# News Headline Generation via Neural Hierarchical Abstractive Summarization

Younghun Song[1], Byungsoo Kim[2], Juneyong Yang[2], and Taewon Yoon[2]

[1]Graduate School of Knowledge Service Engineering , KAIST
younghun.song@kaist.ac.kr

[2] School of Computing , KAIST
{bskim90,laoconeth,dbsus13}@kaist.ac.kr

## Abstract

The task of abstractive summarization remains as one of the most challenging problems in NLP. Recent upsurge in deep learning ignited the adoption of neural models into abstractive summarization tasks, but neural approaches to document-to-sentence abstractive summarization have received little attention from the NLP community, compared to sentence-to-sentence abstractive summarization. In this project report, we propose a novel neural hierarchical abstractive summarization model that first carries out an extractive summarization and then use the intermediate results as inputs to the abstractive summarization layer. Although we have failed to match the state-of-the-art abstractive summarization algorithms, we believe that the model will perform much better if we mend the issues found during our project development.

## 1 Problem Definition

Abstractive summarization is the task of generating a shorter version of a given corpus while attempting to preserve its meaning. In NLP, a summary refers to a compressed paraphrasing of a corpus. Previous works in abstractive summarization [8, 11, 4, 12] have mainly focused on a problem of summarizing only one sentence. In this setting, a summary is just a shorter sentence that preserves the meaning of the longer original sentence. In contrast, our team's goal is to generate a headline given a news article. This problem is far more challenging than the previous abstractive summarization tasks, since it requires compact compression of long news articles, which usually consist of more than 10 sentences. To tackle this challenging task, we first formally define our problem as the follows: generating a single (summary) sentence that represents the overall context of a long corpus.

## 2 Related Works

Since abstractive summarization is similar to sequence-to-sequence labeling tasks, recent approaches on abstractive summarization relied on either an RNN encoder-decoder model with attention mechanism [8, 11, 4], or an abstractive meaning representation tree that supports rich feature representations [12].

Another popular line of research on summarization tasks is extractive summarization, which tries to choose the right sentences from a corpus that well summarize the corpus. In general, extractive approaches are computationally more efficient than abstractive approaches. Traditional approaches [13] were heavily handcrafted: handcrafted features were extracted from a corpus and were fed into off-the-shelf classifiers such as SVM or Naive Bayes. However, similar to abstractive approaches, recent works on extractive summarization were also largely based on the RNN encoder-decoder

model with attention mechanism [3, 5]. Among them, papers like [5] take an advantage of auxiliary classification tasks in the multi-task learning framework to boost the performance, which was one of the main inspirations of our model architecture.

In this work, we propose a new hierarchical abstractive summarization model that can get the best of both worlds. Our model consists of two components: an extractive summarizer and an abstractive summarizer. The extractive summarizer chooses candidate sentences from a news article. The candidate sentences are then fed to the abstractive summarizer, which generates a headline that well represents the article. In the experiment, we show that our hierarchical abstractive summarization model consistently generates better headlines than baseline models in the sense of ROUGE-1 f1 score.

## 3 Model Description

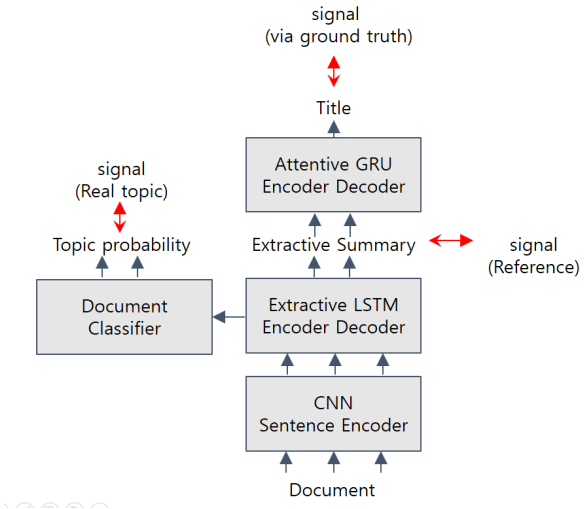


Figure 1: Hierarchical structure of the model. The main aspect of our model is that it generates the headline from the extractive summary that our extractive summarizer generates, which makes our model to have hierarchical structure.

