Neural Summarization by Extracting Sentences and Words

Jianpeng Cheng **Mirella Lapata**
ILCC, School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
jianpeng.cheng@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

Traditional approaches to extractive summarization rely heavily on human-engineered features. In this work we propose a data-driven approach based on neural networks and continuous sentence features. We develop a general framework for single-document summarization composed of a hierarchical document encoder and an attention-based extractor. This architecture allows us to develop different classes of summarization models which can extract sentences or words. We train our models on large scale corpora containing hundreds of thousands of document-summary pairs[1]. Experimental results on two summarization datasets demonstrate that our models obtain results comparable to the state of the art without any access to linguistic annotation.

1 Introduction

The need to access and digest large amounts of textual data has provided strong impetus to develop automatic summarization systems aiming to create shorter versions of one or more documents, whilst preserving their information content. Much effort in automatic summarization has been devoted to sentence extraction, where a summary is created by identifying and subsequently concatenating the most salient text units in a document.

Most extractive methods to date identify sentences based on human-engineered features. These include surface features such as sentence position and length (Radev et al., 2004), the words in the title, the presence of proper nouns, content features such as word frequency (Nenkova et al., 2006), and event features such as action nouns (Filatova and Hatzivassiloglou, 2004). Sentences are typically assigned a score indicating the strength of presence of these features. Several methods have been used in order to select the summary sentences ranging from binary classifiers (Kupiec et al., 1995), to hidden Markov models (Conroy and O'Leary, 2001), graph-based algorithms (Erkan and Radev, 2004; Mihalcea, 2005), and integer linear programming (Woodsend and Lapata, 2010).

In this work we propose a data-driven approach to summarization based on neural networks and continuous sentence features. There has been a surge of interest recently in repurposing sequence transduction neural network architectures for NLP tasks such as machine translation (Sutskever et al., 2014), question answering (Hermann et al., 2015), and sentence compression (Rush et al., 2015). Central to these approaches is an encoder-decoder architecture modeled by recurrent neural networks. The encoder reads the source sequence into a list of continuous-space representations from which the decoder generates the target sequence. An attention mechanism (Bahdanau et al., 2015) is often used to locate the region of focus during decoding.

We develop a general framework for single-document summarization which can be used to extract sentences or words. Our model includes a neural network-based hierarchical document reader or encoder and an attention-based content extractor. The role of the reader is to derive the meaning representation of a document based on its sentences and their constituent words. Our models adopt a variant of neural attention to extract sentences or words. Contrary to previous work where attention is an *intermediate* step used to blend hidden units of an encoder to a vector propagating additional information to the decoder, our model applies attention *directly* to select sentences or words of the input document as the output summary. Similar neural attention architectures have been previously used for geometry reasoning (Vinyals et al., 2015), under the name *Pointer Networks*.

[1]Resources are available for download at http://homepages.inf.ed.ac.uk/s1537177/resources.html

One stumbling block to applying neural network models to extractive summarization is the lack of training data, i.e., documents with sentences (and words) labeled as summary-worthy. Inspired by previous work on summarization (Woodsend and Lapata, 2010; Svore et al., 2007) and reading comprehension (Hermann et al., 2015) we retrieve hundreds of thousands of news articles and corresponding highlights from the DailyMail website. Highlights usually appear as bullet points giving a brief overview of the information contained in the article (see Figure 1 for an example). Using a number of transformation and scoring algorithms, we are able to match highlights to document content and construct two large scale training datasets, one for sentence extraction and the other for word extraction. Previous approaches have used small scale training data in the range of a few hundred examples.

Our work touches on several strands of research within summarization and neural sequence modeling. The idea of creating a summary by extracting words from the source document was pioneered in Banko et al. (2000) who view summarization as a problem analogous to statistical machine translation and generate headlines using statistical models for selecting and ordering the summary words. Our word-based model is similar in spirit, however, it operates over continuous representations, produces multi-sentence output, and jointly selects summary words and organizes them into sentences. A few recent studies (Kobayashi et al., 2015; Yogatama et al., 2015) perform sentence extraction based on pre-trained sentence embeddings following an unsupervised optimization paradigm. Our work also uses continuous representations to express the meaning of sentences and documents, but importantly employs neural networks more directly to perform the actual summarization task.

Rush et al. (2015) propose a neural attention model for abstractive sentence compression which is trained on pairs of headlines and first sentences in an article. In contrast, our model summarizes documents rather than individual sentences, producing multi-sentential discourse. A major architectural difference is that our decoder selects output symbols from the document of interest rather than the entire vocabulary. This effectively helps us sidestep the difficulty of searching for the next output symbol under *a large vocabulary*, with *low-frequency words* and *named entities* whose representations can be challenging to learn. Gu et al. (2016) and Gulcehre et al. (2016) propose a similar "copy" mechanism in sentence compression and other tasks; their model can accommodate both generation and extraction by selecting which sub-sequences in the input sequence to copy in the output.

