

Incorporating Topic Sentence on Neural News Headline Generation

Jan Wira Gotama Putra¹, Hayato Kobayashi^{2,3}, Nobuyuki Shimizu²

¹ Tokyo Institute of Technology

² Yahoo Japan Corporation

³ RIKEN AIP

gotama.w.aa@m.titech.ac.jp, {hakobaya, nobushim}@yahoo-corp.jp

Abstract. Most past studies on neural news headline generation trained the encoder-decoder model using the first sentence of a document aligned with a headline. However, it is found that the first sentence might not provide sufficient information. We propose using a topic sentence as the input instead of the first sentence for neural news headline generation task. The topic sentence is considered as the most newsworthy sentence and has been studied in the past. Experimental results show that the model trained on the topic sentence has a better generalizability than the model trained using only the first sentence. Training the model using both the first and topic sentences increases the performance even further compared to only training using the topic sentence in certain cases. We conclude that using a topic sentence, while keeping input length as short as possible at the same time, is a preferred strategy for providing more informative information to the neural network compared to using just the first sentence.

Keywords: headline generation, topic sentence, encoder-decoder, summarization

1 Introduction

Automatic summarization aims to understand and condense document(s) into a shorter version while preserving the content in a readable fashion [1,2]. Headline generation is one variant of the summarization tasks, which was introduced in DUC-2003 and DUC-2004 (Task 1) [3,4,5]. A headline can be viewed as a summary (succinct) of document(s) (verbose) [3]. Headline generation can be applied (but not limited) to spoken broadcast news [6] or multi-document summarization (e.g., summarizing news from multiple sources) where a headline is not present [5,7]. In practice, an automatic headline generation system can act as a supporting system where a person writes a headline considering the suggestion from an automatically generated one.

In this study, we are interested in a single document (news) headline generation. There are two approaches for the news headline generation. Firstly, an *extractive* approach generates headlines by selecting one or more important

units (words, phrases, events, sentences) then transforms, combines, or compresses them altogether to satisfy the length constraint of a headline [4,8]. The second approach is an *abstractive* in that it views the headline generation task as a natural language generation problem [3]. The generated headline in this approach possibly includes unseen words in the source document [3,9,10]. The abstractive approach is more desirable since it more closely simulates humans work.

In the abstractive approach, headline generation can be cast as a task of mapping an input sequence of words into a target sequence of words using an encoder-decoder model (Sect. 2) [10,11,12,13,14,15,16]. Most existing studies utilize the first sentence of a news document as input for the encoder-decoder model. They focused on incorporating linguistics information or architectural strategies of the encoder-decoder, while relatively little emphasis has been given on how to choose the input (Sect. 3).

We experiment on using topic sentences as input. A topic sentence is considered to be the most newsworthy sentence in a news document, and its selection is based on sentence linguistic information (Sect. 4) [17,18,19]. We propose feeding the topic sentence to the encoder-decoder model or feeding it as input alongside the first sentence of a news article. There are two key questions addressed in this work: (1) Is the topic sentence more useful than the first sentence for headline generation, and (2) Is the topic sentence helpful in addition to the first sentence for headline generation.

2 Encoder-Decoder Model

An encoder-decoder model maps a sequence of input into a sequence of output [20,21]. Let us denote $\mathbf{x} = (x_1, \dots, x_N)$ as a sequence of N input words. The headline generation task aims to find the best sequence of M words $\mathbf{y} = (y_1, \dots, y_M)$, $M < N$; for the given input sequence \mathbf{x} [10]. This means modeling the conditional probability $p(\mathbf{y} | \mathbf{x})$ of an input-output pair. Usually, additional parameters θ are involved in governing the conditional probability. Therefore, the conditional probability is transformed into $p(\mathbf{y} | \mathbf{x}, \theta)$ which can be factored into ordered conditionals as shown in Equation (1):

$$p(\mathbf{y} | \mathbf{x}, \theta) = \prod_{t=1}^M p(y_t | \{y_1, \dots, y_{t-1}\}, \mathbf{x}, \theta), \quad (1)$$

where M is the length of the output. This formulation can naturally be cast as an encoder-decoder model where the neural network acts as parameters θ .

