EXTRACTIVE TEXT SUMMARIZATION USING RECURRENT NEURAL NETWORKS WITH ATTENTION MECHANISM

Shimirwa Aline Valerie and Jian Xu

School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

ABSTRACT

Extractive summarization aims to select the most important sentences or words from a document to generate a summary. Traditional summarization approaches have relied extensively on features manually designed by humans. In this paper, based on the recurrent neural network equipped with the attention mechanism, we propose a data-driven technique. We set up a general framework that consists of a hierarchical sentence encoder and an attention-based sentence extractor. The framework allows us to establish various extractive summarization models and explore them. Comprehensive experiments are conducted on two benchmark datasets, and experimental results show that training extractive models based on Reward Augmented Maximum Likelihood (RAML)can improve the model's generalization capability. And we realize that complicated components of the state-of-the-art extractive models do not attain good performance over simpler ones. We hope that our work can give more hints for future research on extractive text summarization.

KEYWORDS

Extractive summarization, Recurrent neural networks, Attention mechanism, Maximum Likelihood Estimation, Reward Augmented Maximum Likelihood.

1. INTRODUCTION

Automatic text summarization is one of the challenging and interesting tasks of natural language processing, which can help people to obtain important and relevant information from a large number of documents in a short period. It has gained its popularity due to the importance it has in different information access applications such as search engines, information retrieval, recommendation systems, question answering, etc. When it comes to automatic text summarization, there are two approaches extractive text summarization and abstractive text summarization. While in the extractive summarization approaches the most salient sentences or words from the document are selected and concatenated to form a summary, in the abstractive summarization approaches the sentences in the document are paraphrased to make the summary. Even though the abstractive summarization approaches have made steps in recent years, the extractive approaches are still attractive since they can generate coherent and grammatically correct summaries and are computationally efficient [1]. Thus, in this work, we focus on extractive summarization.

A main requirement for the extractive summarization approach is to have a good method to determine the important contents that represent the important information in the document [2].

David C. Wyld et al. (Eds): CMLA, CIoT, CSEIT, NCS, NLPD, SPM - 2021
pp. 233-248, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111518

Several traditional techniques have been used to extract the important sentences to be included in the summary.

These techniques can be categorized into greed-based [3], graph-based [4], hidden Markov models[5], and constraint optimization[6],etc. These traditional extractive techniques use human-crafted features and are complicated. Moreover, they mostly fail to build a good representation of the document. This leads them to fail to generate good summaries.

In recent years, deep learning-based models have been used for extractive text summarization. These models can learn from the input text data directly, and they have attained state-of-the-art results. To create representation of the sentences and documents, neural network-based extractive models are basically constructed using recurrent neural networks [1,7], convolutional neural networks [8,9], the combination of convolutional and recurrent neural networks [10,11] or transformers[12]. While there has been great effort dedicated to designing neural network-based extractive summarization models, there is still a need to explore what makes them work well and how they can be improved. Therefore, in this paper, we present a recurrent neural network-based extractive model that consists of a hierarchical sentence encoder and an attention-based sequence-to-sequence sentence extractor. And we closely explore how the choice of sentence encoder can influence the model's performance.

Since there is little work that has been done on learning approaches for neural extractive summarization, we also examine how different learning approaches can contribute to the performance and generalization of the model. Existing neural-based extractive summarization systems fail to generalize better on the data they have not seen. We introduce the use of the RAML approach to the summarization task with the expectation that it can improve the generalization ability of the model.

The main contributions of this work are: (1) we adopt the RAML optimization approach to the task of extractive summarization; (2) we present two hierarchical neural structures (Avg-Seq_to_Seq and Rnn-Seq_to_Seq) for the extractive summarization task; (3) we perform a multi-domain test, which allows us to better understand how biases in different datasets influence the performance of our models;(4) we analyze the generalization capability of the models on out-of-domain datasets. For example, we train a model on the CNN dataset and test it on the PubMed dataset to see how the model can generalize to other datasets. Additionally, we demonstrate the effect of the position of the sentences on the performance of our models.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 demonstrates our model. Section 4 describes our experiments and results. Section 5 demonstrates our discussion. Section 6 concludes our work.

2. RELATED WORK

To identify and select the most important sentences in a document or set of documents to make a summary, researchers have used several methods. These methods can be classified into statistical, graph, machine learning, deep learning-based approaches, etc. In this section, we demonstrate some of these approaches.

2.1. Statistical-Based Summarization Approaches

These approaches mainly use statistical features such as term frequency, sentence position, sentence length, TF-IDF (Term Frequency-Inverse Document Frequency), sentence to centroid

similarity, etc., to score the sentences. Then the sentences with high scores are selected to make the summary. Similarity to centroid sentence was used in [17] to score sentences. In their work TF-IDF is used to get centroid sentence then based on the cosine similarity between each sentence and centroid sentence, each sentence is given a score. Eleven features including document frequency, sentence position, normalized sentence length, proper noun, topic frequency, numerical data, headline frequency, start cluster frequency, and skip Bi-gram topic frequency are used in [18] to score sentences and then sentences with high scores are selected until reaching the length limit of the summary. One of the advantages of statistical-based approaches is that they do not require training data or complex linguistic processing. And one of the limitations of them is that they can generate summaries with redundant information because similar sentences with high scores can be included in the summary.

2.2. Graph-Based Summarization Approaches

Researchers have also used graph-based summarization approaches to perform extractive summarization. In the graph-based method, sentences are represented using nodes of a weighted graph. And the similarities between sentences are represented using edges. Sentence similarity values are obtained based on the overlapping phrases or words between sentences, then the sentences which have high similarity with the other ones are selected to generate the final summary. Two well-known graph-based approaches are Lex Rank and Text Rank. Text Rank was introduced by Mihalcea [4] to extract sentences and keywords from a single document. LexRank was introduced by Erkan [19] to compute the importance of the sentence based on the idea of eigenvector centrality in the sentence representation graph. Graph-based approaches generate summaries with less redundant information and they do not require annotated corpora. One of the disadvantages of these methods is that they do not take into account the importance of the words. They treat the weights of the words equally equal.