We evaluate our models both automatically (in terms of ROUGE) and by humans on two datasets: the benchmark DUC 2002 document summarization corpus and our own DailyMail news highlights corpus. Experimental results show that our summarizers achieve performance comparable to state-of-the-art systems employing hand-engineered features and sophisticated linguistic constraints.

2 Problem Formulation

In this section we formally define the summarization tasks considered in this paper. Given a document D consisting of a sequence of sentences $\{s_1, \cdots, s_m\}$ and a word set $\{w_1, \cdots, w_n\}$, we are interested in obtaining summaries at two levels of granularity, namely sentences and words.

Sentence extraction aims to create a summary from D by selecting a subset of j sentences (where $j < m$). We do this by scoring each sentence within D and predicting a label $y_L \in \{0, 1\}$ indicating whether the sentence should be included in the summary. As we apply supervised training, the objective is to maximize the likelihood of all sentence labels $\mathbf{y}_L = (y_L^1, \cdots, y_L^m)$ given the input document D and model parameters θ:

$$\log p(\mathbf{y}_L|D;\theta) = \sum_{i=1}^{m} \log p(y_L^i|D;\theta) \qquad (1)$$

Although extractive methods yield naturally grammatical summaries and require relatively little linguistic analysis, the selected sentences make for long summaries containing much redundant information. For this reason, we also develop a model based on **word extraction** which seeks to find a subset of words[2] in D and their optimal ordering so as to form a summary $\mathbf{y}_s = (w'_1, \cdots, w'_k), w'_i \in D$. Compared to sentence extraction which is a sequence labeling problem, this task occupies the middle ground between full abstractive summarization which can exhibit a wide range of rewrite operations and extractive

[2]The vocabulary can also be extended to include a small set of commonly-used (high-frequency) words.

AFL star blames vomiting cat for speeding
Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat.
The 22-year-old AFL star, who drove 96km/h in a 60km/h road works zone on the South Eastern expressway in February, said he didn't see the reduced speed sign because he was so distracted by his cat vomiting violently in the back seat of his car.
In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h.
He lost four demerit points, instead of seven, because of his significant training commitments.

- *Adelaide Crows defender Daniel Talia admits to speeding but says he didn't see road signs because his cat was vomiting in his car.*
- *22-year-old Talia was fined $824 and four demerit points, instead of seven, because of his 'significant' training commitments.*

Figure 1: DailyMail news article with highlights. Underlined sentences bear label 1, and 0 otherwise.

summarization which exhibits none. We formulate word extraction as a language generation task with an output vocabulary restricted to the original document. In our supervised setting, the training goal is to maximize the likelihood of the generated sentences, which can be further decomposed by enforcing conditional dependencies among their constituent words:

$$\log p(\mathbf{y}_s|D;\theta)=\sum_{i=1}^{k}\log p(w'_i|D,w'_1,\cdots,w'_{i-1};\theta) \quad (2)$$

In the following section, we discuss the data elicitation methods which allow us to train neural networks based on the above defined objectives.

3 Training Data for Summarization

Data-driven neural summarization models require a large training corpus of documents with labels indicating which sentences (or words) should be in the summary. Until now such corpora have been limited to hundreds of examples (e.g., the DUC 2002 single document summarization corpus) and thus used mostly for testing (Woodsend and Lapata, 2010). To overcome the paucity of annotated data for training, we adopt a methodology similar to Hermann et al. (2015) and create two large-scale datasets, one for sentence extraction and another one for word extraction.

In a nutshell, we retrieved[3] hundreds of thousands of news articles and their corresponding highlights from DailyMail (see Figure 1 for an example). The highlights (created by news editors) are genuinely abstractive summaries and therefore not readily suited to supervised training. To create the training data for **sentence extraction**, we reverse approximated the gold standard label of each document sentence given the summary based on their semantic correspondence (Woodsend and Lapata, 2010). Specifically, we designed a rule-based system that determines whether a document sentence matches a highlight and should be labeled with 1 (must be in the summary), and 0 otherwise. The rules take into account the position of the sentence in the document, the unigram and bigram overlap between document sentences and highlights, the number of entities appearing in the highlight and in the document sentence. We adjusted the weights of the rules on 9,000 documents with manual sentence labels created by Woodsend and Lapata (2010). The method obtained an accuracy of 85% when evaluated on a held-out set of 216 documents coming from the same dataset and was subsequently used to label 200K documents. Approximately 30% of the sentences in each document were deemed summary-worthy.

