Extractive Summarization with SWAP-NET: Sentences and Words from Alternating Pointer Networks

Aishwarya Jadhav
Indian Institute of Science
Bangalore, India
aishwaryaj@iisc.ac.in

Vaibhav Rajan
School of Computing
National University of Singapore
vaibhav.rajan@nus.edu.sg

Abstract

We present a new neural sequence-to-sequence model for extractive summarization called SWAP-NET (Sentences and Words from Alternating Pointer Networks). Extractive summaries comprising a salient subset of input sentences, often also contain important key words. Guided by this principle, we design SWAP-NET that models the interaction of key words and salient sentences using a new two-level pointer network based architecture. SWAP-NET identifies both salient sentences and key words in an input document, and then combines them to form the extractive summary. Experiments on large scale benchmark corpora demonstrate the efficacy of SWAP-NET that outperforms state-of-the-art extractive summarizers.

1 Introduction

Automatic summarization aims to shorten a text document while maintaining the salient information of the original text. The practical need for such systems is growing with the rapid and continuous increase in textual information sources in multiple domains.

Summarization tools can be broadly classified into two categories: extractive and abstractive. *Extractive* summarization selects parts of the input document to create its summary while *abstractive* summarization generates summaries that may have words or phrases not present in the input document. Abstractive summarization is clearly harder as methods have to address factual and grammatical errors that may be introduced and problems in utilizing external knowledge sources to obtain paraphrasing or generalization. Extractive summarizers obviate the need to solve these problems by selecting the most salient textual units (usually sentences) from the input documents. As a result, they generate summaries that are grammatically and semantically more accurate than those from abstractive methods. While they may have problems like incorrect or unclear referring expressions or lack of coherence, they are computationally simpler and more efficient to generate. Indeed, state-of-the-art extractive summarizers are comparable or often better in performance to competitive abstractive summarizers (see (Nallapati et al., 2017) for a recent empirical comparison).

Classical approaches to extractive summarization have relied on human-engineered features from the text that are used to score sentences in the input document and select the highest-scoring sentences. These include graph or constraint-optimization based approaches as well as classifier-based methods. A review of these approaches can be found in Nenkova et al. (2011). Some of these methods generate summaries from multiple documents. In this paper, we focus on single document summarization.

Modern approaches that show the best performance are based on end-to-end deep learning models that do not require human-crafted features. Neural models have tremendously improved performance in several difficult problems in NLP such as machine translation (Chen et al., 2017) and question-answering (Hao et al., 2017). Deep models with thousands of parameters require large, labeled datasets and for summarization this hurdle of labeled data was surmounted by Cheng and Lapata (2016), through the creation of a labeled dataset of news stories from CNN and Daily Mail consisting of around 280,000 documents and human-generated summaries.

Recurrent neural networks with *encoder-decoder* architecture (Sutskever et al., 2014) have

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 142–151
Melbourne, Australia, July 15 - 20, 2018.

been successful in a variety of NLP tasks where an encoder obtains representations of input sequences and a decoder generates target sequences. Attention mechanisms (Bahdanau et al., 2015) are used to model the effects of different loci in the input sequence during decoding. Pointer networks (Vinyals et al., 2015) use this mechanism to obtain target sequences wherein each decoding step is used to *point* to elements of the input sequence. This pointing ability has been effectively utilized by state-of-the-art extractive and abstractive summarizers (Cheng and Lapata, 2016; Nallapati et al., 2016; See et al., 2017).

In this work, we design SWAP-NET a new deep learning model for extractive summarization. Similar to previous models, we use an encoder-decoder architecture with attention mechanism to select important sentences. Our key contribution is to design an architecture that utilizes key words in the selection process. Salient sentences of a document, that are useful in summaries, often contain key words and, to our knowledge, none of the previous models have explicitly modeled this interaction. We model this interaction through a two-level encoder and decoder, one for words and the other for sentences. An attention-based mechanism, similar to that of Pointer Networks, is used to learn important words and sentences from labeled data. A switch mechanism is used to select between words and sentences during decoding and the final summary is generated using a combination of selected sentences and words. We demonstrate the efficacy of our model on the CNN/Daily Mail corpus where it outperforms state-of-the-art extractive summarizers. Our experiments also suggest that the semantic redundancy in SWAP-NET generated summaries is comparable to that of human-generated summaries.