Our model consists of one main task - headline abstractive summarization - and two related sub-tasks - extractive summarization and document classification. We used this approach since we assumed that the features learned from extractive summarization and document classification will help extract only the important sentences from the document, which should also be useful for the abstractive summarization. The overview of our model architecture is illustrated in Figure 1. The extractive summarizer first produces an extractive summary, and this extractive summary is then used in the following abstractive summarization task(headline generation). This is why we call our model "Hierarchical". Each models learn from multi-signals: the signals from their own results and the signals from the upper/lower part of the model. Documents are represented as sequences of word embeddings.

### 3.1 Word Embeddings

We have initialized the word embeddings using the common crawl 840B tokens version[1] of GloVe[10]. The embeddings were fined-tuned as we trained our model. Special characters and too common/rare words were removed before the training. This word embedding representation of the documents is used as inputs to both the CNN sentence encoder and the attentive GRU encoder-decoder.

[1] https://nlp.stanford.edu/projects/glove/

For each documents or sentence inputs, sentences are converted into $N \times D$(number of words in sentence * word embedding dimension) matrices. For example, the sentence "I love machine learning" is represented as

$$sentence = \begin{bmatrix} 1 & 4 & -2 & 0 \\ 3 & 1 & 3 & -2 \\ 2 & -1 & 2 & 1 \end{bmatrix}$$

when the embeddings for each words are $I = (1, 3, 2), love = (4, 1, -1), machine = (-2, 3, 2), learning = (0, -2, 1)$.

### 3.2 Sentence Encoder

Since the number of words in each sentences are different, it is desirable if we can somehow fit the sentences into a fixed-length representation and feed the fixed-length vectors as inputs to the extractive LSTM encoder-decoder. That is precisely what the CNN sentence encoder in our model does. The overall structure follows Kim [6], but let us briefly explain how our encoder works.

Let $x_i \in \mathbb{R}^{|V|}$ be the $i$-th word embedding in a sentence consisting of $N$ words, and set $q =$ size of the sliding window. Then, concatenate each words in the sentence as $x_{i:i+q-1} = [x_i : x_{i+1} : \cdots : x_{i+q-1}] \in \mathbb{R}^{|V| \times q}$ and pass the concatenated vectors through the convolution filters so that we obtain the following *feature* for the $i$-th position of the sentence:

$$s_w^i = f(w \cdot x_{i:i+q} + b)$$

where $w$ is the weight corresponding to a particular filter. This feature is calculated for all possible positions in the sentence, and we finally obtain the single sentence feature by max pooling as

$$s_w = \max_i s_w^i$$

This sentence feature is obtained over multiple features and multiple sliding windows to get the finalized fixed-length sentence embeddings, which are then used as inputs to both sentence extraction and document classification.