For the creation of the **word extraction** dataset, we examine the lexical overlap between the highlights and the news article. In cases where all highlight words (after stemming) come from the original document, the document-highlight pair constitutes a valid training example and is added to the word extraction dataset. For out-of-vocabulary (OOV) words, we try to find a semantically equivalent replacement present in the news article. Specifically, we check if a neighbor, represented

[3]The script for constructing our datasets is modified from the one released in Hermann et al. (2015).

by pre-trained[4] embeddings, is in the original document and therefore constitutes a valid substitution. If we cannot find any substitutes, we discard the document-highlight pair. Following this procedure, we obtained a word extraction dataset containing 170K articles, again from the DailyMail.

4 Neural Summarization Model

The key components of our summarization model include a neural network-based hierarchical document reader and an attention-based hierarchical content extractor. The hierarchical nature of our model reflects the intuition that documents are generated compositionally from words, sentences, paragraphs, or even larger units. We therefore employ a representation framework which reflects the same architecture, with global information being discovered and local information being preserved. Such a representation yields minimum information loss and is flexible allowing us to apply neural attention for selecting salient sentences and words within a larger context. In the following, we first describe the document reader, and then present the details of our sentence and word extractors.

4.1 Document Reader

The role of the reader is to derive the meaning representation of the document from its constituent sentences, each of which is treated as a sequence of words. We first obtain representation vectors at the sentence level using a single-layer convolutional neural network (CNN) with a max-over-time pooling operation (Kalchbrenner and Blunsom, 2013; Zhang and Lapata, 2014; Kim et al., 2016). Next, we build representations for documents using a standard recurrent neural network (RNN) that recursively composes sentences. The CNN operates at the word level, leading to the acquisition of sentence-level representations that are then used as inputs to the RNN that acquires document-level representations, in a hierarchical fashion. We describe these two sub-components of the text reader below.

Convolutional Sentence Encoder We opted for a convolutional neural network model for representing sentences for two reasons. Firstly, single-layer CNNs can be trained effectively (without any long-term dependencies in the model) and secondly, they have been successfully used for sentence-level classification tasks such as sentiment analysis (Kim, 2014). Let d denote the dimension of word embeddings, and s a document sentence consisting of a sequence of n words $(w_1, \cdots, w_n)$ which can be represented by a dense column matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$. We apply a temporal narrow convolution between $\mathbf{W}$ and a kernel $\mathbf{K} \in \mathbb{R}^{c \times d}$ of width c as follows:

$$\mathbf{f}^i_j = \tanh(\mathbf{W}_{j:j+c-1} \otimes \mathbf{K} + b) \tag{3}$$

where $\otimes$ equates to the Hadamard Product followed by a sum over all elements. $\mathbf{f}^i_j$ denotes the j-th element of the i-th feature map $\mathbf{f}^i$ and b is the bias. We perform max pooling over time to obtain a *single* feature (the ith feature) representing the sentence under the kernel $\mathbf{K}$ with width c:

$$\mathbf{s}_{i,\mathbf{K}} = \max_j \mathbf{f}^i_j \tag{4}$$

In practice, we use multiple feature maps to compute a list of features that match the dimensionality of a sentence under each kernel width. In addition, we apply multiple kernels with different widths to obtain a set of different sentence vectors. Finally, we sum these sentence vectors to obtain the final sentence representation. The CNN model is schematically illustrated in Figure 2 (bottom). In the example, the sentence embeddings have six dimensions, so six feature maps are used under each kernel width. The blue feature maps have width two and the red feature maps have width three. The sentence embeddings obtained under each kernel width are summed to get the final sentence representation (denoted by green).

Recurrent Document Encoder At the document level, a recurrent neural network composes a sequence of sentence vectors into a document vector. Note that this is a somewhat simplistic attempt at capturing document organization at the level of sentence to sentence transitions. One might view the hidden states of the recurrent neural network as a list of partial representations with each focusing mostly on the corresponding input sentence given the previous context. These representations altogether constitute the document representation, which captures local and global sentential information with minimum compression.

The RNN we used has a Long Short-Term Memory (LSTM) activation unit for ameliorating the vanishing gradient problem when training long sequences (Hochreiter and Schmidhuber,

[4]We used the Python Gensim library and the 300-dimensional *GoogleNews* vectors.