2.3. Machine Learning-Based Summarization Approaches

Different machine learning methods have been used to carry out extractive text summarization task. Some of those methods are Support Vector Machine(SVM) [20], Naïve Bayesian [21], Hidden Markov Models [5], etc. A binary classifier is proposed in [21] to score sentences using Bayes' rule. In their work, the probability of each sentence to be included in the summary is obtained by using manually crafted features. In [5] hidden Markov model algorithm identifies the likelihood of each sentence to be select for the summary. SVM is used in [20] for query-based summarization to declare appropriate sentences to put in the summary. The advantage of machine learning-based approaches is that they can explore many features and can represent documents in a better way than statistical and graphical approaches but they also need human crafted features to generate summaries with high accuracy and they need large labeled corpora.

2.4. Deep Learning-Based Summarization Approaches

In recent years, deep learning-based approaches have gained popularity over the above-mentioned traditional approaches because they can directly learn from the data. Neural network-based extractive summarization models have achieved state-of-the-art results. For instance, SummaRuNNer [1] uses bidirectional RNNs at the word level to encode sentences and another bidirectional RNNs at the sentence level to predict which sentences are to be extracted. In their work, the sentence extractor generates document representations and calculates distinct scores for novelty, location and salience of the sentences. In [7], authors propose convolutional neural network (CNN)-based model to encode sentences at the word level and design an extractor to predict which sentence should be included in the summary at the sentence level. Authors [13] propose an end-to-end neural extractive summarization model that learns to score and select

sentences jointly. In their work to obtain sentence representations, they use a hierarchical encoder, and then the output summary is obtained by extracting one sentence at a time. Based on the previous works that used hierarchical architectures [7, 13, 15], we also present a recurrent neural network-based model that consists of hierarchical sentence encoder and sequence-to-sequence based sentence extractor.

Although neural extractive summarization models have achieved great performance, most of the existing works, during training of these models use MLE. MLE approach maximizes the likelihood of the ground truth labels and disregards the structure of the output space by taking all the output which do not match the ground truth labels as equally wrong, irrespective of their structural closeness to the ground truth target. This leads to the inconsistency between training and testing objectives (i.e., during training the model learns to maximize the likelihood of the ground truth labels while during testing the objective is to generate the summaries with a high ROUGE score concerning the reference summary). This inconsistency can cause the overfitting of the ground truth labels and leads to poor generalization capability on test datasets. Some researchers have tried to eliminate this inconsistency by optimizing task reward (ROUGE evaluation metric) directly using Reinforcement learning (RL) approaches. For example, authors [10] proposed an approach that optimizes the ROUGE metric globally and use reinforcement learning objective to rank sentences that can be included in the summary. Authors [11] proposed a consistency model that takes syntactic coherence and cross-sentence semantic patterns. They used the RL objective to train their model. In their work, the output of the model and the reward calculated using the ROUGE package are combined to capture the cross-sentence consistent patterns. The limitation of the reinforcement learning approaches is that they suffer from problems of high variance in the gradients and poor sample effectiveness (sampling from a non-stationary model distribution). In this paper, we adopt a learning approach called RAML to the task of extractive summarization with the expectation that it can improve the performance and generalization of our models. RAML approach was proposed by [16] to include task reward into Maximum-likelihood training. It was successfully applied to machine translation task and speech recognition. It combines the straightforwardness and computational effectiveness of MLE with the advantages of maximizing task reward. Unlike MLE that maximizes the log-likelihood of the ground-truth labels, RAML can sample from the exponentiated payoff distribution which permits the estimation of anticipated maximum likelihood. In this paper, we not only train our models based on the MLE approach but also, train them based on the RAML approach.

3. NEURAL NETWORK-BASED EXTRACTIVE SUMMARIZATION MODEL

In this paper, we treat the task of extractive summarization as a sequence labeling problem or a classification problem. Given a document d consists of n sentences $d = \{s_1, s_2, s_3 \ldots\ldots, s_n\}$. we aim to generate a summary by predicting the corresponding labels of sequences $y_1, y_2, y_3 \ldots\ldots, y_n \in \{0, 1\}^n$, where $y_j=1$ denotes that the j^{th} sentence should be included in the summary, otherwise $y_j=0$. Based on the extraction probabilities, sentences are selected until reaching the length limit of the output summary. Since each sentence itself is a sequence of words $s_j = \{w_1, w_2, w_3 \ldots\ldots, w_L\}$, we set word budget $b \in \mathbb{N}$ to put a constraint on the limit length of the output summary $\sum_j^n y_j \ . L \leq b$. Generally, our proposed model is suitable for a single document. It consists of the following components, as shown in Figure 1.

3.1. Embedding Layer

The embedding layer is the first layer in our model. It converts positive integers (indices) of the words in the training dataset into dense vector representations of fixed size. These dense vector representations capture the syntactic and semantic potential meaning of the words. Instead of

training our dense vector representation of the words, we initialize the embedding layer with glove pre-trained word embeddings with 200 dimensions.

3.2. Sentence Encoder

The sentence encoder converts the sequence of word embeddings of each sentence into a fixed-length vector. We get sentence representations using two different approaches the first one is by using a Recurrent neural network (Rnn) and another one is simply by averaging word embeddings (Avg). By using the Rnn approach, at each time step, a Bidirectional Recurrent Neural Network (Bi-RNN) runs at the word level of each sentence and then constructs sentence representation s_j. We employ a Bidirectional Gated Recurrent Unit (Bi-GRU) [24] as RNN cells. Bi-directional GRU consists of forwarding and backward GRU. Forward GRU reads the word embeddings in a sentence from left to write to generate a sequence of hidden states $(\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, \ldots, \overrightarrow{h_L})$. The backward GRU reads word embeddings in the sentence from right to left to form another sequence of hidden states $(\overleftarrow{h_1}, \overleftarrow{h_2}, \overleftarrow{h_3}, \ldots, \overleftarrow{h_L})$.

$$\overrightarrow{h_j} = \overrightarrow{GRU}\left(w_j, \overrightarrow{h_{j-1}}\right) \quad (1)$$

$$\overleftarrow{h_j} = \overleftarrow{GRU}\left(w_j, \overleftarrow{h_{j+1}}\right) \quad (2)$$

