Vietnam Journal of Science and Technology ***62*** *(2024)*

doi:10.15625/2525-2518/20241

A method to utilize prior knowledge for extractive summarization based on pre-trained language models

Le Ngoc Thang[1], Nguyen Minh Tien[2, *], Do Nhat Minh[3], Nguyen Chi Thanh, Le Quang Minh[1]

[1]*Information Technology Institute, Vietnam National University, No.144 Xuan Thuy street, Cau Giay district, Ha Noi, Viet Nam*

[2]*Hung Yen University of Technology and Education, Nhan Hoa Ward, My Hao Town, Hung Yen Province, Viet Nam*

[3]*Posts and Telecommunications Institute of Technology, No.122 Hoang Quoc Viet street, Cau Giay District, Ha Noi, Viet Nam*

[4]*Institute of Information Technology, AMST, No.17 Hoang Sam street, Cau Giay district, Ha Noi, Viet Nam*

[*]Email: *tiennm@utehy.edu.vn*

Received: 1 March 2024; Accepted for publication: 3 April 2024

Abstract. This paper presents a novel model for extractive summarization that integrates context representation from a pre-trained language model (PLM), such as BERT, with prior knowledge derived from unsupervised learning methods. Sentence importance assessment is crucial in extractive summarization, with prior knowledge providing indicators of sentence importance within a document. Our model introduces a method for estimating sentence importance based on prior knowledge, complementing the contextual representation offered by PLMs like BERT. Unlike previous approaches that primarily relied on PLMs alone, our model leverages both contextual representation and prior knowledge extracted from each input document. By conditioning the model on prior knowledge, it emphasizes key sentences in generating the final summary. We evaluate our model on three benchmark datasets across two languages, demonstrating improved performance compared to strong baseline methods in extractive summarization. Additionally, our ablation study reveals that injecting knowledge into certain first attention layers yields greater benefits than others. The model code is publicly available for further exploration.

Keywords: Text summurization, knowledge injection, pre-trained models, Transformer, BERT.

Classification numbers: 4.7.4.

1. INTRODUCTION

Text summarization is the process of extracting the most important information from the original text to create a simplified version for different purposes or tasks [1]. Typically a summary is a short and concise version of the original text. The task of automatic document

(text) summarization has been studied for a long time (first introduced by Luhn [2] in the 1950s). The task is challenging due to two main reasons. First, text summarization is in the understanding level of natural language processing (NLP), in which summarization models need to understand the content of input documents deeply to create final summaries. Second, the evaluation of summarization is a non-trivial task that involves both automatic and human evaluation [3]. In practice, the definition of what is a good summary is not so clear and requires experts for judgment. Even using different learning algorithms, the text summarization task can be categorized into extraction and abstraction. Extractive summarization methods work by identifying important sentences of the text to form a summary [4, 5]. On the other hand, abstractive summarization methods aim at re-producing important content of an input document in an abstractive way that mimics humans' brains [6, 7].

There are two main approaches to text summarization. The first approach is to use unsupervised learning methods. The main idea of this approach is to rank sentences based on their importance by using graph-based algorithms [8], integer linear programming [9], matrix factorization [10], submodular functions [11], or text matching [4]. The second approach is to use supervised learning methods. The summarizer can be trained by using sentence classification [12], sequence labeling [13], or learning-to-rank [14] with deep neural networks [15], or text-to-text formulation [16] by using Transformer [17]. Recent methods based on pre-trained language models (PLMs) have achieved promising results [18 - 21].

This paper focuses on the extractive summarization task with the consideration of prior knowledge that conditions the fine-tuning of PLMs. To do that, we introduce a model that takes into account the power of PLMs for context representation and prior knowledge as specific indicators of the downstream task. More precisely, given an input document, the model first maps sentences of the document into contextual vectors. At the same time, the prior knowledge is created from the sentences by using unsupervised learning methods. The prior knowledge is fused to the attention layers of PLMs for fine-tuning. The final summary of a document is formed by extracting top m important sentences based on the assessment of the model. In summary, this paper makes three main contributions as the following.