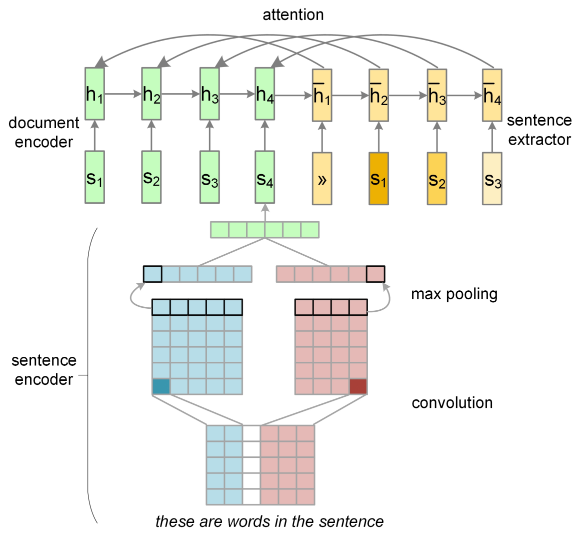

Figure 2: A recurrent convolutional document reader with a neural sentence extractor.

1997). Given a document $d = (s_1, \cdots, s_m)$, the hidden state at time step t, denoted by $\mathbf{h_t}$, is updated as:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \mathbf{W} \cdot \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{s}_t \end{bmatrix} \tag{5}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \tag{6}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{7}$$

where $\mathbf{W}$ is a learnable weight matrix. Next, we discuss a special attention mechanism for extracting sentences and words given the recurrent document encoder just described, starting from the sentence extractor.

4.2 Sentence Extractor

In the standard neural sequence-to-sequence modeling paradigm (Bahdanau et al., 2015), an attention mechanism is used as an intermediate step to decide which input region to focus on in order to generate the next output. In contrast, our sentence extractor applies attention to directly extract salient sentences after reading them.

The extractor is another recurrent neural network that labels sentences sequentially, taking into account not only whether they are individually relevant but also mutually redundant. The complete architecture for the document encoder and the sentence extractor is shown in Figure 2. As can be seen, the next labeling decision is made

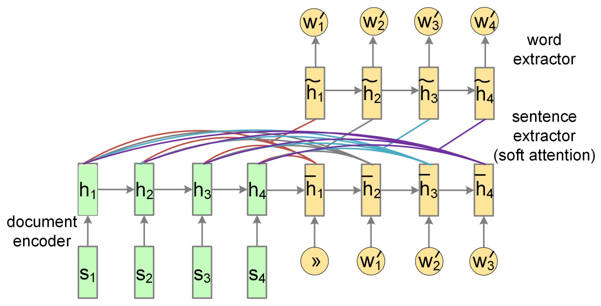

Figure 3: Neural attention mechanism for word extraction.

with both the encoded document and the previously labeled sentences in mind. Given encoder hidden states $(h_1, \cdots, h_m)$ and extractor hidden states $(\bar{h}_1, \cdots, \bar{h}_m)$ at time step t, the decoder attends the t-th sentence by relating its current decoding state to the corresponding encoding state:

$$\bar{\mathbf{h}}_t = \text{LSTM}(p_{t-1}\mathbf{s}_{t-1}, \bar{\mathbf{h}}_{t-1}) \tag{8}$$

$$p(y_L(t) = 1|D) = \sigma(\text{MLP}(\bar{\mathbf{h}}_t : \mathbf{h}_t)) \tag{9}$$

where MLP is a multi-layer neural network with as input the concatenation of $\bar{\mathbf{h}}_t$ and $\mathbf{h}_t$. p_{t-1} represents the degree to which the extractor believes the previous sentence should be extracted and memorized (p_{t-1}=1 if the system is certain; 0 otherwise).

In practice, there is a discrepancy between training and testing such a model. During training we know the true label p_{t-1} of the previous sentence, whereas at test time p_{t-1} is unknown and has to be predicted by the model. The discrepancy can lead to quickly accumulating prediction errors, especially when mistakes are made early in the sequence labeling process. To mitigate this, we adopt a curriculum learning strategy (Bengio et al., 2015): at the beginning of training when p_{t-1} cannot be predicted accurately, we set it to the true label of the previous sentence; as training goes on, we gradually shift its value to the predicted label $p(y_L(t-1) = 1|d)$.