An encoder encodes the input as a single representation \mathbf{c} (Sect. 2.1). Then, the decoder decodes this representation to generate a sequence of output words \mathbf{y} (Sect. 2.2). Input-output pair examples are fed into the encoder-decoder model, which is trained to tune the value for parameters θ governing the conditional probability. This is typically achieved by minimizing the negative log likelihood

of the conditional probability over a set of training data D , as shown in Equation (2):

$$\mathcal{L} = - \sum_{(\mathbf{x}, \mathbf{y}) \in D} p(\mathbf{y} | \mathbf{x}, \theta), \quad (2)$$

The minimization process typically uses stochastic gradient descent method. Once the model is trained, the encoder-decoder generates a headline \mathbf{y}^* by finding the most probable sequence of output for a new input sequence \mathbf{x} , as shown in Equation (3):

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} p(\mathbf{y} | \mathbf{x}, \theta). \quad (3)$$

The decoding process typically uses beam search algorithm.

2.1 Encoder

An encoder represents an input sequence of N words $\mathbf{x} = (x_1, \dots, x_N)$ into a single vector representation \mathbf{c} . A word x_t is typically represented as a one-hot vector encoding or as a corresponding embedding vector (denoted as \mathbf{e}_t). Past studies proposed using a recurrent neural network (RNN) encoder as shown in Equation (4) [11,12,13,15]:

$$\begin{aligned} \mathbf{h}_t &= f(\mathbf{h}_{t-1}, \mathbf{e}_t) \\ &= f(\mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{U}_e \mathbf{e}_t), \end{aligned} \quad (4)$$

where f is a non-linear activation function; \mathbf{U}_h and \mathbf{U}_e are weight matrices. Depending on the current word vector \mathbf{e}_t and the previous hidden state \mathbf{h}_{t-1} , the RNN computes the current hidden state \mathbf{h}_t .

In practice, we can replace f using variant neural network choices for example, using long short-term memory (LSTM) as shown in Equation (5) [22]:

$$\begin{aligned} \mathbf{h}_t &= \phi(\mathbf{h}_{t-1}, \mathbf{e}_t) \\ &= \mathbf{o}_t \tanh(\mathbf{m}_t) \\ \mathbf{i}_t &= \sigma(\mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{W}_i \mathbf{e}_t) \\ \mathbf{o}_t &= \sigma(\mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{W}_o \mathbf{e}_t) \\ \mathbf{g}_t &= \sigma(\mathbf{U}_g \mathbf{h}_{t-1} + \mathbf{W}_g \mathbf{e}_t) \\ \mathbf{m}'_t &= \tanh(\mathbf{U}_m \mathbf{h}_{t-1} + \mathbf{W}_m \mathbf{e}_t) \\ \mathbf{m}_t &= \mathbf{g}_t \mathbf{m}_{t-1} + \mathbf{i}_t \mathbf{m}'_t \end{aligned} \quad (5)$$

where \mathbf{i}_t , \mathbf{o}_t , \mathbf{g}_t , and \mathbf{m}_t are the input gate, output gate, forget gate, and memory cell, respectively. \mathbf{U}_i , \mathbf{U}_o , \mathbf{U}_g , \mathbf{U}_m , \mathbf{W}_i , \mathbf{W}_o , \mathbf{W}_g , and \mathbf{W}_m are weight matrices. The last hidden state \mathbf{h}_N is considered as the input sequence representation \mathbf{c} [21]. Another way to compute \mathbf{c} is using the weighted sum of hidden states; $\mathbf{c} = q(\{\mathbf{h}_1, \dots, \mathbf{h}_N\})$, where q is a non-linear activation function.

The RNN may be replaced with a bidirectional RNN (BiRNN) to take into account both the preceding and following words for computing the hidden state

\mathbf{h}_t in question [12,13,15]. The BiRNN processes the hidden state \mathbf{h}_t as the concatenation of a forward hidden state \mathbf{h}_t^\rightarrow and a backward hidden state \mathbf{h}_t^\leftarrow . The forward hidden state \mathbf{h}_t^\rightarrow is computed as in a normal RNN, taking into account the current input \mathbf{e}_t and the previous hidden state $\mathbf{h}_{t-1}^\rightarrow$. Meanwhile, the backward hidden state \mathbf{h}_t^\leftarrow is computed by taking into account the current input \mathbf{e}_t and the next hidden state $\mathbf{h}_{t+1}^\leftarrow$.