2 Problem Formulation

Let D denote an input document, comprising of a sequence of N sentences: $s_1, \ldots, s_N$. Ignoring sentence boundaries, let $w_1, \ldots, w_n$ be the sequence of n words in document D. An extractive summary aims to obtain a subset of the input sentences that forms a salient summary.

We use the interaction between words and sentences in a document to predict important words and sentences. Let the target sequence of indices of important words and sentences be $V = v_1, \ldots, v_m$, where each index v_j can point to either a sentence or a word in an input document. We design a supervised sequence-to-sequence recurrent neural network model, SWAP-NET, that uses these target sequences (of sentences and words) to learn salient sentences and key words. Our objective is to find SWAP-NET model parameters M that maximize the probability $p(V|M, D) = \prod_j p(v_j|v_1, \ldots, v_{j-1}, M, D) = \prod_j p(v_j|v_{<j}, M, D)$. We omit M in the following to simplify notation. SWAP-NET predicts both key words and salient sentences, that are subsequently used for extractive summary generation.

3 Background

We briefly describe Pointer Networks (Vinyals et al., 2015). Our approach, detailed in the following sections, uses a similar attention mechanism.

Given a sequence of n vectors $X = x_1, ...x_n$ and a sequence of indices $R = r_1, ...r_m$, each between 1 and n, the Pointer Network is an encoder-decoder architecture trained to maximize $p(R|X; \theta) = \prod_{j=1}^{m} p_\theta(r_j|r_1, ...r_{j-1}, X; \theta)$, where θ denotes the model parameters. Let the encoder and decoder hidden states be $(e_1, ..., e_n)$ and $(d_1, ..., d_m)$ respectively. The attention vector at each output step j is computed as follows:

$$u_i^j = v^T \tanh(W_e e_i + W_d d_j), \ i \in (1, \ldots, n)$$

$$\alpha_i^j = \text{softmax}(u_i^j), \ i \in (1, \ldots, n)$$

The softmax normalizes vector u^j to be an *attention* mask over inputs. In a pointer network, the same attention mechanism is used to select one of the n input vectors with the highest probability, at each decoding step, thus effectively *pointing* to an input:

$$p(r_j|r_1, ...r_{j-1}, X) = \text{softmax}(u^j)$$

Here, v, W_d, and W_e are learnable parameters of the model.

4 SWAP-NET

We use an encoder-decoder architecture with an attention mechanism similar to that of Pointer Networks. To model the interaction between words and sentences in a document we use two encoders and decoders, one at the word level and the other at the sentence level. The sentence-level decoder learns to *point* to important sentences while the