4.3 Word Extractor

Compared to sentence extraction which is a purely sequence labeling task, word extraction is closer to a generation task where relevant content must be selected and then rendered fluently and grammatically. A small extension to the structure of the sequential labeling model makes it suitable for generation: instead of predicting a label for the next sentence at each time step, the model directly outputs the next word in the summary. The model uses a *hierarchical* attention architecture: at time step t, the decoder softly[5] attends each document sentence and subsequently attends each word in the document and computes the probability of the next word to be included in the summary $p(w'_t = w_i|d, w'_1, \cdots, w'_{t-1})$ with a softmax classifier:

$$\bar{\mathbf{h}}_t = \text{LSTM}(\mathbf{w'}_{t-1}, \bar{\mathbf{h}}_{t-1})^6 \quad (10)$$

$$a^t_j = \mathbf{z}^{\text{T}} \tanh(\mathbf{W}_e \bar{\mathbf{h}}_t + \mathbf{W}_r \mathbf{h}_j), h_j \in D \quad (11)$$

$$b^t_j = \text{softmax}(a^t_j) \quad (12)$$

$$\tilde{\mathbf{h}}_t = \sum_{j=1}^{m} b^t_j \mathbf{h}_j \quad (13)$$

$$u^t_i = \mathbf{v}^{\text{T}} \tanh(\mathbf{W}_{e'} \tilde{\mathbf{h}}_t + \mathbf{W}_{r'} \mathbf{w}_i), w_i \in D \quad (14)$$

$$p(w'_t = w_i|D, w'_1, \cdots, w'_{t-1}) = \text{softmax}(u^t_i) \quad (15)$$

In the above equations, $\mathbf{w}_i$ corresponds to the vector of the i-th word in the input document, whereas $\mathbf{z}$, $\mathbf{W}_e$, $\mathbf{W}_r$, $\mathbf{v}$, $\mathbf{W}_{e'}$, and $\mathbf{W}_{r'}$ are model weights. The model architecture is shown in Figure 3.

The word extractor can be viewed as a conditional language model with a vocabulary constraint. In practice, it is not powerful enough to enforce grammaticality due to the lexical diversity and sparsity of the document highlights. A possible enhancement would be to pair the extractor with a neural language model, which can be pre-trained on a large amount of unlabeled documents and then jointly tuned with the extractor during decoding (Gulcehre et al., 2015). A simpler alternative which we adopt is to use n-gram features collected from the document to rerank candidate summaries obtained via beam decoding. We incorporate the features in a log-linear reranker whose feature weights are optimized with minimum error rate training (Och, 2003).

5 Experimental Setup

In this section we present our experimental setup for assessing the performance of our summarization models. We discuss the datasets used for training and evaluation, give implementation details, briefly introduce comparison models, and explain how system output was evaluated.

Datasets We trained our sentence- and word-based summarization models on the two datasets created from DailyMail news. Each dataset was split into approximately 90% for training, 5% for validation, and 5% for testing. We evaluated the models on the DUC-2002 single document summarization task. In total, there are 567 documents belonging to 59 different clusters of various news topics. Each document is associated with two versions of 100-word[7] manual summaries produced by human annotators. We also evaluated our models on 500 articles from the DailyMail test set (with the human authored highlights as goldstandard). We sampled article-highlight pairs so that the highlights include a minimum of 3 sentences. The average byte count for each document is 278.

Implementation Details We trained our models with Adam (Kingma and Ba, 2014) with initial learning rate 0.001. The two momentum parameters were set to 0.99 and 0.999 respectively. We performed mini-batch training with a batch size of 20 documents. All input documents were padded to the same length with an additional mask variable storing the real length for each document. The size of word, sentence, and document embeddings were set to 150, 300, and 750, respectively. For the convolutional sentence model, we followed Kim et al. (2016)[8] and used a list of kernel sizes $\{1, 2, 3, 4, 5, 6, 7\}$. For the recurrent document model and the sentence extractor, we used as regularization dropout with probability 0.5 on the LSTM input-to-hidden layers and the scoring layer. The depth of each LSTM module was 1. All LSTM parameters were randomly initialized over a uniform distribution within [-0.05, 0.05]. The word vectors were initialized with 150 dimensional pre-trained embeddings.[9]

Proper nouns pose a problem for embedding-based approaches, especially when these are rare

[5]A simpler model would use hard attention to select a sentence first and then a few words from it as a summary, but this would render the system non-differentiable for training. Although hard attention can be trained with the REINFORCE algorithm (Williams, 1992), it requires sampling of discrete actions and could lead to high variance.

[6]We empirically found that feeding the previous sentence-level attention vector as additional input to the LSTM would lead to small performance improvements. This is not shown in the equation.

[7]According to the DUC2002 guidelines `http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html`, the generated summary should be within 100 words.

[8]The CNN-LSTM architecture is publicly available at `https://github.com/yoonkim/lstm-char-cnn`.