Where the initial state of forwarding Bi-GRU is set to zero vector ($\overrightarrow{h_1} = 0$)as well as the initial state of backward Bi-GRU ($\overleftarrow{h_L} = 0$).After reading the words in the sentence, the sentence representation at the word level is constructed by concatenating the hidden states of last forward and backward GRU:

$$s_j = \left[\overrightarrow{h_L}; \overleftarrow{h_1}\right] \quad (3)$$

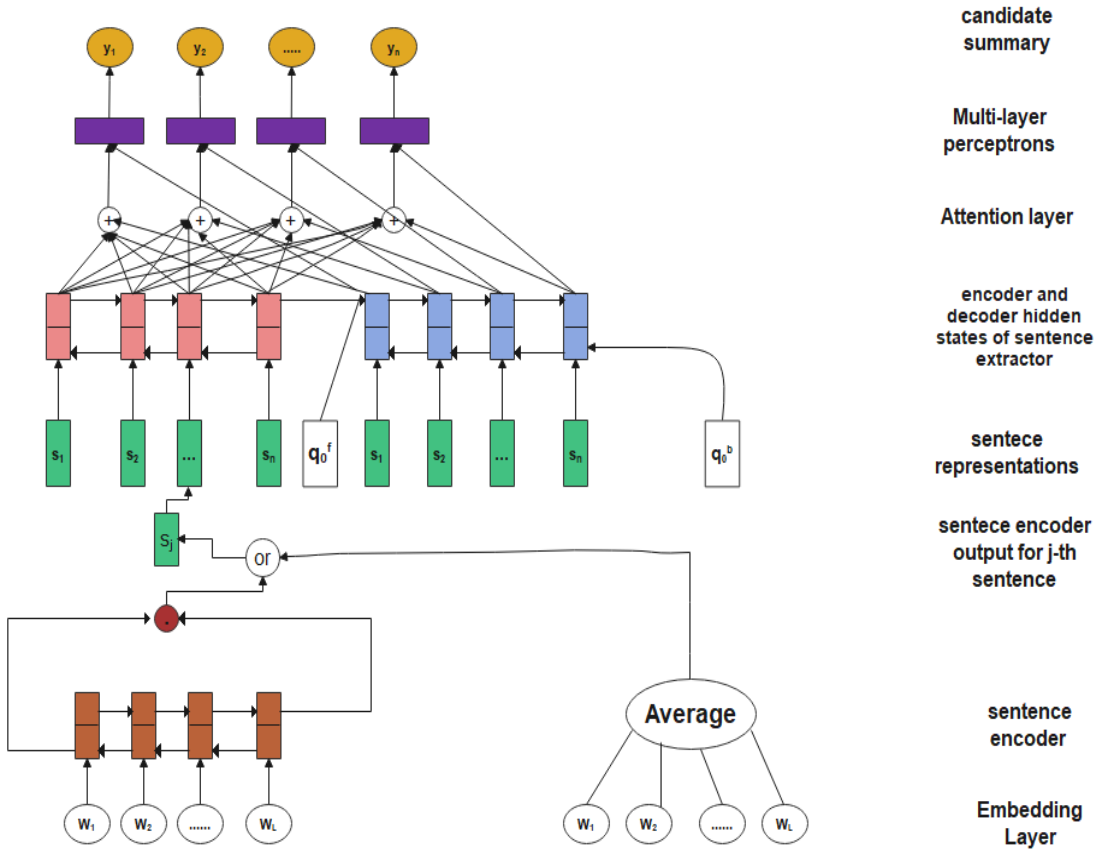

Figure 1. Overview of the two different hierarchical architectures (Rnn- Seq_to_Seq architecture or Avg-Seq_to_Seq architecture) for extractive summarization.

In Figure 1 vertical brown blocks present the sentence encoder's hidden states. ⊙ presents concatenation. Vertical green blocks indicate the output of the sentence encoder. White block (q_0^f) presents forward learned embeddings ("begin decoding") and white block (q_0^b) presents backward learned embeddings. Vertical pink boxes present hidden states of the encoder part of the sentence extractor. Vertical blue boxes indicate hidden states of the decoder part of the sentence extractor. ⊕ presents attention layer. Horizontal purple blocks indicate multi-layer perceptron. Yellow ovals indicate the candidate summary.

By using the averaging approach, each sentence representation s_j is obtained by averaging its word embeddings:

$$s_j = \frac{1}{L}\sum_{j=1}^{L} w_j \qquad (4)$$

3.3. Sentence Extractor

Sentence extractors take in sentence hidden vectors $s_{1:n}$ and generate the sequence of labels $y_{1:n}$. We use the attention-based sequence-to-sequence sentence extractor (Seq_to_Seq) which consists of an encoder, a decoder, an attention layer, and multi-layer perceptrons.

3.3.1. Encoder Part of Sentence Extractor

The encoder part of the sentence extractor takes sentence representations *(s_1, s_2, s_3, ..., s_n)* from the sentence encoder as inputs and encodes them using Bi-GRU. Forward and backward hidden states of Bi-GRU are concatenated to produce a sequence of contextualized sentence representations (sentence embedding s'_j).

$$\overrightarrow{s_j} = \overrightarrow{GRU_{enc}}(s_j, \overrightarrow{s_{J-1}}) \quad (5)$$

$$\overleftarrow{s_j} = \overleftarrow{GRU_{enc}}(s_j, \overleftarrow{s_{J+1}}) \quad (6)$$

We set the initial state of forward Bi-GRU to zero vectors ($\overrightarrow{s_1} = 0$). As well as the initial state of backward Bi-GRU($\overleftarrow{s_n}$=0). Sentence representation hidden vectors at sentence level:

$$s'_j = [\overrightarrow{s_j}; \overleftarrow{s_j}] \quad (7)$$

3.3.2. Decoder Part of Sentence Extractor

The decoder part of the sentence extractor takes in the sentences from the sentence encoder as inputs and then transforms them into a query vector that sees to the output of the encoder part of the sentence extractor.

$$\overrightarrow{q_j} = \overrightarrow{GRU_{dec}}(s_j, \overrightarrow{q_{J-1}}) \quad (8)$$

$$\overleftarrow{q_j} = \overleftarrow{GRU_{dec}}(s_j, \overleftarrow{q_{J+1}}) \quad (9)$$

$$q_j = [\overrightarrow{q_j}; \overleftarrow{q_j}] \quad (10)$$

The final outputs of the forward and backward encoder are fed to the first decoder steps. q_0^f, q_0^b are learned vectors of the first step of the decoder (i.e., start decoding).