2.2 Decoder

As described in the previous subsection, an encoder produces a single vector \mathbf{c} representing the entire input sequence. A decoder uses this representation to produce a sequence of output words $\mathbf{y} = (y_1, \dots, y_M)$. This output is regarded as a generated headline. A hidden state \mathbf{h}'_t represents a conditional probability distribution of words to select the current output y_t , which is typically represented in a vector form \mathbf{e}'_t . The hidden state \mathbf{h}'_t is computed by taking into account the input sequence \mathbf{c} , the generated word of the previous time step y_{t-1} (in its vector form \mathbf{e}'_{t-1}), and a decoder hidden state of the previous time step \mathbf{h}'_{t-1} . The hidden state \mathbf{h}'_t formulation is shown in Equation (6):

$$\begin{aligned} \mathbf{h}'_t &= f'(\mathbf{h}'_{t-1}, \mathbf{e}'_{t-1}, \mathbf{c}) \\ &= f'(\mathbf{U}'_h \mathbf{h}'_{t-1} + \mathbf{U}'_y \mathbf{e}'_{t-1} + \mathbf{U}'_c \mathbf{c}) \end{aligned} \quad (6)$$

where f' is a non-linear activation function; \mathbf{U}'_h , \mathbf{U}'_y , and \mathbf{U}'_c are weight matrices. f' can be replaced with variant neural network choices. In this way, the RNN encoder-decoder can model the factored input-output conditional probability previously shown in Equation (1).

An attention mechanism can also be taken into account to softly align input and output words [23,24]. It specifies which input word the decoder should focus on in a particular decoding step. With the attention mechanism, Equation (6) is transformed into Equation (7):

$$\mathbf{h}'_t = f'(\mathbf{h}'_{t-1}, \mathbf{e}'_{t-1}, \mathbf{c}, \mathbf{k}_t), \quad (7)$$

where \mathbf{k}_t denotes how much a particular hidden state of source input affects the t -th generated word. \mathbf{k}_t is typically computed in Equation (8):

$$\begin{aligned} \mathbf{k}_t &= \sum_{i=1}^N \alpha_{t,i} \mathbf{h}_i \\ \alpha_{t,i} &= \frac{\exp(\mathbf{h}_i \cdot \mathbf{h}'_{t-1})}{\sum_{z=1}^N \exp(\mathbf{h}_z \cdot \mathbf{h}'_{t-1})} \end{aligned} \quad (8)$$

where N is the length of input, \mathbf{h}_i is the encoder hidden state at time i , \mathbf{h}'_{t-1} is the decoder hidden state at time $t-1$.

3 Related Work

Past studies on neural headline generation mostly used the first sentence-headline pair as input-output for the encoder-decoder model [10,11,12,13,14,15]. They put more emphasis on how to incorporate linguistics information or architectural strategies of the encoder-decoder model. For example, Rush et al. [10] focused on how the encoder-decoder model can take into account word context. In addition to the word context, Chopra et al. [11] tried to incorporate information on the position of the input words as well. Nallapati et al. [13] and Takase et al. [14] tried to incorporate linguistic and structural information when representing the input sequence. Kikuchi et al. [12] was concerned with how to control the output length of the decoder. On the other hand, Ayana et al. [15] worked on comparing network architectures and training strategies.

These earlier studies used the first sentence of a news document as the input sequence. However, Tan et al. [16] questioned the effectiveness of using only the first sentence as the input because information in the text is distributed across sentences [5,7]. Tan et al. [16] used the full-document or sentences extracted by using statistical ranking techniques (e.g., LexRank, TextRank) as the input for the encoder-decoder. However, a long input sequence can possibly degrade the performance of the encoder-decoder [16,25]. It is relatively difficult to determine the saliency of sentences (i.e. processing unit) while tracking what has been said and what is still left to be said in a summary [16,26,27,28]. To address these existing problems, this work aims to use/add the topic sentence, where selection is based on linguistic information (Sect. 4), as the encoder-decoder model input.

4 Key Information in News

Ideally, news headline generation requires building a representation of understanding the 5W1H⁴ information of a news document [18]. However, extraction of the 5W1H information is relatively difficult. Instead, we used the topic sentence for the headline generation task as a proxy for the 5W1H information. It is defined as key information in news [19], as follows:

Topic sentence contains the core elements $\langle \textit{subject}, \textit{verb}, \textit{object} \rangle$ and at least one subordinate element ***time*** or ***location***.

The core functional elements of a sentence are $\langle \textit{subject}, \textit{verb} (\textit{predicate}), \textit{object} \rangle$ [17,18]. On top of these core elements, *time* and *location* are also important because they provide additional factual news information [18].

We hypothesize that incorporating the topic sentence is likely to provide better generalizability of the encoder-decoder model than only using the first sentence. Generalization means allowing the model to predict the headline of unseen news. We consider this strategy as a middle-ground for the presented problems (i.e., only using the first sentence is not enough, need to consider the

⁴ what, where, whom, when, who, how

5W1H information, and using the full-document/long input is relatively difficult). In contrast to statistical ranking techniques, the topic sentence considers the 5W1H information (indirectly) while the statistical ranking methods do not.