### 3.3 Extractive Summarizer and Document Classifier

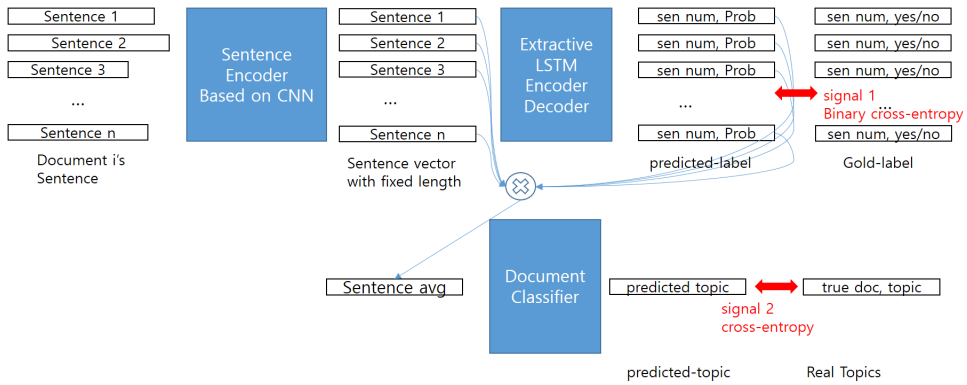


Figure 2: Extractive summarizer and document classifier structure. Both extractive summarization and document classification uses the output from the CNN sentence encoder, which generates fixed-length sentence vectors for each variable-length sentences.

Our architecture for the extractive summarizer and document classifier basically follows that of Isonuma et al. [5]. For extractive summarization, we used Long Short Term Memory(LSTM) Encoder-Decoder structure. The extractive LSTM encoder-decoder receives the fixed-length sentence vectors as inputs from the CNN sentence encoder, and for each sentences, predicts the probabilities for being present in the extractive summary.

Formally, let $s_t$ be the $t$-th sentence input. Then, $p_t$ is the probability of $t$-th sentence to be included in the extractive summary, calculated by the extractive LSTM layer as the follows:

$$
\begin{aligned}
p_t &= \sigma(w_y \cdot [h_t : \bar{h}_t] + b_y) \\
h_t &= LSTM(s_t, h_{t-1}) = t\text{-th hidden state of the LSTM encoder} \\
\bar{h}_t &= LSTM(p_{t-1} \cdot s_{t-1}, \bar{h}_{t-1}) = t\text{-th hidden state of the LSTM decoder} \\
w_y, b_y &= \text{weights \& bias for the LSTM layer}
\end{aligned}
$$

For document classification task, golden labels for document classes are necessary. However, our dataset didn't have any document class labels, so we have utilized Latent Dirichlet Allocation(LDA) [2] to generate the latent topics as the golden class labels for each news articles. Then, we feed the weighted average of the sentences $s_{avg}$ in every document as the input for the document classifier, where $s_{avg} = \frac{\sum_t p_t \cdot s_t}{\sum_t p_t}$. The document classifier receives this input and outputs the probability of the document being in a specific class as the follows:

$$
\begin{aligned}
q_c &= \sigma(w_c \cdot s_{avg} + b_c) \\
c &: \text{class index}
\end{aligned}
$$

### 3.4 Abstractive Summarizer

For abstractive summarization, we used an attentive Gated Recurrent Unit(GRU) encoder-decoder architecture. An input to the attentive GRU encoder-decoder is a sequence of words from the *summarized* document that the extractive summarizer produced. More specifically, an input is built as follows:

1. The Extractive summarizer chooses the sentences from a document to extract: $s_1, s_3, s_{10}, \ldots, s_{|D|}$
2. All of the chosen sentences are concatenated to produce a long sequence of words as a document summary: $doc_{summary} = [s_1 : s_3 : s_{10} : \cdots : s_{|D|}]$
3. $doc_{summary}$ is used as an input to the attentive GRU encoder-decoder.

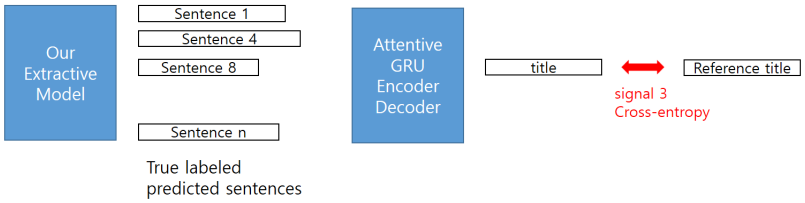


Figure 3: Abstractive summarization model, attentive GRU Encoder Decoder is used

Our attentive GRU encoder-decoder follows the same structure as Bahdanau et al. [1]. Given an i-th input sequence $doc^{(i)}_{summary} = [x_1, x_2, \ldots, x_{M-1}, x_M]$, our abstractive summarizer's objective is to predict target word sequences $headline^{(i)} = [y_1, y_2, \ldots, y_N]$ as the headline of the input document, where $N < M$ since any summarization(or headline) must be shorter than the original document. Hence, the single-document training objective of the abstractive summarizer is to maximize the following joint probability with respect to its parameter set $\theta$:

$$
p(\hat{y_1}, \hat{y_2}, \ldots, \hat{y_N}|\theta) = \prod_{t=1}^{N} p(\hat{y_t}|\hat{y_1}, \hat{y_2}, \ldots, \hat{y_{t-1}}, \mathbf{x}, \theta) = \prod_{t=1}^{N} p(\hat{y_t}|\hat{\mathbf{y}_{t-1}}, \mathbf{x}, \theta)
$$

$\hat{y_t}$ = predicted word at time $t$

$p(\hat{y_t}|\hat{\mathbf{y}_{t-1}}, \mathbf{x}, \theta)$ = pdf over the vocabulary calculated from the attentive GRU decoder

Extending this objective to the whole corpus, the final training objective is defined as the follows:

$$\prod_{i=1}^{N(docs)} \prod_{t=1}^{N_{(i)}} p(\hat{y_t}^{(i)} | \hat{\mathbf{y}_{t-1}}^{(i)}, \mathbf{x}^{(i)}, \theta)$$

# 4 Experimental Setup

## 4.1 Dataset

The original dataset we planned to use was the scisumm-corpus[2]. The corpus was released by WING-NUS - a Web IR / NLP Group from the National University of Singapore - to facilitate the research on summarization algorithms.

However, we found out that most of the scisumm-corpus was in fact broken, so we switched to an alternative source: the Kaggle News Summary Dataset.[3] The dataset includes the author names, headlines, summaries, and the complete contents of the news articles from Hindustan, Indian times and Guardian, written from Feb. 2017 to Aug. 2017. See Table 1, 2 for the detailed statistics about the dataset.

Table 1: Dataset splits

| # Documents | |
|---|---|
| Train | 3000 |
| Test | 1396 |
| Dropped(N/A) | 119 |
| Total | 4515 |

Table 2: Corpus Statistics

| Statistics | |
|---|---|
| vocabulary size | 51732 |
| sentences per document(min) | 1 |
| sentences per document(average) | 14 |
| sentences per document(max) | 696 |

## 4.2 Evaluation and Baselines

We report the ROUGE-1 F1 score of our model, which is widely used for measuring the quality of an abstractive summarization algorithm. We used two baselines for comparison: the first baseline naively picks the first sentence of an article as a title(FIRST), and the other baseline is a Deep Attentive LSTM encoder-decoder(DALSTM)[8] that uses 3-stage LSTM with attention.

## 4.3 Implementation Details

### Environment

We used PyTorch[4] for implementation. Our final model took about 30 minutes to train on a Intel(R) Core(TM) i7-7700 CPU and Nvidia GTX 1080 Ti GPU.

### Hyperparameters and Training Details

Table 3 includes the summary of the hyperparameters and training details for our experiment, some of which were already stated in Section 3. Word embedding was initialized using the GloVe pretrained vectors and was fine-tuned during the training process. The architecture of the CNN Sentence Encoder follows Kim [6]. Different RNN architectures were used for the extractive and abstractive summarizer: LSTM for the extractive summarizer and GRU for the abstractive summarizer. The structure of the extractive summarizer is the same as in Isonuma et al. [5].

To optimize over both the extractive and abstractive objectives, we used Stochastic Gradient Descent with learning rate = 0.005 to *alternately* optimize each objectives. We also tried the Adam[7] optimizer, but the training loss exploded so we did not use it for our experiment.