3.3.3. Attention Layer

The attention mechanism is commonly used in abstractive summarization [25] and neural machine translation [26]. It plays a role in enabling models to concentrate on important information of the input while predicting the next output. In the attention layer of our model, given a query vector representation q and a sequence of sentence embeddings [s'_1, s'_2, s'_3, …, s'_n], the attention mechanism computes an alignment score between q and each sentence s'_j. The scores are transformed into probabilities by using a SoftMax function. These probabilities are used as weights to sum all sentences and create a contextual embedding for q.

$$\alpha_{j,i} = \frac{exp(q_j \,.\, s_i)}{\sum_i^n exp(q_j \,.\, s_i)} \quad (11)$$

$$s''_j = \sum_{i=1}^{n} (\alpha_{j,i}\, s_i) \quad (12)$$

3.3.4. Multi-Layer Percetrons

Multi-layer perceptrons (MLP) take in a concatenation of the attention-weighted encoder output and decoder output as input to compute the probability of extracting each sentence.

$$a_j = Relu\big(U \cdot [s_j''; q_j] + u\big) \quad (13)$$

$$p\big(y_j = 1 | s_j\big) = \sigma\big(V \cdot a_j + v\big) \quad (14)$$

where U and V are learned weights, u and v are learned bias.

3.4. Model Training

We first train our model by maximizing the likelihood of the ground truth labels. To achieve this objective, the cross-entropy loss is minimized as follows:

$$\mathcal{L}_{MLE}(\theta) = -\sum_{d=1}^{D}\sum_{j}^{n} log\, p\left(y_j^{(d)} \middle| s_j^{(d)}; \theta\right) \quad (15)$$

Where D represents the total number of documents in the training dataset. $s^{(d)}$ represents the contextualized sentence vectors, n symbolizes the total number of sentences in the document. $y^{(d)}$ is each document's label vector. θ represents model parameters. When minimizing the above objective, the conditional probability of the output targets is escalated, and at the same time, the conditional probability of alternative wrong outputs is decreased. This can lead to overfitting on target outputs and decreases the generalization capability. To solve this issue, we train our model based on the RAML approach which was proposed in [16]. RAML simply attaches a step of sampling on top of the ordinary maximum likelihood estimation objective. And it can sample from an output distribution called exponentiated payoff distribution which serves as a central to linking between MLE and RL objectives, and is defined as follows:

$$q(y|y';\tau) = \frac{1}{Z(y';\tau)} exp\left\{\frac{r(y,y')}{\tau}\right\} \quad (16)$$

where$Z(\hat{y};\tau) = \sum_{y\in Y} \exp\left\{\frac{r(y,y')}{\tau}\right\}$, hyper-parameter τthat controls the smoothness of the best distribution around correct labels.

The RAML objective is defined as follows:

$$\mathcal{L}_{RAML}(\theta;\tau) = \sum_{d=1}^{D}\{-\sum_{y\epsilon Y} q(y|y';\tau)\, log\, p(y|s;\theta)\} \quad (17)$$

As stated by [16] RAML approach can be treated as a hybrid between MLE and RL. The connection can be seen by rewriting $\mathcal{L}_{MLE}, \mathcal{L}_{RL,}\, and\, \mathcal{L}_{RAML}$ using Kullback-Leibler Divergence:

$$\mathcal{L}_{MLE}(\theta) = \sum_{y\epsilon Y} D_{KL}(\delta(y,y')||p(y|s;\theta)) \quad (18)$$

$$1/\tau\,.\,\mathcal{L}_{RL}(\theta;\tau) + constant = \sum_{y\in Y} D_{KL}\,(p(y|s;\theta)||q(y|s;\tau)) \quad (19)$$

$$\mathcal{L}_{RAML}(\theta;\tau) + \text{constant} = \sum_{y\in Y} D_{KL}\left(q(y|s;\tau)||p(y|s;\theta)\right) \quad (20)$$

Though the core capability of the RAML approach lies in sampling from a static distribution, that distribution is difficult to define and we think that the training process can be destabilized when the sampling is introduced during computing gradients. Therefore, in this paper, instead of sampling, we pre-calculate the reward (ROUGE R1 score) for each possible output summary of each document as R1(y, y'). Then after normalizing the scores, the top-scored candidates T (we

use T=25 in our experiments) are used to calculate the weighted cross-entropy loss. During optimization, the weighted cross-entropy loss is defined as follows:

$$\mathcal{L}(y, y') = \sum_{i=1}^{T} -w_i \cdot \sum_{j=1}^{n} y_j^i \ log y_j' \qquad (21)$$

where $y' \in \{0,1\}^n$ demonstrates the vector of predicted sentences to be included in the summary,$y^i \in \{0,1\}^n$ denotes candidate(i) labels, and w_i represents weighted vector of the Rouge scores:

$$w_i = \frac{RougeR1(y',y)}{\sum_j RougeR1(y',y)} \qquad (22)$$

4. EXPERIMENTS

The purpose of our experiments is to answer the following questions: (1) how different architectures of our models influence their performance? (2) how sentence positions affect the performance of the models? (3) how the MLE and RAML influence the generalization capability of the model on out-of-domain datasets? (4) how our models perform compared to the state-of-the-art baselines on CNN and PubMed datasets?

4.1. Datasets

We conduct experiments on two well know datasets from different domains (CNN and PubMed) to evaluate how different biases in each domain can affect the performance of our models. the statistics of the datasets are shown in Table 1.

Table 1. statistics of the datasets used in our experiments (CNN, PubMed): train, valid, and test split. The average number of words in the document and in the summary and the domain they belong to.