A recent study by Tan et al. [29] proposed an encoder-decoder model which jointly recognizes salient sentences and generates a summary. It will be interesting to investigate the properties of the salient sentences selected by their model with respect to the 5W1H information and the headline generation task, in future study.

5 Experimental Setting

5.1 Dataset

We used the annotated Gigaword dataset [30] (around 10 million documents) for our experiment. The annotation was only used for tokenization and sentence splitting during the pre-processing step. Pre-processing also included replacing digits by “#”, and replacing low-frequency words by “<unk>”, following the setting reported by Rush et al. [10]⁵. The first sentence, topic sentence, and original (reference) headline were extracted from each document.

We extracted the earliest sentence containing $\langle subject, verb, object \rangle$ and at least one subordinate element *time* or *location* as the topic sentence. This followed the inverted pyramid structure rationale for news stories that the earlier sentences contain more general information than later sentences. We analyzed the sentences using the dependency parser and named entity tagger in spaCy⁶ for a realistic setting. DATE and TIME tags were used for recognizing *time* information, and GPE (i.e., countries, cities, states) and LOC (non-GPE locations) tags were used for *location*⁷. We only extracted one topic sentence to keep the length of the input as small as possible to avoid the vanishing gradient problem. In the case where there were no sentences satisfying the requirements for a topic sentence, we used the first sentence as the topic sentence. In the case where topic sentence was the same as the first sentence, we only used the first sentence when feeding both first and topic sentences.

The dataset was split into training, validation, and testing data using the documents split provided by Rush et al. [10]. In the original script, a document whose headline consisted of more than three and fewer than 50 words was included. The first sentence also contained 10–100 words with at least one word in common with the corresponding headline. We added an additional filter to the topic sentence in a document, being that it should follow the same rule as the first sentence.

The remaining documents after filtering can be seen in Table 1. The “# docs” column denotes the number of documents. The “not found” column represents the percentage of cases when there were no sentences satisfying the topic sentence

⁵ <https://github.com/facebookarchive/NAMAS>

⁶ <https://spacy.io/>

⁷ <https://spacy.io/usage/linguistic-features#section-named-entities>

Table 1. Filtered gigaword dataset

Data	# docs	not found	found-1	found-2 ⁺⁺
Train	2,755,324	5.54%	73.43%	21.06%
Valid	139,284	5.69%	72.76%	21.58%
Test	134,432	5.90%	72.91%	21.19%

extraction rule. The “found-1” denotes the percentage of cases when the first sentence satisfied the topic sentence requirement. The “found-2⁺⁺” shows the percentage of cases when the topic sentence was not the first sentence of the document (e.g., 2nd, 3rd, or 4th sentence). The reference headline in this dataset consisted of eight tokens on average. While most of the topic sentences were the first sentence, the rest 21% (found-2⁺⁺) were meaningful for analysis compared to using only the first sentence directly as the input. In most cases, there was a sentence satisfying the topic sentence requirements. This proved that using the topic sentence for headline generation was possible in a real-world setting. We used ROUGE (ROUGE-1, ROUGE-2, and ROUGE-L) to measure the model performance [31].

5.2 Architectural Choice

We trained the encoder-decoder model using three variants of input: (1) first sentence, (2) topic sentence, and (3) both first and topic sentences in which each was aligned with a reference headline. We used the default encoder-decoder implementation available in Open Source Neural Machine Translation (OpenNMT) [32]⁸. This architecture is regarded as the standard sequence to sequence architecture. The reason for using this architecture was to investigate the consequence of using different input types, rather than an architectural choice in the neural headline generation task. The encoder and decoder were a 2-layer LSTM BiRNN and a 2-layer LSTM RNN, respectively with 500 hidden units. A global attention mechanism and dropout (0.3) were used. The network was trained using stochastic gradient descent with batch size 64 and an initial learning rate of 1.0. The learning rate decreased as the model converged. The number of maximum epochs was 13. We left the control for output length to the OpenNMT. To avoid a bias of parameter initialization, we trained five models for each input-output pair and presented the average performance in Sect. 6.

When we fed both first and topic sentences, we placed the first sentence first followed by the topic sentence later (separated by “.”). This placement emphasized that the topic sentence information was kept during the decoding process as it was closer in position to the decoder (the topic sentence was more important than the first sentence for the generalization purpose). This rationale was backed up by our empirical results.