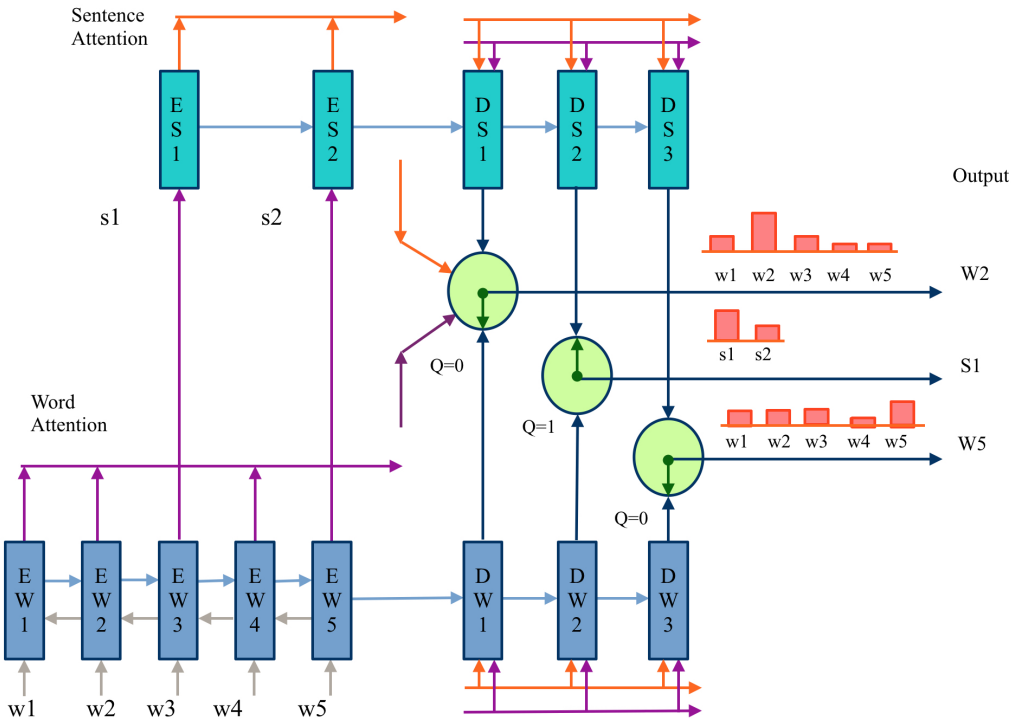

Figure 1: SWAP-NET architecture. EW: word encoder, ES: sentence encoder, DW: word decoder, DS: sentence decoder, Q: switch. Input document has words $[w1, \ldots, w5]$ and sentences $[s1, s2]$. Target sequence shown: $v_1 = w2, v_2 = s1, v_3 = w5$. Best viewed in color.

word-level decoder learns to *point* to important words. A switch mechanism is trained to select either a word or a sentence at each decoding step. The final summary is created using the output words and sentences. We now describe the details of the architecture.

4.1 Encoder

We use two encoders: a bi-directional LSTM at the word level and a LSTM at the sentence level. Each word w_i is represented by a K-dimensional embedding (e.g., via word2vec), denoted by x_i. The word embedding x_i is encoded as e_i using bi-directional LSTM for $i = 1, \ldots, n$. The vector output of BiLSTM at the end of a sentence is used to represent that entire sentence, which is further encoded by the sentence-level LSTM as $E_k = \text{LSTM}(e_{k^l}, E_{k-1})$, where k^l is the index of the last word in the k^{th} sentence in D and E_k is the hidden state at the k^{th} step of LSTM, for $k = 1, \ldots, N$. See figure 1.

4.2 Decoder

We use two decoders – a sentence-level and a word-level decoder, that are both LSTMs, with each decoder *pointing* to sentences and words respectively (similar to a pointer network). Thus, we can consider the output of each decoder step to be an index in the input sequence to the encoder. Let m be the number of steps in each decoder. Let $T_1, \ldots, T_m$ be the sequence of indices generated by the sentence-level decoder, where each index $T_j \in \{1, \ldots, N\}$; and let $t_1, \ldots, t_m$ be the sequence of indices generated by the word-level decoder, where each index $t_j \in \{1, \ldots, n\}$.

4.3 Network Details

At the j^{th} decoding step, we have to select a sentence or a word which is done through a binary switch Q_j that has two states $Q_j = 0$ and $Q_j = 1$ to denote word and sentence selection respectively. So, we first determine the switch probability $p(Q_j|v_{<j}, D)$. Let α^s_{kj} denote the probability of selecting the k^{th} input sentence at the j^{th} decoding step of sentence decoder:

$$\alpha^s_{kj} = p(T_j = k|v_{<j}, Q_j = 1, D),$$

and let α^w_{ij} denote the probability of selecting the i^{th} input word at the j^{th} decoding step of word decoder:

$$\alpha^w_{ij} = p(t_j = i|v_{<j}, Q_j = 0, D),$$

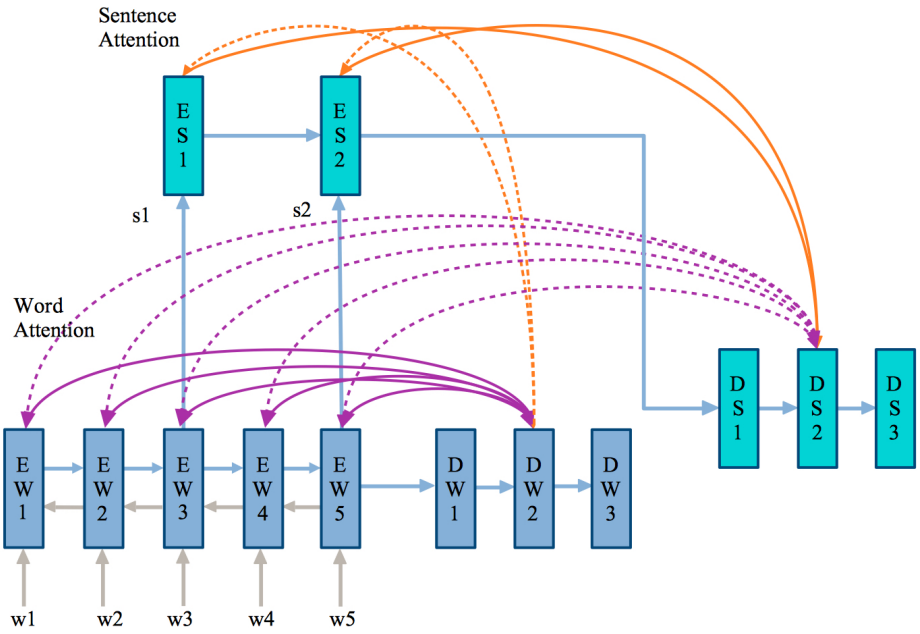

Figure 2: Illustration of word and sentence level attention in the second decoder step (Eq. 1 and Eq. 2). Purple: attention on words, Orange: attention on sentences, Unidirectional dotted arrows: attention from previous step, Bidirectional arrows: attention from previous and to present step. Best viewed in color.

both conditional on the corresponding switch selection. We set v_j based on the probability values:

$$v_j = \begin{cases} k = \arg\max_k p^s_{kj} & \text{if } \max_k p^s_{kj} > \max_i p^w_{ij} \\ i = \arg\max_i p^w_{ij} & \text{if } \max_i p^w_{ij} > \max_k p^s_{kj} \end{cases}$$

$$p^s_{kj} = \alpha^s_{kj} p(Q_j = 1 | v_{<j}, D),$$

$$p^w_{ij} = \alpha^w_{ij} p(Q_j = 0 | v_{<j}, D).$$

These probabilities are obtained through the attention weight vectors at the word and sentence levels and the switch probabilities:

$$\alpha^w_{ij} = \text{softmax}(v_t^T \phi(w_h h_j + w_t e_i)),$$

$$\alpha^s_{kj} = \text{softmax}(V_T^T \phi(W_H H_j + W_T E_k)).$$

Parameters v_t, w_h, w_t, V_T, W_H and W_T are trainable parameters. Parameters h_j and H_j are the hidden vectors at the j^{th} step of the word-level and sentence-level decoder respectively defined as:

$$h_j = LSTM(h_{j-1}, a_{j-1}, \phi(A_{j-1})) \quad (1)$$

$$H_j = LSTM(H_{j-1}, A_{j-1}, \phi(a_{j-1})) \quad (2)$$

where $a_j = \sum_{i=0}^{n} \alpha^w_{ij} e_i$, $A_j = \sum_{k=0}^{N} \alpha^s_{kj} E_k$. The non-linear transformation, ϕ (we choose $\tanh$), is used to connect the word-level encodings to the sentence decoder and the sentence-level encodings to the word decoder. Specifically, the word-level decoder updates its state by considering a sum of sentence encodings, weighted by the attentions from the previous state and *mutatis mutandis* for the sentence-level decoder.

The switch probability $p(Q_j | v_{<j}, D)$ at the j^{th} decoding step is given by:

$$p(Q_j = 1 | v_{<j}, D) = \sigma(w_Q^T (H_{j-1}, A_{j-1}, \phi(h_{j-1}, a_{j-1})))$$

$$p(Q_j = 0 | v_{<j}, D) = 1 - p(Q_j = 1 | v_{<j}, D)$$

where w_Q is a trainable parameter and σ denotes the sigmoid function and ϕ is the chosen non-linear transformation ($\tanh$).

During training the loss function l_j at j^{th} step is set to $l_j = -\log(p^s_{kj} q^s_j + p^w_{ij} q^w_j) - \log p(Q_j | v_{<j}, D)$. Note that at each decoding step, switch is either $q^w_j = 1, q^s_j = 0$ if the j^{th} output is a word or $q^w_j = 0, q^s_j = 1$ if the j^{th} output is a sentence. The switch probability is also considered in the loss function.

