but additionally randomly sampled a small part of the training data as our validation set.

Table 1 gives more details about the two datasets. We can see from the table that averaged document length of the CNN corpus is about seven time as long as the LCSTS corpus, and the summary is about 2-3 times longer.

| | **CNN** | | | **LCSTS** | | |
|---|---|---|---|---|---|---|
| | Train | Valid | Test | Train | Valid | Test |
| Doc.L. | 775.2 | 761.3 | 765.4 | 103.7 | 100.3 | 108.1 |
| Sum.L. | 48.4 | 36.5 | 36.6 | 17.9 | 18.2 | 18.7 |
| # Doc. | 81,824 | 1,184 | 1,093 | 2,400,000 | 591 | 725 |

Table 1: The CNN and LCSTS dataset. The first two rows of the table are the averaged document length (Doc.L.) and summary length (Sum.L.) in terms of numbers of word tokens. The bottom row lists the number of documents in the datasets.

## 4.2 Training details

We used mini-batch stochastic gradient descent (SGD) to optimize log-likelihood, and Adadelta [Zeiler, 2012] to automatically adapt the learning rate of parameters ($\epsilon = 10^{-6}$ and $\rho = 0.95$).

For the CNN dataset, training was performed with shuffled mini-batches of size 64 after sorting by length. We limit our vocabulary to include the top 25,000 most frequent words. Other words were replaced with the token $\langle UNK \rangle$, as discussed earlier in the paper. Based on the validation data, we set embedding dimension to be 120, the vector length in hidden layers to be 500 for uni-GRU and 600 for bi-GRU. An end-of-sentence token was inserted between every sentence, and an end-of-document token was added at the end. The beam size of decoder was set to be 5.

For the LCSTS data, a larger mini-batch size 256 was found to be better based the observation on the validation set. Same as in [Hu *et al.*, 2015], we used characters rather than words as our tokens. The vocabulary size is 4000, embedding dimension is 500, and the vector size of the hidden-layer nodes is 500. Beam search size is 5, same as in the CNN dataset.

We make our code publicly available[2]. Our implementation uses python and is based on the Theano library [Bergstra *et al.*, 2010].

## 5 Experimental results

## 5.1 Results on the CNN dataset

**Overall performance** Our results on the CNN dataset are presented in Table 2. We used Rouge scores [Lin, 2004] to measure performance. Since the summary lengths are not preset to be the same, we report $F_1$ Rouge. The upper part of the table includes the baseline results of a number of typical summarization algorithms, which we listed in the table as Luhn [Luhn, 1958], Edmundson [Edmundson, 1969], LSA [Steinberger and Jezek, 2004], Lex-rank [Erkan and Radev, 2004], Text-rank [Mihalcea and Tarau, 2004], Sum-basic [Vanderwende *et al.*, 2007], and KL-sum [Haghighi and Vanderwende, 2009]. These baseline results are implemented in the open-source tool SUMY[3].

The results at the lower half of the table show that the bi-GRU encoder achieves a better performance than the uni-GRU encoder. This is consistent with the results on the LCSTS dataset reported later in Table 4. We show that two-level output model we discussed in the method section is beneficial, which is also consistent with the results on the LCSTS dataset. In addition, the unknown replacement technique yields an additional improvement.

Over the strong model that has used these technologies (the row marked as "+UNK replace"), the model in the last row that incorporates all distraction modeling (M1, M2 and M3) finally achieves a Rouge-1 score of 27.1, a Rouge-2 score of 8.2, and a Rouge-L score of 18.7, significantly improving the three Rouge scores by 5.8, 1.9, and 2.3, respectively. These are also the largest improvement presented in the table, compared with the other techniques listed. The table also lists the details of how the model M1, M2, and M3 improve the performance additively. Again, the neural models do not engineer any features and use only content but not any additional formality features such as locations of input sentences, which may bring additional improvement.

| System | Rouge-1 | Rouge-2 | Rouge-L |
|---|---|---|---|
| Luhn | 23.2 | 7.2 | 15.5 |
| Edmundson | 24.5 | 8.2 | 16.7 |
| LSA | 21.2 | 6.2 | 14.0 |
| Lex-rank | 26.1 | **9.6** | 17.7 |
| Text-rank | 23.3 | 7.7 | 15.8 |
| Sum-basic | 22.9 | 5.5 | 14.8 |
| KL-sum | 20.7 | 5.9 | 13.7 |
| Uni-GRU | 18.4 | 4.8 | 14.3 |
| Bi-GRU | 19.5 | 5.2 | 15.0 |
| +Two-level out | 20.2 | 5.9 | 15.7 |
| +UNK replace | 21.3 | 6.3 | 16.4 |
| +Distraction M1 | 22.2 | 6.5 | 16.7 |
| +Distraction M2 | 24.4 | 7.7 | 17.8 |
| +Distraction M3 | **27.1** | 8.2 | **18.7** |

Table 2: Results on the CNN dataset.