⁸ <https://github.com/OpenNMT/OpenNMT-py>

Table 2. Evaluation result on full Gigaword test set. R denotes full-length *F1* ROUGE score and CR denotes copy rate score

Model	First				Topic				First and topic			
	R-1	R-2	R-L	CR	R-1	R-2	R-L	CR	R-1	R-2	R-L	CR
OF	40.83	20.32	37.97	0.81	29.45	12.06	26.97	0.72	23.26	7.90	20.89	0.69
OT	40.72	19.68	37.76	0.80	33.73	14.37	30.77	0.71	26.69	8.98	23.69	0.71
OFT	41.47	20.49	38.46	0.83	32.00	13.03	29.11	0.76	26.49	8.91	23.45	0.75

6 Experimental Result and Discussion

6.1 Full Gigaword

We fed each model using first, topic and the combination of both sentences as the input during testing on a Gigaword test set (134K). Details of the results are shown in Table 2 whose columns denote the corresponding testing input type. OF, OT, and OFT refer to our models trained on first, topic, and both first and topic sentences, respectively. All models produced seven tokens on average. OT performed the best when fed only using topic sentence, OFT performed the best when fed using just the first sentences, while OFT and OT performed comparably when fed a combination of first and topic sentences as input during testing. On the other hand, OF remained as the worst model across types of input.

We tested the difference in performance (five iterations) between pairs of models using a two-tailed t-test (two-sample unequal variance), and considered it statistically significant if the difference for all ROUGE scores between models was significant at $p < 0.05$. OFT significantly outperformed OT when fed using the first sentence. However, it did not significantly outperform OF with this input. Similarly, OF also did not significantly outperform OT when using the first sentence as input during testing. The difference in performance between the models was significant when fed using the topic sentence as input. When fed both first and topic sentences, both OFT and OT significantly outperformed OF, while the difference between OFT and OT was not significant. Based on this fact, we argued that the topic sentence can enhance the model performance across all types of input compared to when only using the first sentence as input during training. Therefore, the models trained by using a topic sentence have a better generalizability than the models trained using only the first sentence. This result is interesting in the sense that the performance was improved on the test set only by using different training input types.

Another interesting thing to note is the copy rate. A copy rate shows how much the model used the words found in input data as the headline words (computed using recall). OT and OF showed a relatively similar copy rate across the types of input while presenting different performance scores. OT and OF performed comparably when fed using only the first sentence despite the OT trained using the topic sentence (the difference was not statistically significant at $p < 0.05$). However, OT outperformed OF on other types of input. Simply said, copying words sourced from the topic sentence did a good job for the headline generation task.

Readers might wonder why the performance trend between OT and OFT was not consistent across all types of input. OFT outperformed OT when fed using the first sentence during testing, but not in other types of input. We inferred that there was a possibility of output words sourced from the topic sentence (in training data) despite fed using the unseen first sentence during testing (or vice versa). We supposed that OFT associated words in the first sentence with the words in the topic sentence during training while OT could not do so. When we fed both models using the topic sentence or both first and topic sentences, OFT used words in both first and topic sentences while OT only used words from the topic sentence resulting in a lower OFT’s performance score. When we tested using the first sentence, OFT probably copied words from the topic sentence as the result of association capability, resulting in a higher performance score. This explanation (although relatively weak) also supports our previous argument that copying words from the topic sentence was helpful in this task. This reasoning is relatively understandable since the RNN has been proven to work well in language modeling tasks.

Generally, the performance scores were higher when the models were fed using only the first sentence rather than other types of input during testing. The first sentence possibly contained more specific information than the document, which was useful for generating headlines (as a headline should express a particular news story), while the topic sentence was useful for the generalization purpose.

Some readers might argue that the performance of OFT was higher than OF and OT when fed using the first sentence only because more information was fed during training for OFT. In that case, the performance of OFT when fed using other types of input during testing should have been better than other models as well. In fact, this was not the case. Instead OFT was beaten by OT on other types of input. This means that training an encoder-decoder model using a longer input does not guarantee improvement in performance across all types of input. We have to provide an “optimal” input-output pair to train the neural network. In this experiment, it was relatively difficult to conclude which one was more optimal between only using the topic sentence and using both first and topic sentences as input for the training, as no trend was observed. We were only sure that using the topic sentence as opposed/in addition to the first sentence provided better generalizability.