Datasets	Number-of-documents			Average Number-of-tokens		Domain
	Train	Validation	Test	Document	Summary	
CNN	90,152	1,220	1,093	761	46	News
PubMed	115,498	6,562	6,602	3,224	203	Scientific paper

CNN is a dataset that was first created by [27] for question answering, then was modified for text summarization task by [28]. This dataset is composed of news articles that are paired with human-generated summaries. For the data preprocessing, the non-anonymized version of the dataset is used in our experiments as in [25].

PubMed is the dataset that was introduced by [29]. the dataset is collected from scientific repositories PubMed.com. The statistics of the dataset are shown in Table 1. In our experiments, we use about 3% of PubMed as validation data and about another 3% for the test; the rest is used for training as in [29].

4.2. Implementation

In our experiments, each document is truncated to 50 sentences and we use padding to keep the lengths of documents. we use pre-trained Glove vectors with 200-dimensions to initialize word embeddings. Weperform mini-batch training with a batch size of 32 documents for 15 training epochs. In the Rnn-based sentence encoder, for each direction, we use a single-layer GRU with 300-dimensional hidden layers, and dropout is applied to GRU with drop probability equals 0.25. In the sentence extractor, for each direction, a single-layered GRU is used with a hidden layer

size of 300. We set the hidden layer size for MLP to 100. The model parameters in the sentence encoder and sentence extractor are initialized using a normal distribution with the Xavier scheme [30]. Our models are optimized using Adam optimizer [31] with an initial learning rate of 0.0001, and momentum parameters $\beta 1 = 0.9, \beta 2 = 0.999$. We use gradient clipping to regularize our models. All our experiments are implementedusing Pytorch on the computer that has 256 RAM and NVIDIA GeForce RTX 2080 Ti GPU.

4.3. Evaluation

We use the Rouge metric [32]to evaluate the quality of the summaries. In the reported experimental results, unigram and bigram overlap (R-1, R-2) are reported as a means of evaluating in formativeness. And longest common subsequence (R-L) is reported as means of evaluating the fluency.

4.4. Model Comparison

We compare the performance of our models with other well-known extractive models including:

- LSA[33]: Extractive model that uses latent semantic analysis approach to discover important sentences
- SumBasic[34]: A summarization model which can generate summaries for single and multi-document.
- LexRank[35]: Based on the idea of eigenvector centrality in a graph representing sentences, this model computes the importance of the sentences.
- NN-SE[7]: A neural network-based extractive model, which can be used to extract words and sentences.
- Refresh[10]: A neural network-based summarization model trained using Reinforcement learning objective to globally optimize evaluation metric(ROUGE).
- Banditsum[36]: A neural extractive summarization model that treats extractive summarization as a context bandit problem.

4.5. Results and Analysis

4.5.1. Influence of Different Architectures on the Performance of the Model.

To understand how different architectures can influence the performance of the model, we examine the performance of Avg (averaging word embeddings) and the Rnn approach to encode sentences at the word level. As it is shown in Figure 2 the approach of averaging word embeddings of each sentence to obtain its representation performs slightly better than the Rnn based sentence representation on the CNN dataset. Whereas on PubMed dataset, the averaging approach results in high performance than the Rnn. Moreover, the time taken to train our averaging word embedding-based model is less than the time taken to train our Rnn based sentence encoder. This implies that it is not always necessary to build very complex architectures to get good performance.

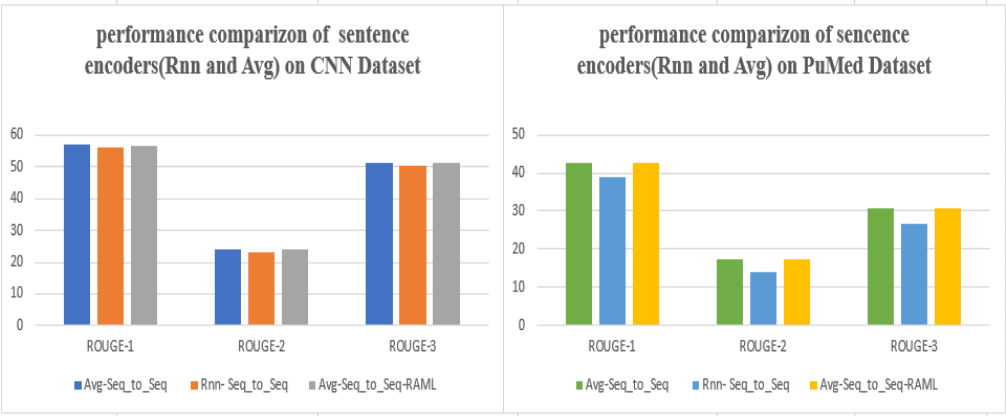

Figure 2. Performance comparison of sentence encoders on CNN and PubMed datasets.

4.5.2. Impact of the Position of the Sentences

When it comes to the extractive summarization for news, the position of the sentence is a very important feature [37]. When deep learning-based summarization models are trained on news datasets, these models mostly select the first sentences in the document and this causes the problem of lead bias. To answer the question (2), we test the performance of our models when the order of the sentences in the document is kept intact and when we shuffle them. As it is shown in Table 2, when sentences are shuffled during training the performance of our model trained based on MLE (Avg-Seq_to_Seq-MLE) drops significantly on CNN and PubMed datasets. This implies that the model has learned the position feature in PubMed/CNN datasets even though the model has no explicit position features. On the other hand, Figure 3 shows that the performance of our model trained using RAML(Avg-Seq_to_Seq-RAML) on shuffle sentences is not significantly dropped. The reason could be that this model is forced to learn from richer distribution of labels. Thus, it is less vulnerable to the lead bias.

Table 2. performance of our models on CNN and PubMed test set using full-length ROUGE F1 scores when using shuffled and in-order sentences during model training.