[9]We used the *word2vec* (Mikolov et al., 2013) skip-gram model with context window size 6, negative sampling size 10 and hierarchical softmax 1. The model was trained on the *Google 1-billion* word benchmark (Chelba et al., 2014).

or unknown (e.g., at test time). Rush et al. (2015) address this issue by adding a new set of features and a log-linear model component to their system. As our model enjoys the advantage of generation by extraction, we can force the model to inspect the context surrounding an entity and its relative position in the sentence in order to discover extractive patterns, placing less emphasis on the meaning representation of the entity itself. Specifically, we perform named entity recognition with the package provided by Hermann et al. (2015) and maintain a set of randomly initialized entity embeddings. During training, the index of the entities is permuted to introduce some noise but also robustness in the data. A similar data augmentation approach has been used for reading comprehension (Hermann et al., 2015).

A common problem with extractive methods based on sentence labeling is that there is no constraint on the number of sentences being selected at test time. We address this by reranking the positively labeled sentences with the probability scores obtained from the softmax layer (rather than the label itself). In other words, we are more interested in is the relative ranking of each sentence rather than their exact scores. This suggests that an alternative to training the network would be to employ a ranking-based objective or a *learning to rank* algorithm. However, we leave this to future work. We use the three sentences with the highest scores as the summary (also subject to the word or byte limit of the evaluation protocol).

Another issue relates to the word extraction model which is challenging to batch since each document possesses a distinct vocabulary. We sidestep this during training by performing negative sampling (Mikolov et al., 2013) which trims the vocabulary of different documents to the same length. At each decoding step the model is trained to differentiate the true target word from 20 noise samples. At test time we still loop through the words in the input document (and a stop-word list) to decide which word to output next.

System Comparisons We compared the output of our models to various summarization methods. These included the standard baseline of simply selecting the "leading" three sentences from each document as the summary. We also built a sentence extraction baseline classifier using logistic regression and human engineered features. The classifier was trained on the same datasets as our neural network models with the following features: sentence length, sentence position, number of entities in the sentence, sentence-to-sentence cohesion, and sentence-to-document relevance. Sentence-to-sentence cohesion was computed by calculating for every document sentence its embedding similarity with every other sentence in the same document. The feature was the normalized sum of these similarity scores. Sentence embeddings were obtained by averaging the constituent word embeddings. Sentence-to-document relevance was computed similarly. We calculated for each sentence its embedding similarity with the document (represented as bag-of-words), and normalized the score. The word embeddings used in this baseline are the same as the pre-trained ones used for our neural models.

In addition, we included a neural abstractive summarization baseline. This system has a similar architecture to our word extraction model except that it uses an open vocabulary during decoding. It can also be viewed as a hierarchical document-level extension of the abstractive sentence summarizer proposed by Rush et al. (2015). We trained this model with negative sampling to avoid the excessive computation of the normalization constant.

Finally, we compared our models to three previously published systems which have shown competitive performance on the DUC2002 single document summarization task. The first approach is the phrase-based extraction model of Woodsend and Lapata (2010). Their system learns to produce highlights from parsed input (phrase structure trees and dependency graphs); it selects salient phrases and recombines them subject to length, coverage, and grammar constraints enforced via integer linear programming (ILP). Like ours, this model is trained on document-highlight pairs, and produces telegraphic-style bullet points rather than full-blown summaries. The other two systems, TGRAPH (Parveen et al., 2015) and URANK (Wan, 2010), produce more typical summaries and represent the state of the art. TGRAPH is a graph-based sentence extraction model, where the graph is constructed from topic models and the optimization is performed by constrained ILP. URANK adopts a unified ranking system for both single- and multi-document summarization.

Evaluation We evaluated the quality of the summaries automatically using ROUGE (Lin and Hovy, 2003). We report unigram and bigram over-

DUC 2002	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	43.6	21.0	40.2
LREG	43.8	20.7	40.3
ILP	45.4	21.3	42.8
NN-ABS	15.8	5.2	13.8
TGRAPH	48.1	**24.3**	—
URANK	**48.5**	21.5	—
NN-SE	47.4	23.0	**43.5**
NN-WE	27.0	7.9	22.8

DailyMail	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	20.4	7.7	11.4
LREG	18.5	6.9	10.2
NN-ABS	7.8	1.7	7.1
NN-SE	**21.2**	**8.3**	**12.0**
NN-WE	15.7	6.4	9.8

Table 1: ROUGE evaluation (%) on the DUC-2002 and 500 samples from the DailyMail.

lap (ROUGE-1,2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

In addition, we evaluated the generated summaries by eliciting human judgments for 20 randomly sampled DUC 2002 test documents. Participants were presented with a news article and summaries generated by a list of systems. These include two neural network systems (sentence- and word-based extraction), the neural abstractive system described earlier, the lead baseline, the phrase-based ILP model[10] of Woodsend and Lapata (2010), and the human authored summary. Subjects were asked to rank the summaries from best to worst (with ties allowed) in order of informativeness (does the summary capture important information in the article?) and fluency (is the summary written in well-formed English?). We elicited human judgments using Amazon's Mechanical Turk crowdsourcing platform. Participants (self-reported native English speakers) saw 2 random articles per session. We collected 5 responses per document.