6.2 Output Examples

To demonstrate the output of our models qualitatively, some randomly picked output examples during testing are shown. We show both output examples of short input (Fig. 1) and long input (Fig. 2). Each output of OF, OT, and OFT in these figures were taken from the best performing model among five iterations (ref. Sect. 5.2). **Ref** denotes the original (reference) headline.

Take a look at Fig. 1 showing the output examples (short input) when the models were fed using the first sentence⁹. Generally, OT and OFT were better in

⁹ We do not show the output examples when models were fed using the topic sentence due to space.

picking up factual information, e.g., “*information technology*” / “*it*” on example 1; “*fall #.# percent*” / “*since #####*” on example 2; and OF’s inability to produce “*iran*” on example 3. However, there were also cases where the incorrect factual information were reproduced, which was not just limited to OF. We acknowledge that the outputs were still far from perfect.

OF and OT tended to generate headlines according to the information it had seen in the corresponding input (paraphrasing the input). On the other hand OFT tended to be able to include other information which was not seen in the input as well; e.g., “*it*” is possibly a shortened form of “*information technology*” on example 1, the phrase “*first time*” on example 2. This is the main support for our argument in Sect. 6.1 that there is a possibility of outputting words sourced from the topic sentence (in training data) despite fed using the unseen first sentence during testing (or vice versa) for OFT (words association capability).

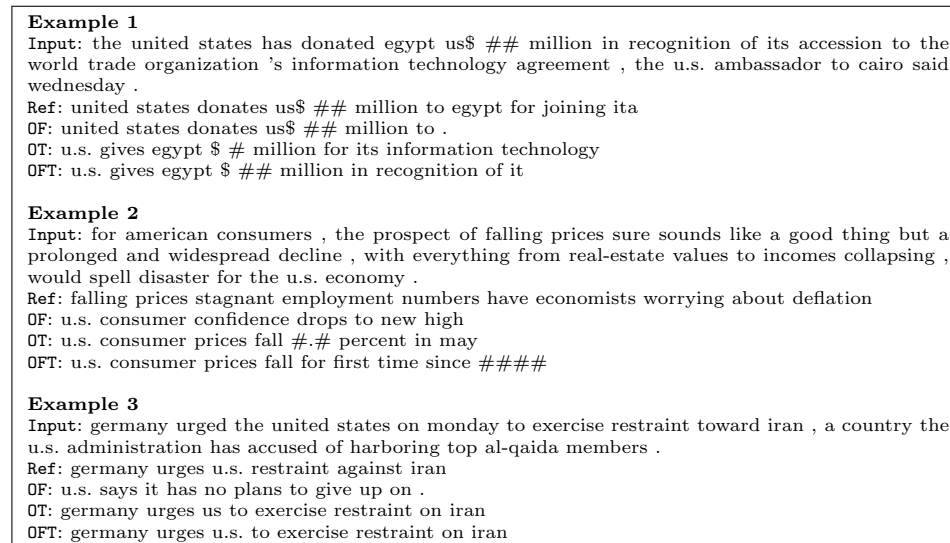


Fig. 1. Output Examples (Short Input)

Output examples when models were fed using long input (both first and topic sentences) is shown in Fig. 2. Similar to the trend of the short input, OT and OFT were better at picking up factual information than OF. Although the produced factual information could be incorrect (e.g., OT in example 1 in Fig. 2). There were cases (not always) in which the models were “confused” when fed using long input (e.g., example 2). This problem was not exclusive to OF and OT, but also happened to OFT as well. Certainly, it was more difficult to extract information from the longer input.

<p>Example 1 First Sentence: microsoft corp. has agreed to pay aol time warner \$ ### million to settle an antitrust suit over allegations that the software giant used strong-arm tactics to make its internet explorer the dominant web browser , displacing aol 's netscape . Topic Sentence: under thursday 's settlement , aol - time warner also gets free license to microsoft browsing software for seven years microsoft also would license its digital media technology to aol , as well as work with the company to promote digital media initiatives . Ref: microsoft to pay aol \$ ### million in settlement OF: microsoft agrees to pay aol time warner OT: microsoft to pay \$ # million to settle antitrust suit OFT: microsoft to pay \$ ### million to settle antitrust suit</p> <p>Example 2 First Sentence: michael jordan wo n't return as president of basketball operations for the washington wizards . Topic Sentence: team owner abe pollin made the decision after meeting with jordan on wednesday the move ends jordan 's # # years with the wizards , the last two as a player . Ref: michael jordan leaving wizards owner says OF: with . . . to be . OT: . to be . as new president OFT: new york 's . to be .</p>
--

Fig. 2. Output Examples (Long Input)

6.3 Additional Small Test Set

For the sake of comparison with existing models, we also evaluated our models using a commonly used small test set. The small test set comprised of 2000 first sentence - headline pairs sampled from the Gigaword dataset by Rush et al. [10].