Models	**Sentence shuffling**	**CNN**			**PubMed**		
		R-1	R-2	R-3	R-1	R-2	R-3
Avg-Seq_to_Seq-MLE	shuffled	52.84	20.14	45.16	39.71	14.53	29.82
	normal	**56.84**	**24.04**	**51.15**	**42.71**	**17.33**	**30.82**
Avg-Seq_to_Seq-RAML	shuffled	56.69	23.76	51.06	42.55	17.26	30.77
	normal	**56.71**	**23. 97**	**51.09**	**42.69**	**17.29**	**30.79**

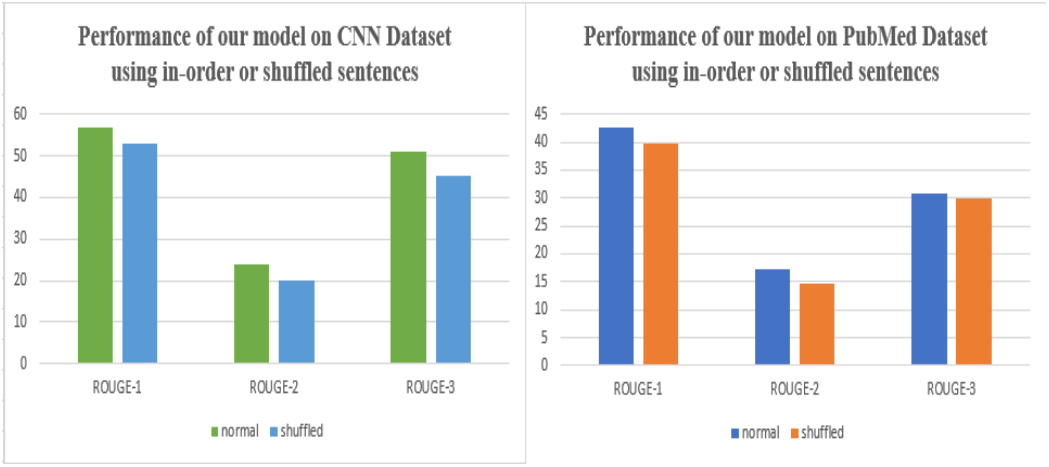

Figure 3. performance of our model trained using MLE approach on CNN and PubMed test set. where shuffled or in-order sentences are used during model training.

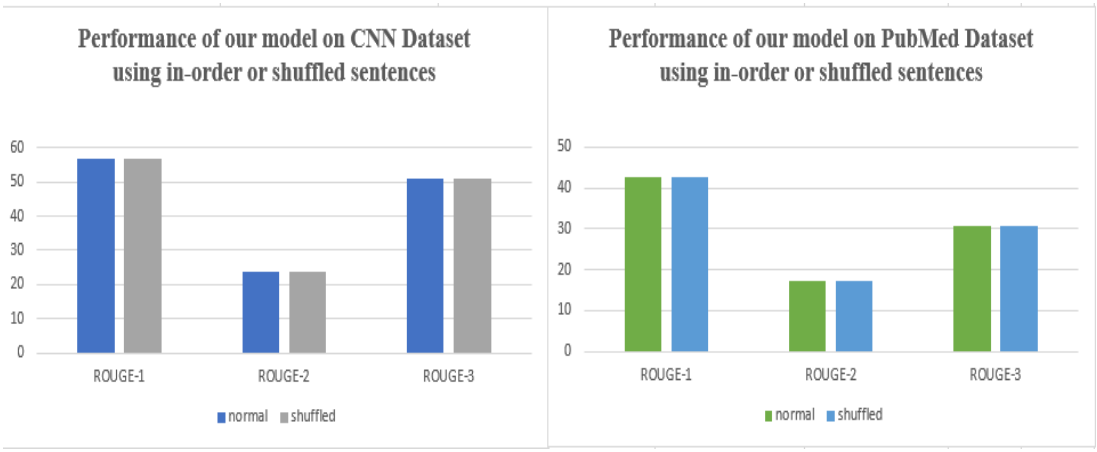

Figure 4. performance of our model trained using RAML approach on CNN and PubMed test set. where shuffled or in-order sentences are used during model training.

4.5.3. Generalization Capability on Out of Domain Dataset

Domain transfer is when a model is trained on one dataset but needs to have a better performance on the other datasets from different domains. Most of the time we want to train the model on a particular domain and be able to reuse it in another domain without retraining it. Let's say, for example we train our model to summarize a dataset of news articles. we do not want to retrain the model if we want to summarize research papers, personal stories, blogs, etc. To answer question (3), we first choose the Avg-Seq_to_Seq model and then train it on the CNN dataset using the MLE approach then transfer this model to the PubMed dataset. As it is shown in Table 3, our Avg-Seq_to_Seq model trained on the CNN dataset achieves 56.84% according to the R-1(ROUGE-1) measure on the test set of the CNN dataset. And the model achieves 35.17% on PubMed test data. The performance of the model drops almost 21.67%. Training models based on the MLE approach tend to cause poor generalization because these models mostly tend to overfit particular features in the training set that might not be in other datasets. We then train our model based on the RAML approach with the expectation that the model will perform better than its counterpart trained based on vanilla Maximum likelihood estimation. RAML approach can incorporate reward task into MLE training and this can improve the generalization ability of the

model. To our surprise, the model trained on the CNN dataset using RAML has slightly better performance on the PubMed dataset compared to the model trained on the CNN dataset using MLE and transferred to PubMed.

Table 3. experiment results for domain transfer, where we transfer a model trained on CNN dataset to PubMed dataset. Meaning we train a model using CNN dataset and test it using the PubMed dataset.

Models (CNN to PubMed)	R-1	R-2	R-3
Avg-Seq_to_Seq-MLE	35.17	11.59	25.26
Avg-Seq_to_Seq-RAML	**35.61**	**11.73**	**25.41**

4.5.4. Performance Comparison of the Summarization Models

To answer the fourth question (4), we inspect the ROUGE scores of the summaries generated by our models (Avg-Seq_to_Seq, Rnn- Seq_to_Seq trained using MLE or RAML) and the baselines that have achieved state-of-the-art results on CNN and PubMed datasets. As it is shown in Table 4, our models have attained significant results in terms of R-1,2, L on CNN dataset compared to Refresh [10], Banditsum [36], and NN-SE [7]. These results approve the efficacy of our models. Our sentence encoder (Avg or Rnn) followed by the attention-based sequence-to-sequence sentence extractor helps to get better representations of the documents and generate good summaries. We also examine our models on the PubMed dataset and compare our results with other summarization models. As it is also shown in Table 4, NN-SE [7] has slightly outperformed our models. This might be caused by the fact that during training, the NN-SE model takes into account previous predictions to inform future predictions while our models do not.