[2]https://github.com/WING-NUS/scisumm-corpus

[3]https://www.kaggle.com/sunnysai12345/news-summary

[4]http://pytorch.org/

Table 3: Hyperparameters and training details

| | | |
|---|---|---|
| **Word Embedding** | vocabulary size | 30000 |
| | dimension | 200 |
| | pretrained | GloVe |
| | fine-tuning | **yes** |
| **CNN Sentence Encoder** | kernel sizes | [1,2,3,4,5] |
| | number of kernels | 100 |
| | sentence embedding dim | 250 |
| **Extractive LSTM Encoder-Decoder** | number of layers | 1 |
| | hidden size | 400 |
| | # document classes | 6 |
| **Abstractive Attentive GRU Encoder-Decoder** | number of layers | 1 |
| | hidden size | 400 |
| | attention type | Bahdanau et al. [1] |

# 5 Results

As mentioned in section 4.2, we report the ROUGE-1 F1 score of 2 baselines - FIRST, DALSTM - and our model in Table 4. We observed that our model outperforms both baselines by a large margin. However, the quality of the generated summaries are actually somewhat questionable. Table 5 shows the 2 sample summaries generated by our model alongside their reference summaries. We can see that the generated summary does capture a portion of meaningful words in the reference summary, but is incomplete in terms of the semantics and grammatical structure.

Table 4: Abstract Summarization Performance Comparison

| ROUGE-1 f1 Score | |
|---|---|
| FIRST | 0.018 |
| DALSTM | 0.051 |
| Our Model | 0.137 |

Table 5: Abstract Summarization: Generated Summary vs. Reference Summary

| Examples of generated summaries(titles) |
|---|
| (reference): depression needs expression UNK not suppression UNK pm modi<br>(generated): india UNK s pm modi |
| (reference): delhi metro to get UNK driverless UNK trains in june<br>(generated): delhi metro in delhi |

# 6 Discussions

## 6.1 Encoder-Decoder Models and the Difficulty of the Document Abstractive Summarization Task

Although our model did succeed in outperforming the naive baselines, the reported ROUGE-1 f1 score is substantially lower than the cutting-edge models such as Rush et al. [11]. Despite of the fact that our dataset is small and of bad quality, it seems that our model's performance is lackluster in terms of absolute ROUGE-1 score.

However, it is unfair to directly compare the scores since the task itself is quite different in its scale(these models focus on abstractive summarization of short documents, typically of 3 4 sentences). In our experiments we have performed more aligned comparisons by conducting baseline experiments on our dataset - with FIRST and DALSTM. It is shown that the DALSTM model underperforms significantly regarding this task. Although the RNN encoder-decoder model performs very well on

language generation tasks such as machine translation[1], its performance deteriorates as the sequence length grows, thus being increasingly unsuitable for the task as the document length increases. Unlike the DALSTM baseline, our model has been proven to be better suited for the task. This is mostly credited to the hierarchical structure of the model, in which the extractive summarization module drastically compresses documents, preserving only the gist of them. Moreover, our summarization model operates on sentence embeddings rather than word embeddings during the extraction stage, resulting in a shorter encoding-decoding process.

However, we suspect that this sentence embedding is preventing the model from capturing *moderately* important words from the document. Signals from the word embeddings of the moderately important words are squashed when a sentence is encoded into a sentence vector as a whole, so the model might have difficulties in retrieving these words effectively. In an ideal situation where we have a fictitious powerful summarizer, the model would successfully generate appropriate words given the contexts(sentence embeddings) only. However, from the results of our experiment, it is obvious that this situation is unrealistic.

### 6.2 Regarding the Issues in the Dataset

During the experiment phase, we have encountered numerous issues regarding the dataset. In order to train and test our model, document-summary(extractive)-title triples plus topic labels were required. Due to this constraint, we were unable to utilize common datasets of large size. After the failure of scisumm-corpus, we were forced to settle with our current dataset since finding a sound dataset composed of stated triplets were very difficult. As a consequence this non-ideality of our chosen dataset is suspect to hinder the observed performance of our model. More fine-grained analysis of such drawbacks are elaborated below.