6 Results

Table 1 (top half) summarizes our results on the DUC 2002 test dataset using ROUGE. NN-SE represents our neural sentence extraction model,

[10]We are grateful to Kristian Woodsend for giving us access to the output of his system. Unfortunately, we do not have access to the output of TGRAPH or URANK for inclusion in the human evaluation.

Models	1st	2nd	3rd	4th	5th	6th	MeanR
LEAD	0.10	0.17	0.37	0.15	0.16	0.05	3.27
ILP	0.19	0.38	0.13	0.13	0.11	0.06	2.77
NN-SE	0.22	0.28	0.21	0.14	0.12	0.03	2.74
NN-WE	0.00	0.04	0.03	0.21	0.51	0.20	4.79
NN-ABS	0.00	0.01	0.05	0.16	0.23	0.54	5.24
Human	0.27	0.23	0.29	0.17	0.03	0.01	2.51

Table 2: Rankings (shown as proportions) and mean ranks given to systems by human participants (lower is better).

NN-WE our word extraction model, and NN-ABS the neural abstractive baseline. The table also includes results for the LEAD baseline, the logistic regression classifier (LREG), and three previously published systems (ILP, TGRAPH, and URANK).

The NN-SE outperforms the LEAD and LREG baselines with a significant margin, while performing slightly better than the ILP model. This is an encouraging result since our model has only access to embedding features obtained from raw text. In comparison, LREG uses a set of manually selected features, while the ILP system takes advantage of syntactic information and extracts summaries subject to well-engineered linguistic constraints, which are not available to our models. Overall, our sentence extraction model achieves performance comparable to the state of the art without sophisticated constraint optimization (ILP, TGRAPH) or sentence ranking mechanisms (URANK). We visualize the sentence weights of the NN-SE model in the top half of Figure 4. As can be seen, the model is able to locate text portions which contribute most to the overall meaning of the document.

ROUGE scores for the word extraction model are less promising. This is somewhat expected given that ROUGE is *n*-gram based and not very well suited to measuring summaries which contain a significant amount of paraphrasing and may deviate from the reference even though they express similar meaning. However, a meaningful comparison can be carried out between NN-WE and NN-ABS which are similar in spirit. We observe that NN-WE consistently outperforms the purely abstractive model. As NN-WE generates summaries by picking words from the original document, decoding is easier for this model compared to NN-ABS which deals with an open vocabulary. The extraction-based generation approach is more robust for proper nouns and rare words, which pose a serious problem to open vocabulary mod-

sentence extraction:
a gang of at least three people poured gasoline on a car that stopped to fill up at *entity5* gas station early on Saturday morning and set the vehicle on fire
the driver of the car, who has not been identified, said he got into an argument with the suspects while he was pumping gas at a *entity13* in *entity14*
the group covered his white *entity16* in gasoline and lit it ablaze while there were two passengers inside
at least three people poured gasoline on a car and lit it on fire at a *entity14* gas station explosive situation
the passengers and the driver were not hurt during the incident but the car was completely ruined
the man's grandmother said the fire was lit after the suspects attempted to carjack her grandson, *entity33* reported
she said:' he said he was pumping gas and some guys came up and asked for the car
' they pulled out a gun and he took off running
' they took the gas tank and started spraying
' no one was injured during the fire , but the car 's entire front end was torched , according to *entity52*
the *entity53* is investigating the incident as an arson and the suspects remain at large
surveillance video of the incident is being used in the investigation
before the fire , which occurred at 12:15am on Saturday , the suspects tried to carjack the man hot case
the *entity53* is investigating the incident at the *entity67* station as an arson

word extraction:
gang poured gasoline in the car, *entity5* Saturday morning. the driver argued with the suspects. his grandmother said the fire was lit by the suspects attempted to carjack her grandson.

entities:
entity5:California *entity13*:76-Station *entity14*: South LA *entity16*:Dodge Charger *entity33*:ABC *entity52*:NBC *entity53*:LACFD *entity67*:LA76

Figure 4: Visualization of the summaries for a DailyMail article. The top half shows the relative attention weights given by the sentence extraction model. Darkness indicates sentence importance. The lower half shows the summary generated by the word extraction.

els. An example of the generated summaries for NN-WE is shown at the lower half of Figure 4.