4.4 Summary Generation

Given a document whose summary is to be generated, its sentences and words are given as input to the trained encoder. At the j^{th} decoding step, either a sentence or a word is chosen based on the probability values α^s_{kj} and α^w_{ij} and the switch probability $p(Q_j|v_{<j}, D)$. We assign importance scores to the selected sentences based on their probability values during decoding as well as the probabilities of the selected words that are present in the selected sentences. Thus sentences with words selected by the decoder are given higher importance. Let the k^{th} input sentence s_k be selected at the j^{th} decoding step and i^{th} input word w_i be selected at the l^{th} decoding step. Then the importance of s_k is defined as

$$I(s_k) = \alpha^s_{kj} + \lambda \sum_{w_i \in s_k} \alpha^w_{il} \tag{3}$$

In our experiments we choose $\lambda = 1$. The final summary consists of three sentences with the highest importance scores.

5 Related Work

Traditional approaches to extractive summarization rely on human-engineered features based on, for example, part of speech (Erkan and Radev, 2004) and term frequency (Nenkova et al., 2006). Sentences in the input document are scored using these features, ranked and then selected for the final summary. Methods used for extractive summarization include graph-based approaches (Mihalcea, 2005) and Integer Linear Programming (Gillick and Favre, 2009). There are many classifier-based approaches that select sentences for the extractive summary using methods such as Conditional Random Fields (Shen et al., 2007) and Hidden Markov models (Conroy and O'leary, 2001). A review of these classical approaches can be found in Nenkova et al. (2011).

End-to-end deep learning based neural models that can effectively learn from text data, without human-crafted features, have witnessed rapid development, resulting in improved performance in multiple areas such as machine translation (Chen et al., 2017) and question-answering (Hao et al., 2017), to name a few. Large labelled corpora based on news stories from CNN and Daily Mail, with human generated summaries have become available (Cheng and Lapata, 2016), that have spurred the use of deep learning models in summarization. Recurrent neural network based architectures have been designed for both extractive (Cheng and Lapata, 2016; Nallapati et al., 2017) and abstractive (See et al., 2017; Tan et al., 2017) summarization problems. Among these, the work of Cheng and Lapata (2016) and Nallapati et al. (2017) are closest to our work on extractive single-document summarization.

An encoder-decoder architecture with an attention mechanism similar to that of a pointer network is used by Cheng and Lapata (2016). Their hierarchical encoder uses a CNN at the word level leading to sentence representations that are used in an RNN to obtain document representations. They use a hierarchical attention model where the first level decoder predicts salient sentences used for an extractive summary and based on this output, the second step predicts keywords which are used for abstractive summarization. Thus they do not use key words for extractive summarization and for abstractive summarization they generate key words based on sentences predicted independently of key words. SWAP-NET, in contrast, is simpler using only two-level RNNs for word and sentence level representations in both the encoder and decoder. In our model we predict both words and sentences in such a way that their attentions interact with each other and generate extractive summaries considering both the attentions. By modeling the interaction between these key words and important sentences in our decoder architecture, we are able to extract sentences that are closer to the gold summaries.

SummaRuNNer, the method developed by Nallapati et al. (2017) is not similar to our method in its architecture but only in the aim of extractive summary generation. It does not use an encoder-decoder architecture; instead it is an RNN based binary classifier that decides whether or not to include a sentence in the summary. The RNN is multi-layered representing inputs, words, sentences and the final sentence labels. The decision of selecting a sentence at each step of the RNN is based on the content of the sentence, salience in the document, novelty with respect to previously selected sentences and other positional features. Their approach is considerably simpler than that of Cheng and Lapata (2016) but obtains summaries closer to the gold summaries, and additionally, facilitates interpretable visualization and

training from abstractive summaries. Their experiments show improved performance over both abstractive and extractive summarizers from several previous models (Nallapati et al., 2017).