Table 3. Evaluation result on 2000 first sentence - headline pairs of sampled Gigaword test set. R denotes full-length F1 ROUGE score

Model	R-1	R-2	R-L
This study			
OF	28.38	13.00	26.27
OT	28.77	12.69	26.40
OFT	29.37	13.13	27.08
Other Studies			
ABS+	29.78	11.89	26.97
words-lvt2k-1sent	32.67	15.59	30.64
OpenNMT Benchmark	33.13	16.09	31.00
RAS-Elman	33.78	15.96	31.15
MRT	36.54	16.59	31.15

Table 3 shows the results of the average full-length F1 ROUGE scores of our models (trained five times for each input type), which were evaluated using the small test set. We compared our models to the ABS+ [10], words-lvt2k-1sent [13], OpenNMT Benchmark¹⁰ (Equal as OF trained on 3.7M dataset) [32], Ras-Elman [11], and MRT [15]. Note that models from other studies¹¹ were

¹⁰ <http://forum.opennmt.net/t/text-summarization-on-gigaword-and-rouge-scoring/85>.

¹¹ Performance of models from other studies are taken directly from their papers.

trained using 3.7M training data sourced from the same annotated Gigaword corpus we used, which was pre-processed using the same script by Rush et al. [10]. However, as we added an additional filter into the preprocessing script (Sect. 5.1)¹², our models were trained using less data (2.7M). Despite this difference, our models performed competitively with the baseline (ABS+). It would be interesting to investigate the performance when models from other studies are trained using fewer 2.7M documents (as the result of our additional filter) in the future.

The difference in the performance of our models (OT, OF, and OFT) was not statistically significant at $p < 0.05$, although OFT showed a slightly higher score. This result was interesting in the sense that the performance improved on a standard test set even though the input type was different during training. This result suggests that abundant of information exists showing that the topic sentence can improve the performance over the testing data despite the difference of input type used in training.

7 Conclusion and Future Work

We experimented with incorporating topic sentence which has been well studied in the past, and gave an empirical proof that the topic sentence is useful for the headline generation task. We trained the encoder-decoder model using: (1) first sentence, (2) topic sentence, and (3) both first and topic sentences-headline pairs. We found that the model trained using the topic sentences has a better generalizability compared with that of the model trained using only the first sentence. Training the model using both the first and topic sentences increases the performance even further in certain cases. This fact proves that the topic sentence is useful for news headline generation task.

For future work, we will assess the difference in using topic sentences as opposed to other sentence selection/ranking methods. In addition, it will be interesting to investigate whether using other types of subsets of a full news document can improve the performance even more. Another interesting direction is to automatically decide the optimal subsets of texts (with respect to 5W1H) as the input for news headline generation.

Acknowledgement

We would like to express our gratitude to the anonymous reviewer for helpful comments. This work was conducted when the first author was a research intern at Yahoo Japan Corporation.

References

1. Hovy, E., Lin, C.Y.: Automated text summarization and the summarist system. In: Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998.

¹² It implies we discarded more documents than the original script.