#### 6.2.1 Topic Label Generation

The dataset did not contain the topic labels necessary to train our model. To overcome this deficiency, we used LDA to generate topic labels, as introduced in section 3.3. Although LDA is a reliable and effective unsupervised topic modeling method, the produced topic labels are not human-annotated, thus the labels are not so reliable. This might have degraded the results in some cases.

#### 6.2.2 Out of vocabulary words in GloVe

During the preprocessing stage, we have noticed that many words in the dataset are not present in the GloVe dictionary. Although GloVe dictionary contains 400,000 vocabularies, numerous Indian words transcribed to English were not present, causing them to be initialized randomly or mapped to UNK tokens. So for those words that are meaningful among the Indian communities - especially proper nouns - were lost during this process. For example, from the Table 5, we can see that internationally well-known proper nouns such as "modi"(the Indian prime minister's name", "delhi"(the capital of India) are contained in the corpus and thus appear in the summary when they are needed. However, words such as "rakshabandhan", which is one of the major festivals of india, are absent from the GloVe dictionary. This information loss is crucial, considering the fact that proper nouns convey a significant proportion of contextual information in a summary, especially because of the fact that the dataset consists of news articles.

#### 6.2.3 Skewedness of words in our dataset dictionary

Unlike the Gigaword corpus[9], our dataset is much skewed in the sense that they are solely composed of news articles of Indian origin.

#### 6.2.4 Maintaining stopword lists

Due to the time constraint, stopword list maintenance via human inspection of the dataset was not done thoroughly. Since our dataset is comprised of news articles, certain portion of irrelevant information is suspected to be included in the document, such as Instagram links, irrelevant hashtags and so on. These words which have penetrated the stopword list would have obscured the model during the training phase.

## 7 Future Works

The optimal choice of work to improve our model would be to acquire a perfect document-(extractive)summary-title-topic quadruples, yet more generalized than the current dataset. However this task is orthogonal to the model development and improvement tasks.

Architectural improvements might also be possible. We can reason that the sentences extracted by the extractive summarizer are not of equal importance. We can suppose that the output probabilities from the extractive summarizer convey this information. However, in the current scheme, this relative importance is simply abandoned after we extract the importance sentences to form the extractive summary. For future works, to preserve the relative importance of each sentences, we can weigh the embeddings of each words in the sentences included in the summarized document by their extractive probabilities before we feed them into the abstractive summarizer. Receiving this weighted input, the encoder-decoder model will have external prior knowledge about the relative importance of each words *within* the summarized document, which will contribute to the improvement in the summary(headline) quality.

## 8 Conclusion

We have proposed an abstractive summarization model with a hierarchical architecture. The model utilizes its hierarchical structure via joint training of various tasks using multiple loss signals generated from extractive summarization task, abstractive summarization task and topic prediction task. The 2-staged summarization process divides the load between the 2 summarization stages, which allows the neural abstract summarizer to effectively handle long word streams such as documents. The model has shown significantly better performance compared to the DALSTM model and FIRST heuristic model evaluated on ROUGE-1 F1 score. Despite the difference in datasets, the performance is quite below the numbers reported by other state-of-the-art models. Future works were proposed to tackle the supposed weaknesses in our model.

## References


[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations(ICLR)*, 2015.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] J. Cheng and M. Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[4] S. Chopra, M. Auli, and A. M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[5] M. Isonuma, T. Fujino, J. Mori, Y. Matsuo, and I. Sakata. Extractive summarization using multi-task learning with document classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[6] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations(ICLR)*, 2015.

[8] K. Lopyrev. Generating news headlines with recurrent neural networks. In *arXiv:1512.01712 [cs.CL]*, 2015.

[9] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda. English gigaword. *Linguistic Data Consortium*, 2011.

[10] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[11] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[12] S. Takase, J. Suzuki, N. Okazaki, T. Hirao, and M. Nagata. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[13] K.-F. Wong, M. Wu, and W. Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008.