We note that several elements of our architecture have been introduced and used in earlier work. Pointer networks (Vinyals et al., 2015) used the attention mechanism of (Bahdanau et al., 2015) to solve combinatorial optimization problems. They have also been used to *point* to sentences in extractive (Cheng and Lapata, 2016) and abstractive (Nallapati et al., 2016; See et al., 2017) summarizers. The switch mechanism was introduced to incorporate rare or out-of-vocabulary words (Gulcehre et al., 2016) and are used in several summarizers (e.g. (Nallapati et al., 2016)). However, we use it to select between word and sentence level decoders in our model.

The importance of all the three interactions: (i) sentence-sentence, (ii) word-word and (iii) sentence-word, for summarization, have been studied by Wan et al. (2007) using graph-based approaches. In particular, they show that methods that account for saliency using both the following considerations perform better than methods that consider either one of them alone, and SWAP-NET is based on the same principles.

- A sentence should be salient if it is heavily linked with other salient sentences, and a word should be salient if it is heavily linked with other salient words.
- A sentence should be salient if it contains many salient words, and a word should be salient if it appears in many salient sentences.

6 Data and Experiments

6.1 Experimental Settings

In our experiments the maximum number of words per document is limited to 800, and the maximum number of sentences per document to 50 (padding is used to maintain the length of word sequences). We also use the symbols <GO> and <EOS> to indicate start and end of prediction by decoders. The total vocabulary size is 150,000 words.

We use word embeddings of dimension 100 pretrained using word2vec (Mikolov et al., 2013) on the training dataset. We fix the LSTM hidden state size at 200. We use a batch size of 16 and the ADAM optimizer (Kingma and Ba, 2015) with parameters: learning rate = 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$ to train SWAP-NET. We employ gradient clipping to regularize our model and an early stopping criterion based on the validation loss.

During training we find that SWAP-NET learns to predict important sentences faster than to predict words. To speed up learning of word probabilities, we add the term $-\log \alpha_{ij}^{w}$ to our loss function l_j in the final iterations of training. It is possible to get the same sentence or word in multiple (usually consecutive) decoding steps. In that case, in Eq. 3 we consider the maximum value of alpha obtained across these steps and calculate maximum scores of distinct sentences and words.

We select 3 top scoring sentences for the summary, as there are 3.11 sentences on average in the gold summary of the training set (similar to settings used by others, e.g., (Narayan et al., 2017)).

6.2 Baselines

Two state-of-the-art methods for extractive summarization are **SummaRuNNer** (Nallapati et al., 2017) and **NN**, the neural summarizer of Cheng and Lapata (2016). SummaRuNNer can also provide extractive summaries while being trained abstractively (Nallapati et al., 2017); we denote this method by **SummaRuNNer-abs**. In addition, we compare our method with the **Lead-3** summary which consists of the first three sentences from each document. We also compare our method with an abstractive summarizer that uses a similar attention-based encoder-decoder architecture (Nallapati et al., 2016), denoted by **ABS**.

6.3 Benchmark Datasets

For our experiments, we use the CNN/DailyMail corpus (Hermann et al., 2015). We use the anonymized version of this dataset, from Cheng and Lapata (2016), which has labels for important sentences, that are used for training. To obtain labels for words, we extract keywords from each gold summary using RAKE, an unsupervised keyword extraction method (Rose et al., 2010). These keywords are used to label words in the corresponding input document during training. We replace numerical values in the documents by zeros to limit the vocabulary size.

We have 193,986 training documents, 12,147 validation documents and 10,346 test documents from the DailyMail corpus and 83,568 training documents, 1,220 validation documents and 1,093 test documents from CNN subset with labels for sentences and words.

6.4 Evaluation Metrics

We use the ROUGE toolkit (Lin and Hovy, 2003) for evaluation of the generated summaries in comparison to the gold summaries. We use three variants of this metric: ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) that are computed by matching unigrams, bigrams and longest common subsequences respectively between the two summaries. To compare with (Cheng and Lapata, 2016) and (Nallapati et al., 2017) we use limited length ROUGE recall at 75 and 275 bytes for the Daily-Mail test set, and full length ROUGE-F1 score, as reported by them.

6.5 Results on Benchmark Datasets

Performance on Daily Mail Data

Models	R1	R2	RL
Lead-3	21.9	7.2	11.6
NN	22.7	8.5	12.5
SummaRuNNner-abs	23.8	9.6	13.3
SummaRuNNner	26.2	**10.8**	**14.4**
SWAP-NET	**26.4**	10.7	**14.4**

Table 1: Performance on Daily-Mail test set using the limited length recall of Rouge at 75 bytes.