- TIPSTER '98, Stroudsburg, PA, USA, Association for Computational Linguistics (1998) 197–214
2. Zhou, L., Hovy, E.: Template-filtered headline summarization. In: In the Proceedings of the ACL workshop, Text Summarization Branches Out. (2004) 56–60
 3. Banko, M., Mittal, V.O., Witbrock, M.J.: Headline generation based on statistical translation. In: Proceedings of the ACL, Stroudsburg, PA, USA, Association for Computational Linguistics (2000) 318–325
 4. Zajic, D., Dorr, B.J., Schwartz, R.: Bbn/umd at duc-2004: Topiary. In: Proceedings of the NAACL Workshop on Document Understanding. (2004) 112–119
 5. Alfonseca, E., Pighin, D., Garrido, G.: Heady: News headline abstraction through event pattern clustering. In: Proceedings of the ACL (Volume 1: Long Papers), Sofia, Bulgaria, Association for Computational Linguistics (August 2013) 1243–1253
 6. Kong, S.y., Wang, C.c., Kuo, K.c., Lee, L.s.: Automatic title generation for chinese spoken documents with a delicate scored viterbi algorithm. In: Proceedings of Spoken Language Technology (SLT) Workshop. (2008) 165–168
 7. Tzouridis, E., Nasir, J., Brefeld, U.: Learning to summarise related sentences. In: Proceedings of COLING: Technical Papers, Dublin, Ireland, Dublin City University and Association for Computational Linguistics (August 2014) 1636–1647
 8. Wang, R., Dunnion, J., Carthy, J.: Machine learning approach to augmenting news headline generation. In: Proceedings of the IJCNLP, Jeju Island, Korea, Association for Computational Linguistics (October 2005) 155–160
 9. Zajic, D., Dorr, B.: Automatic headline generation for newspaper stories. In: Proceedings of the Workshop on Automatic Summarization. (2002) 78–85
 10. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the EMNLP, Lisbon, Portugal, Association for Computational Linguistics (September 2015) 379–389
 11. Chopra, S., Auli, M., Rush, A.M.: Abstractive sentence summarization with attentive recurrent neural networks. In: Proceedings of the NAACL-HLT, San Diego, California, Association for Computational Linguistics (June 2016) 93–98
 12. Kikuchi, Y., Neubig, G., Sasano, R., Takamura, H., Okumura, M.: Controlling output length in neural encoder-decoders. In: Proceedings of the EMNLP, Austin, Texas, Association for Computational Linguistics (November 2016) 1328–1338
 13. Nallapati, R., Zhou, B., dos Santos, C.N., Çağlar Gülçehre and Bing Xiang: Abstractive text summarization using sequence-to-sequence rnns and beyond. In: CoNLL. (2016)
 14. Takase, S., Suzuki, J., Okazaki, N., Hirao, T., Nagata, M.: Neural headline generation on abstract meaning representation. In: Proceedings of the EMNLP, Austin, Texas, Association for Computational Linguistics (November 2016) 1054–1059
 15. Ayana, Shen, S.Q., Lin, Y.K., Tu, C.C., Zhao, Y., Liu, Z.Y., Sun, M.S.: Recent advances on neural headline generation. *Journal of Computer Science and Technology* **32**(4) (Jul 2017) 768–784
 16. Tan, J., Wan, X., Xiao, J.: From neural sentence summarization to headline generation: A coarse-to-fine approach. In: Proceedings of the IJCAI. (2017) 4109–4115
 17. Leskovec, J., Milic-Frayling, N., Grobelnik, M.: Extracting summary sentences based on the document semantic graph. Microsoft Research (2005)
 18. Genest, P.E., Lapalme, G.: Framework for abstractive summarization using text-to-text generation. In: Proceedings of the Workshop on Monolingual Text-To-Text Generation, Portland, Oregon, Association for Computational Linguistics (June 2011) 64–73

19. Wang, W.: Chinese news event 5w1h semantic elements extraction for event ontology population. In: Proceedings of the 21st International Conference on World Wide Web. WWW '12 Companion, New York, NY, USA, ACM (2012) 197–202
20. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the EMNLP, Doha, Qatar, Association for Computational Linguistics (October 2014) 1724–1734
21. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of the NIPS, Cambridge, MA, USA, MIT Press (2014) 3104–3112
22. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8) (November 1997) 1735–1780
23. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of the ICLR. (2015)
24. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the EMNLP, Lisbon, Portugal, Association for Computational Linguistics (September 2015) 1412–1421
25. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, Association for Computational Linguistics (October 2014) 103–111
26. Kiddon, C., Zettlemoyer, L., Choi, Y.: Globally coherent text generation with neural checklist models. In: Proceedings of the EMNLP, Austin, Texas, Association for Computational Linguistics (November 2016) 329–339
27. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Proceedings of the ACL (Volume 1: Long Papers), Vancouver, Canada, Association for Computational Linguistics (July 2017) 1073–1083
28. Cheng, J., Lapata, M.: Neural summarization by extracting sentences and words. *CoRR abs/1603.07252* (2016)
29. Tan, J., Wan, X., Xiao, J.: Abstractive document summarization with a graph-based attentional neural model. In: Proceedings of the ACL (Volume 1: Long Papers), Vancouver, Canada, Association for Computational Linguistics (July 2017) 1171–1181
30. Napoles, C., Gormley, M., Van Durme, B.: Annotated gigaword. In: Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction. AKBC-WEKEX '12, Stroudsburg, PA, USA, Association for Computational Linguistics (2012) 95–100
31. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Proceedings of the ACL workshop on Text Summarization Branches Out. (2004) 10
32. Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.: Opennmt: Open-source toolkit for neural machine translation. In: Proceedings of ACL, System Demonstrations, Vancouver, Canada, Association for Computational Linguistics (July 2017) 67–72