Table 4. the performance comparison of our models with other different extractive summarization modelson the CNN and PubMed test set using full-length ROUGE F-1 scores.

Models	**CNN**			**PubMed**		
	R-1	R-2	R-L	R-1	R-2	R-L
SumBasic+[34]	–	–	–	37.15	11.36	33.43
LSA+[33]	–	–	–	33.89	9.93	29.70
LexRank+[35]	–	–	–	39.19	13.89	**34.59**
Refresh*[10]	30.40	11.70	26.90	–	–	–
Banditsum*[36]	30.70	11.60	27.40	–	–	–
NN-SE*(~)[7]	28.40	10.00	25.00	**43.89**	**18.78**	30.17
Avg- Seq_to_Seq (ours)	**56.84**	**24.04**	**51.16**	42.71	17.33	30.82
Rnn- Seq_to_Seq (ours)	56.18	23.27	50.35	38.95	14.04	26.70
Avg-Seq_to_Seq-RAML (ours)	56.71	23. 97	51.09	42.69	17.29	30.79

In Table 4, the result with * are obtained from [36], results with + are gotten from [29], – illustrates that the correlated result is not reported. and results with *~ are reported from [36] for CNN and from [38] for PubMed respectively. The top section of the table represents traditional approaches; the second and the third sections represent other deep learning-based extractive models and our models respectively.

4.5.5. Run Time Comparison of Our Models based on Sentence Encoder

To train Avg-Seq_to_Seq onCNN and PubMed took8 and 10 hours respectively on a single GPU. Training Rnn-Seq_to_Seq on CNN dataset took 12 hours and 15 hours on PubMed (i.e., training Rnn-Seq_to_Seq took about 1.5 times as much time as training Avg-Seq_to_Seq). Moreover, our models trained using RAML took much time compared to when we train them based on MLE.

5. DISCUSSION

On the CNN dataset, our models generate summaries with sentences from the top of the document rather than from other parts. This means that they are severally hindered by lead bias. We think the lead bias problem is caused by the fact that in news documents most important information is mostly at the beginning of the document, and the details come after. Our RNN-based sequence-to-sequence extractor eagerly learns the position features and heavily relied on them. Shuffling sentences in documents reduces the lead bias however, the overall performance of the models drops; without position, our models are not capable to identify important sentences in news domain. Additionally, there is a drop in the performance of our models on the PubMed dataset. The reason of this, is that the PubMed dataset contains long documents. Our models were not able to learn better representation for these long documents. Graph-based neural network approach and incorporating semantic units such as latent topics, entities and queries can improve extractive summarizer on long documents. We leave this for our future work. Results shown in Table 3 show that our model trained with RAML has potential because it seems to perform better on out-of-domain datasets compared to the model trained with MLE. However, more work is needed to fine-tuneτ hyper-parameter that controls the smoothness of the best distribution around correct labels.

6. CONCLUSIONS

In this paper, we develop a recurrent neural network-based extractive text summarization model and investigate two kinds of hierarchical network structures, to see the effect of different model architectures on the performance of the model. Our experimental results on two datasets from different domains show that our model attains results that are comparable to other deep learning-based state-of-the-art extractive models as well as the state-of-the-art models that use manually engineered features. By comparing the two different approaches of sentence encoders, the performance of Avg-Seq_to_Seq architecture is slightly better than that of Rnn-Seq_to_Seq architecture. We adopt the RAML approach to the task of extractive summarization expecting that it can improve the performance and generalization of our models. Though the RAML approach does not improve the performance of our models over the MLE approach, it poses a potential behavior of improving the generalization ability of the models on out-of-domain datasets. In the future, we plan to investigate more on how different learning criteria used to train neural summarization models influence the generalization and performance of the model. Also, we want to join the extractive approach and abstractive approach to build a model which can generate abstractive summaries. Moreover, we plan to explore more on how different deep learning architectures such as CNNs, RNNs, and transformers influence the performance of other different NLP tasks.

ACKNOWLEDGMENTS

We would like to thank two anonymous reviewers for their helpful comments on various aspects of this work. The work was supported in part by the National Natural Science Foundation of China under grant number 61872186 and Science and Technology on Information System Engineering Laboratory (No.05201901).

REFERENCES

[1] R. Nallapati, F. Zhai, and B. Zhou, “SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents,” 31st AAAI Conf. Artif. Intell. AAAI 2017, pp. 3075-3081, 2017.

[2] M. Isonuma, T. Fujino, J. Mori, Y. Matsuo, and I. Sakata, "Extractive summarization using multi-task learning with document classification," EMNLP 2017 - Conf. Empir. Methods Nat. Lang. Process. Proc., pp. 2101-2110, 2017, doi: 10.18653/v1/d17-1223.

[3] J. Carbonell and J. Goldstein, "Use of MMR, diversity-based reranking for reordering documents and producing summaries," SIGIR Forum (ACM Spec. Interes. Gr. Inf. Retrieval), pp. 335-336, 1998, doi: 10.1145/290941.291025.

[4] D. R. Radev and G. Erkan, "LexRank : Graph-based Centrality as Salience in Text Summarization," J. Artif. Intell. Res., vol. 22, no. 1, pp. 457-479, 2004, [Online]. Available: https://arxiv.org/abs/1109.2128.

[5] J. M. Conroy and D. P. O'leary, "Text summarization via hidden Markov models," SIGIR Forum (ACM Spec. Interes. Gr. Inf. Retrieval), no. August 2014, pp. 406-407, 2001, doi: 10.1145/383952.384042.

[6] R. McDonald, "A study of global inference algorithms in multi-document summarization," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 4425 LNCS, pp. 557-564, 2007, doi: 10.1007/978-3-540-71496-5_51.

[7] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," 54th Annu. Meet. Assoc. Comput. Linguist. ACL 2016 - Long Pap., vol. 1, pp. 484-494, 2016, doi: 10.18653/v1/p16-1046.

[8] W. Yin and Y. Pei, "Optimizing sentence modeling and selection for document summarization," IJCAI International Joint Conference on Artificial Intelligence, vol. 2015-Janua. pp. 1383-1389, 2015.