Table 1 (lower half) also shows system results on the 500 DailyMail news articles (test set). In general, we observe similar trends to DUC 2002, with NN-SE performing the best in terms of all ROUGE metrics. Note that scores here are generally lower compared to DUC 2002. This is due to the fact that the gold standard summaries (aka highlights) tend to be more laconic and as a result involve a substantial amount of paraphrasing.

The results of our human evaluation study are shown in Table 2. Specifically, we show, proportionally, how often our participants ranked each system 1st, 2nd, and so on. Perhaps unsurprisingly, the human-written descriptions were considered best and ranked 1st 27% of the time, however closely followed by our NN-SE model which was ranked 1st 22% of the time. The ILP system was mostly ranked in 2nd place (38% of the time). The rest of the systems occupied lower ranks. We further converted the ranks to ratings on a scale of 1 to 6 (assigning ratings 6...1 to rank placements 1...6). This allowed us to perform Analysis of Variance (ANOVA) which revealed a reliable effect of system type. Specifically, post-hoc Tukey tests showed that NN-SE and ILP are significantly ($p < 0.01$) better than LEAD, NN-WE, and NN-ABS but do not differ significantly from each other or the human goldstandard.

7 Conclusions

In this work we presented a data-driven summarization framework based on an encoder-extractor architecture. We developed two classes of models based on sentence and word extraction. Our models can be trained on large scale datasets and learn informativeness features based on continuous representations without recourse to linguistic annotations. Two important ideas behind our work are the creation of hierarchical neural structures that reflect the nature of the summarization task and generation by extraction. The later effectively enables us to sidestep the difficulties of generating under a large vocabulary, essentially covering the entire dataset, with many low-frequency words and named entities.

Directions for future work are many and varied. One way to improve the word-based model would be to take structural information into account during generation, e.g., by combining it with a tree-based algorithm (Cohn and Lapata, 2009). It would also be interesting to apply the neural models presented here in a phrase-based setting similar to Lebret et al. (2015). A third direction would be to adopt an information theoretic perspective and devise a purely unsupervised approach that selects summary sentences and words so as to minimize information loss, a task possibly achievable with the dataset created in this work.

Acknowledgments

We would like to thank three anonymous reviewers and members of the ILCC at the School of Informatics for their valuable feedback. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*, San Diego, California.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th ACL*, pages 318–325, Hong Kong.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28*, pages 1171–1179. Curran Associates, Inc.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Trevor Anthony Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, pages 637–674.

Conroy and O'Leary. 2001. Text summarization via hidden Markov models. In *Proceedings of the 34th Annual ACL SIGIR*, pages 406–407, New Oleans, Louisiana.

Güneş Erkan and Dragomir R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of the 2004 EMNLP*, pages 365–371, Barcelona, Spain.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 104–111, Barcelona, Spain.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th ACL*, Berlin, Germany. to appear.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th ACL*, Berlin, Germany. to appear.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1684–1692. Curran Associates, Inc.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the 30th AAAI*, Phoenix, Arizon. to appear.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 EMNLP*, pages 1746–1751, Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. 2015. Summarization based on embedding distributions. In *Proceedings of the 2015 EMNLP*, pages 1984–1989, Lisbon, Portugal.

Julian Kupiec, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR*, pages 68–73, Seattle, Washington.

Rémi Lebret, Pedro O Pinheiro, and Ronan Collobert. 2015. Phrase-based image captioning. In *Proceedings of the 32nd ICML*, Lille, France.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Proceedings of HLT NAACL*, pages 71–78, Edmonton, Canada.

Rada Mihalcea. 2005. Language independent extractive summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 49–52, Ann Arbor, Michigan.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th Annual ACM SIGIR*, pages 573–580, Washington, Seattle.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st ACL*, pages 160–167, Sapporo, Japan.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 EMNLP*, pages 1949–1954, Lisbon, Portugal, September.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization. Technical report, Columbia University Academic Commons.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 EMNLP*, pages 379–389, Lisbon, Portugal.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Krysta Svore, Lucy Vanderwende, and Christopher Burges. 2007. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 EMNLP-CoNLL*, pages 448–457, Prague, Czech Republic.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pages 2674–2682. Curran Associates, Inc.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd COLING*, pages 1137–1145.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th ACL*, pages 565–574, Uppsala, Sweden.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the 2015 EMNLP*, pages 1961–1966, Lisbon, Portugal.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of 2014 EMNLP*, pages 670–680, Doha, Qatar.