**Performance on different lengths of documents** To observe the effectiveness of the distraction model over different document lengths, we further selected all short documents from the CNN training dataset into a subset (subset-1) with average length at 335 word tokens, and a subset of data (subset-2) that have the same number of documents as the subset-1, with

[2]Our code is available at https://github.com/lukecq1231/nats

[3]https://pypi.python.org/pypi/sumy

averaged document length at 680 word tokens. As shown in Table 3, on the data subset-2, the distraction model improves the results more significantly. The relative improvement is 29.0%, 25.6%, and 10.8%, compared with 25.9%, 20.5%, and 8.1% on subset-1, respectively. In general the best performance on both dataset is lower than that using all training data, suggesting using more training data can improve summarization performance.

| | Rouge-1 | Rouge-2 | Rouge-L |
|---|---|---|---|
| | Subset-1 | | |
| w.o. distraction | 18.1 | 3.9 | 13.6 |
| +Distraction | 22.8 | 4.7 | 14.7 |
| Relative Impr. | 25.9% | 20.5% | 8.1% |
| | Subset-2 | | |
| w.o. distraction | 18.6 | 3.9 | 13.8 |
| +Distraction | 24.0 | 4.9 | 15.3 |
| Relative Impr. | 29.0% | 25.6% | 10.8% |

Table 3: Results on two subsets of the CNN datasets with different document lengths.

### 5.2 Results on the LCSTS dataset

We experiment with the proposed model on public LCSTS corpus. The baseline is the best result reported in [Hu *et al.*, 2015][4]. Our modified uni-GRU achieves a slight improvement over the reported results. The Bi-GRU attention-based model achieves a better performance, confirming the usefulness of bi-directional models for summarization as well as that our implementation is the state-of-the-art and serves as a very strong baseline in the CNN dataset discussed above. Note that since the input text length of LCSTS is far shorter than the CNN documents, each containing about 100 words and roughly 6-8 sentences, we show that distraction does not improve the performance, but in contrast, when documents are longer, its benefits are significant, achieving the biggest improvement as discussed earlier. This suggests the effectiveness of distraction modeling in helping summarize the more challenging longer documents, where summarization is often more necessary than for short texts.

| System | Rouge-1 | Rouge-2 | Rouge-L |
|---|---|---|---|
| [Hu *et al.*, 2015] | 29.9 | 17.4 | 27.2 |
| Uni-GRU | 32.1 | 19.9 | 29.4 |
| Bi-GRU | 33.2 | 20.8 | 30.5 |
| +Two-level Att. | 35.2 | 22.6 | 32.5 |
| +UNK replace | 35.2 | 22.6 | 32.5 |
| +Distraction | 35.2 | 22.6 | 32.5 |

Table 4: Results on the LCSTS dataset.

[4]We thank the authors of [Hu *et al.*, 2015] for generously sharing us the latest output of their models, which achieves a better performance than the results reported in [Hu *et al.*, 2015]. We reported here the updated scores higher performance as our baseline.

We also compare our models with the simple baseline that selects the first $N$ numbers of word tokens from the input documents, which reaches its maximal Rouge scores when the first 30 tokens were taken, and achieves Rouge-1, Rouge-2, and Rouge-L at 25.5, 14.1 and 21.4. And our models are significantly better than that. For the CNN data set, choosing the first three sentences achieves the best results, which reach Rouge-1, Rouge-2, and Rouge-L at 26.1, 9.6 and 17.8, respectively. Since the CNN data is news data, the baseline of selecting first several sentences has known to be a very strong baseline. Again, the models we explore here are towards performing abstractive summarization.

## 6 Conclusions and future work

We propose to train neural document summarization models not just to pay attention to specific regions of input documents with attention models, but also distract the models to different content in order to better grasp the overall meaning of input documents. Without engineering any features, we train the models on two large datasets. The models achieve the state-of-the-art performance and they significantly benefit from the distraction modeling, particularly when the input documents are long. We also explore several recent technologies for summarization and show that they help improve summarization performance as well. Even if applied onto the models that have already leveraged these technologies, the distraction models can further improve the performance significantly.