[9] Z. Cao, F. Wei, S. Li, W. Li, M. Zhou, and H. Wang, "Learning summary prior representation for extractive summarization," ACL-IJCNLP 2015 - 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. Asian Fed. Nat. Lang. Process. Proc. Conf., vol. 2, pp. 829-833, 2015, doi: 10.3115/v1/p15-2136.

[10] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," arXiv, pp. 1747-1759, 2018.

[11] Y. Wu and B. Hu, "Learning to extract coherent summary via deep reinforcement learning," 32nd AAAI Conf. Artif. Intell. AAAI 2018, pp. 5602-5609, 2018.

[12] M. Zhong, P. Liu, D. Wang, X. Qiu, and X. Huang, "Searching for effective neural extractive summarization: What works and what's next," arXiv, pp. 1049-1058, 2019.

[13] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao, "Neural document summarization by jointly learning to score and select sentences," arXiv, pp. 654-663, 2018.

[14] F. Mohsen, J. Wang, and K. Al-Sabahi, "A hierarchical self-attentive neural extractive summarizer via reinforcement learning (HSASRL)," Appl. Intell., vol. 50, no. 9, pp. 2633–2646, 2020, doi: 10.1007/s10489-020-01669-5.

[15] K. Al-Sabahi, Z. Zuping, and M. Nadher, "A hierarchical structured self-attentive model for extractive document summarization (HSSAS)," arXiv, pp. 1-8, 2018.

[16] D. S. Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, "Reward Augmented Maximum Likelihood for Neural Structured Prediction," arXiv, no. Ml, 2017.

[17] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," Inf. Process. Manag., vol. 40, no. 6, pp. 919-938, 2004, doi: 10.1016/j.ipm.2003.10.006.

[18] M. Afsharizadeh, H. Ebrahimpour-Komleh, and A. Bagheri, "Query-oriented text summarization using sentence extraction technique," 2018 4th Int. Conf. Web Res. ICWR 2018, pp. 128-132, 2018, doi: 10.1109/ICWR.2018.8387248.

[19] P. T. Rada Mihalcea, "TextRank: Bringing Order into Texts," Proc. 2004 Conf. Empir. methods Nat. Lang. Process., 2004, doi: 10.1016/0305-0491(73)90144-2.

[20] M. Fuentes, E. Alfonseca, and H. Rodríguez, "Support vector machines for query-focused summarization trained and evaluated on pyramid data," no. June, p. 57, 2007, doi: 10.3115/1557769.1557788.

[21] J. Kupiec, J. Pedersen, and F. Chen, "Trainable document summarizer," SIGIR Forum (ACM Spec. Interes. Gr. Inf. Retrieval), pp. 68-73, 1995, doi: 10.1145/215206.215333.

[22] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., pp. 1-16, 2016.

[23] K. Yao, L. Zhang, T. Luo, and Y. Wu, "Deep reinforcement learning for extractive document summarization," Neurocomputing, vol. 284, pp. 52-62, 2018, doi: 10.1016/j.neucom.2018.01.020.

[24] K. Cho et al., “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf., pp. 1724-1734, 2014, doi: 10.3115/v1/d14-1179.
[25] A. See, P. J. Liu, and C. D. Manning, “Get To The Point: Summarization with Pointer-Generator Networks,” arXiv, 2017.
[26] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1-15, 2015.
[27] K. M. Hermann et al., “Teaching machines to read and comprehend,” Adv. Neural Inf. Process. Syst., vol. 2015-Janua, pp. 1693-1701, 2015.
[28] R. Nallapati and B. Xiang, “Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond Cicero dos Santos,” pp. 280-290, 2016.
[29] A. Cohan et al., “A discourse-aware attention model for abstractive summarization of long documents,” NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf., vol. 2, pp. 615-621, 2018, doi: 10.18653/v1/n18-2097.
[30] Y. B. Xavier Glorot, “Understanding the difficulty of training deep feedforward neural networks Xavier,” AISTATS, 2010, doi: 10.1109/LGRS.2016.2565705.
[31] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1-15, 2015.
[32] C.-Y. Lin and E. Hovy, “Automatic evaluation of summaries using N-gram co-occurrence statistics,” no. June, pp. 71–78, 2003, doi: 10.3115/1073445.1073465.
[33] J. Steinberger and K. Ježek, “Using Latent Semantic Analysis in text summarization and summary evaluation,” Proc. ISIM, pp. 93-100, 2004.
[34] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova, “Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion,” Information Processing and Management, vol. 43, no. 6. pp. 1606-1618, 2007, doi: 10.1016/j.ipm.2007.01.023.
[35] G. Erkan and D. R. Radev, “LexRank: Graph-based lexical centrality as salience in text summarization,” J. Artif. Intell. Res., vol. 22, pp. 457-479, 2004, doi: 10.1613/jair.1523.
[36] Y. Dong, Y. Shen, E. Crawford, H. van Hoof, and J. C. K. Cheung, “Banditsum: Extractive summarization as a contextual bandit,” Proc. 2018 Conf. Empir. Methods Nat. Lang. Process. EMNLP 2018, pp. 3739-3748, 2020, doi: 10.18653/v1/d18-1409.
[37] K. Hong and A. Nenkova, “Improving the estimation of word importance for news multi-document summarization,” 14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014. pp. 712-721, 2014, doi: 10.3115/v1/e14-1075.
[38] W. Xiao and G. Carenini, “Extractive summarization of long documents by combining global and local context,” EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf., pp. 3011-3021, 2020, doi: 10.18653/v1/d19-1298.

AUTHORS

Shimirwa Aline Valerie received her Bachelor’s degree in Software Engineering from Nanjing University of Science and Technology in 2019. Currently, she is working toward a Master’s degree in the Department of Computer Science at Nanjing University of Science and Technology. Her research interests include natural language processing, machine learning, data mining, and deep learning.

Jian Xu received a Ph.D. in Computer Science in 2007 from Nanjing University of Science and Technology, Nanjing, China. Now he holds the position of a professor at Nanjing University of Science and Technology. His research interests are event mining, log mining, and their applications to system management. He has published about 30 papers in journals and refereed conference proceedings in those areas.