Models	R1	R2	RL
Lead-3	40.5	14.9	32.6
NN	42.2	17.3	34.8
SummaRuNNner-abs	40.4	15.5	32.0
SummaRuNNner	42.0	16.9	34.1
SWAP-NET	**43.6**	**17.7**	**35.5**

Table 2: Performance on Daily-Mail test set using the limited length recall of Rouge at 275 bytes.

Table 1 shows the performance of SWAP-NET, state-of-the-art baselines NN and SummaRuNNer and other baselines, using ROUGE recall with summary length of 75 bytes, on the entire Daily Mail test set. The performance of SWAP-NET is comparable to that of SummaRuNNer and better than NN and other baselines. Table 2 compares the same algorithms using ROUGE recall with summary length of 275 bytes. SWAP-NET outperforms both state-of-the-art summarizers SummaRuNNer as well as NN.

Performance on CNN/DailyMail Data

SWAP-NET has the best performance on the combined CNN and Daily Mail corpus, outperforming the previous best reported F-score by SummaRuNNer, as seen in table 3, with a consistent improvement of over 2 ROUGE points in all three metrics.

Models	R1	R2	RL
Lead-3	39.2	15.7	35.5
ABS	35.4	13.3	32.6
SummaRuNNer-abs	37.5	14.5	33.4
SummaRuNNer	39.6	16.2	35.3
SWAP-NET	**41.6**	**18.3**	**37.7**

Table 3: Performance on CNN and Daily-Mail test set using the full length Rouge F score.

6.6 Discussion

SWAP-NET outperforms state-of-the-art extractive summarizers SummaRuNNer (Nallapati et al., 2017) and NN (Cheng and Lapata, 2016) on benchmark datasets. Our model is similar, although simpler, than that of NN and the main difference between SWAP-NET and these baselines is its explicit modeling of the interaction between key words and salient sentences.

Automatic keyword extraction has been studied extensively (Hasan and Ng, 2014). We use a popular and well tested method, RAKE (Rose et al., 2010) to obtain key words in the training documents. A disadvantage with such methods is that they do not guarantee representation, via extracted keywords, of all the topics in the text (Hasan and Ng, 2014). So, if RAKE key words are directly applied to the input test document (without using word decoder trained on RAKE words, obtained from gold summary as done in SWAP-NET), then there is a possibility of missing sentences from the missed topics. So, we train SWAP-NET to predict key words and also model their interactions with sentences.

Statistics	Lead-3	SWAP-NET
KW coverage	61.6%	73.8%
Sentences with KW	92.2%	98%

Table 4: Key word (KW) statistics per summary (average percentage) from 500 documents in Daily Mail test set. See text for definitions.

We investigate the importance of modeling this interaction and the role of key words in the final summary. Table 4 shows statistics that reflect the importance of key words in extractive summaries. *Key word coverage* measures the proportion of key

Title: @entity19 vet surprised reason license plate denial
Gold Summary: @entity9 of @entity10 , @entity1 , wanted to get ' @entity11 - 0 ' put on a license plate . that would have commemorated both @entity9 getting the @entity8 in 0 and his @entity16 . the @entity1 @entity21 denied his request , citing state regulations prohibiting the use of the number 0 because of its indecent connotations .
SWAP-NET Summary: **@entity9 of @entity10 wanted to get ' @entity11 ' put on a license plate** , the @entity14 newspaper of @entity10 reported . **that would have commemorated both @entity9 getting the @entity8 in 0 and his @entity16** , according to the newspaper . **the @entity1 @entity21 denied his request , citing state regulations prohibiting the use of the number 0 because of its indecent connotations** @entity9 had been an armored personnel carrier 's gunner during his time in the @entity29 .
SWAP-NET Key words: @entity1, @entity9, @entity8, citing, number, year, indecent, personalized, war, surprised, plate, @entity14, @entity11, @entity10, regulations, reported, wanted, connotations, license, request, according,@entity21, armored, @entity16
Lead 3 Summary: a @entity19 war veteran in @entity1 has said he 's surprised over the reason for the denial of his request for a personalized license plate commemorating the year he was wounded and awarded a @entity8 . **@entity9 of @entity10 wanted to get ' @entity11 ' put on a license plate** , the @entity14 newspaper of @entity10 reported . **that would have commemorated both @entity9 getting the @entity8 in 0 and his @entity16** , according to the newspaper .