From a more general viewpoint, enriching the expressiveness of the control layers that link the input encoding layer and the output decoding layer could be of importance to remedy the shortcomings of the current models. We plan to perform more work along this direction.

## Acknowledgments

The first and the third author of this paper were supported in part by the Science and Technology Development of Anhui Province, China (Grants No. 2014z02006), the Fundamental Research Funds for the Central Universities (Grant No. WK2350000001) and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB02070006).

## References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[Bergstra *et al.*, 2010] J Bergstra, O Breuleux, F Bastien, P Lamblin, R Pascanu, G Desjardins, J Turian, D Warde-Farley, and Y Bengio. Theano: a cpu and gpu math expression compiler. In *SciPy*, volume 4, page 3. Austin, TX, 2010.

[Chen *et al.*, 2015] Zhigang Chen, Wei Lin, Qian Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. Revisiting word embedding for contrasting meaning. In *Proceedings of ACL*, 2015.

[Cho *et al.*, 2014] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[Collobert *et al.*, 2011] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

[Das and Martins, 2007] Dipanjan Das and Andre Martins. A survey on automatic text summarization. 2007.

[Edmundson, 1969] Harold P Edmundson. New methods in automatic extracting. *JACM*, 16(2):264–285, 1969.

[Erkan and Radev, 2004] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, pages 457–479, 2004.

[Haghighi and Vanderwende, 2009] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *NAACL*, 2009.

[Hermann *et al.*, 2015] K. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2015] Baotian Hu, Qingcai Chen, and Fangze Zhu. Lcsts: A large scale chinese short text summarization dataset. In *EMNLP*, 2015.

[Irsoy and Cardie, 2014] Ozan Irsoy and Claire Cardie. Deep recursive neural networks for compositionality in language. In *NIPS*, 2014.

[Jean *et al.*, 2015] Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *ACL*, 2015.

[Kalchbrenner *et al.*, 2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *ACL*, June 2014.

[Li *et al.*, 2015] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*, 2015.

[Lin *et al.*, 2015] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li. Hierarchical recurrent neural network for document modeling. In *EMNLP*, 2015.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. 2004.

[Lopyrev, 2015] Konstantin Lopyrev. Generating news headlines with recurrent neural networks. *CoRR*, abs/1512.01712, 2015.

[Luhn, 1958] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[Luong *et al.*, 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[Mani, 2001] Inderjeet Mani. *Automatic Summarization*. J. Benjamins Pub. Co., Amsterdam, 2001.

[Manning *et al.*, 2014] C Manning, M Surdeanu, J Bauer, J Finkel, S Bethard, and D McClosky. The stanford corenlp natural language processing toolkit. In *ACL*, 2014.

[Mihalcea and Tarau, 2004] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, 2004.

[Mikolov *et al.*, 2013] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[Nenkova and McKeown, 2013] Ani Nenkova and Kathleen McKeown. *A survey of text summarization techniques*. Springer, 2013.

[Rush *et al.*, 2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[Socher *et al.*, 2012] R. Socher, B. Huval, C. Manning, and A. Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, 2012.

[Steinberger and Jezek, 2004] J. Steinberger and K. Jezek. Using latent semantic analysis in text summarization and summary evaluation. In *ISIM*, pages 93–100, 2004.

[Sutskever *et al.*, 2011] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *ICML*, pages 1017–1024, 2011.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *ACL*, 2015.

[Tu *et al.*, 2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Coverage-based neural machine translation. *CoRR*, abs/1601.04811, 2016.

[Vanderwende *et al.*, 2007] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *IP&M*, 43(6):1606–1618, 2007.

[Wang and Cho, 2015] Tian Wang and Kyunghyun Cho. Larger-context language modelling. *CoRR*, abs/1511.03729, 2015.

[Yin and Schütze, 2014] W. Yin and H. Schütze. An exploration of embeddings for generalized phrases. In *ACL 2014 Student Research Workshop*, pages 41–47, June 2014.

[Zeiler, 2012] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[Zhu and Penn, 2006] Xiaodan Zhu and Gerald Penn. Comparing the roles of textual, acoustic and spoken-language features on spontaneous conversation summarization. In *NAACL*, 2006.

[Zhu *et al.*, 2010] Xiaodan Zhu, Gerald Penn, and Frank Rudzicz. Summarizing multiple spoken documents: Finding evidence from untranscribed audio. In *ACL*, 2010.

[Zhu *et al.*, 2015a] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long Short-Term Memory over Recursive Structures. In *ICML*, 2015.

[Zhu *et al.*, 2015b] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over tree structures. *CoRR*, abs/1503.04881, 2015.