Table 5: Sample gold summary and summaries generated by SWAP-NET and Lead-3. Key words are highlighted, bold font indicates overlap with gold summary.

words from those in the gold summary present in the generated summary. SWAP-NET obtains nearly 74% of the key words. In comparison Lead-3 has only about 62% of the key words from the gold summary.

Sentences with key words measures the proportion of sentences containing at least one key word. It is not surprising that in SWAP-NET summaries 98% of the sentences, on average, contain at least one key word: this is by design of SWAP-NET. However, note that Lead-3 which has poorer performance in all the benchmark datasets has much fewer sentences with key words. This highlights the importance of key words in finding salient sentences for extractive summaries.

Gold summary	Lead-3	SWAP-NET
0.81	0.553	0.8

Table 6: Average pairwise cosine distance between paragraph vector representations of sentences in summaries.

We also find the SWAP-NET obtains summaries that have less semantic redundancy. Table 6 shows the average distance between pairs of sentences from the gold summary, and summaries generated from SWAP-NET and Lead-3. Distances are measured using cosine distance of paragraph vectors of each sentence (Le and Mikolov, 2014) from randomly selected 500 documents of the Daily Mail test set. Paragraph vectors have been found to be effective semantic representations of sentences (Le and Mikolov, 2014) and experiments in (Dai et al., 2015) also show that paragraph vectors can be effectively used to measure semantic similarity using cosine distance. For training we use GENSIM (Řehůřek and Sojka, 2010) with embedding size 200 and initial learning rate 0.025. The model is trained on 500 documents from DailyMail dataset for 10 epochs and learning rate is decreased by 0.002 at each epoch.

The average pair-wise distance of SWAP-NET is very close to that of the gold summary, both

nearly 0.8. In contrast, the average pairwise distance in Lead-3 summaries is 0.553 indicating higher redundancy. This highly desirable feature of SWAP-NET is likely due to use of of key words, that is affecting the choice of sentences in the final summary.

Table 5 shows a sample gold summary from the Daily Mail dataset and the generated summary from SWAP-NET and, for comparison, from Lead-3. We observe the presence of key words in all the overlapping segments of text with the gold summary indicating the importance of key words in finding salient sentences. Modeling this interaction, we believe, is the reason for the superior performance of SWAP-NET in our experiments.

An implementation of SWAP-NET and all the generated summaries from the test sets are available online in a github repository[1].

7 Conclusion

We present SWAP-NET, a neural sequence-to-sequence model for extractive summarization that outperforms state-of-the-art extractive summarizers SummaRuNNer (Nallapati et al., 2017) and NN (Cheng and Lapata, 2016) on large scale benchmark datasets. The architecture of SWAP-NET is simpler than that of NN but due to its effective modeling of interaction between salient sentences and key words in a document, SWAP-NET achieves superior performance. SWAP-NET models this interaction using a new two-level pointer network based architecture with a switching mechanism. Our experiments also suggest that modeling sentence-keyword interaction has the desirable property of less semantic redundancy in summaries generated by SWAP-NET.

8 Acknowledgment

The authors thank the ACL reviewers for their valuable comments. Vaibhav Rajan acknowledges the support from Singapore Ministry of Education Academic Research Fund Tier 1 towards funding this research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations.*

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 406–407. ACM.

Andrew M Dai, Christopher Olah, and Quoc V Le. 2015. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, Association for Computational Linguistics, pages 10–18.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1262–1273.

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*, pages 1693–1701.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[1] https://github.com/aishj10/swap-net

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*, pages 1188–1196.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78.

Rada Mihalcea. 2005. Language independent extractive summarization. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 49–52.

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*, pages 3111–3119.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Nikos Papasarantopoulos, Shay B Cohen, and Mirella Lapata. 2017. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

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 International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 573–580.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *Proceedings of International Joint Conferences on Artificial Intelligence*, volume 7, pages 2862–2867.

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

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1171–1181.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 552–559.