- It introduces a model that considers prior knowledge for text summarization. The model takes into account prior knowledge that suggests BERT to more focus on certain sentences based on salient indicators. The code is also accessible.
- It confirms the contribution of prior knowledge injection by comparing the model to state-of-the-art (SOTA) methods of extractive summarization. Experimental results on three benchmark datasets show that the proposed model achieves promising results.
- It provides an ablation analysis that examines the behavior of the model in various aspects. The ablation analysis facilitates future studies on text summarization with knowledge injection.

The rest of the paper is organized as follows. Section 2 reviews related works relevant to our study. Section 3 introduces the proposal for extractive summarization with knowledge injection. Section 4 describes the experimental settings for evaluation. Section 5 reports the comparison of the proposed model with SOTA methods. Section 6 draws conclusions and future work.

2. RELATED WORK

This section reviews related work of text summarization in two main approaches: unsupervised learning methods and supervised learning methods.

2.1. Unsupervised learning methods

Unsupervised learning methods for text summarization involve algorithms that do not rely on predefined training data or human inputs. These methods typically utilize statistical analysis and NLP techniques to identify important sentences and phrases in the text for final summary creation. There is a wide range of methods such as the method based on term frequency-inverse document frequency (TF-IDF) [24], clustering [25], latent semantic analysis (LSA) [26], graph-based algorithms [8], discourse-based models [27], integer linear programming [9], non-negative matrix factorization (NMF)[10], submodular functions [11], or text matching [4]. Among these methods, graph-based algorithms [8] and NMF [10] have been widely used due to their efficiency for the extractive summarization task. We follow the graph-based and matrix factorization methods to design our model. In which, the prior knowledge of an input document is represented by a correlation matrix generated from a graph-based or NMF algorithm then fused with the attention weights of PLMs for fine-tuning.

2.2. Deep learning methods

2.2.1. Traditional methods

Supervised learning methods have proven to be efficient when annotated samples are available. Even using different learning algorithms, supervised learning methods can be divided into classification [12], sequence labeling [13], learning-to-rank [14], or text-to-text formulation [16] with the use of the encoder-decoder architecture. For classification, the summarization task is typically formulated as a binary classification problem in which a summarizer is required to distinguish whether an input sentence is important or not [12]. Sequence labeling considers the task of summarization as a sequence tagging problem, in which a summarization model learns to assign a tag (summary or non-summary) for each sentence of an input document [13]. Another direction is learning-to-rank, where summarization models learn knowledge from training data and rank sentences based on their importance [14]. The three mentioned formulations are effective for extractive summarization. To deal with abstractive summarization, the text-to-text formulation was proposed [16]. The formulation relies on the encoder-decoder architecture [28] that allows us to train summarizers in a more natural way. For representation learning, summarization models are trained by using a wide range of indicators, from hand-crafted features [13] to features automatically learned from deep neural networks [15,16].

2.2.2. Pre-trained models

The emergence of Transformer [17] boosts the performance of summarization models, in which pre-trained-based summarizers obtain the best results on various benchmark datasets [12,18,19,21]. For example, HIBERT (hierarchical bidirectional encoder representations) uses a hierarchical approach to summarize long documents [18]. The model first encodes individual sentences using a bidirectional transformer. Then, a second transformer is used to encode the document as a whole, taking into account the interactions between sentences. Finally, a decoder transformer generates the summary based on the encoded document. PNBERT is a pre-trained model for text summarization [19], based on the BERT architecture and pointer network. BERTSum is a variant model of BERT, which uses fine tuning layers to add document context from the BERT outputs for extractive summarization [12]. DiscoBERT is a pre-trained transformer model based on BERT that considers the discourse aspect of a document [21].

We follow the direction of using supervised learning methods for text summarization. However, instead of training the model from scratch, we utilize BERT as the backbone of the

model to take advantage of context representation from PLMs. This allows the proposed model to automatically learn hidden features from data and effectively represent input tokens. Though sharing the idea of using prior knowledge with previous studies [15,23], our model distinguishes three important points. First, we define prior knowledge as hidden features learned from unsupervised learning methods, e.g., LexRank or NMF, while Cao et al. [15] used a set of hand-crafted features (sentence position or sentence length) as prior representation. Our definition is more general and allows us to utilize hidden indicators from any machine learning methods without human involvement. Second, we use BERT as the backbone while Cao et al. trained the model from scratch using CNN. In addition, Xia et al. [23] used prior knowledge in textual matching while we employ knowledge injection for text summarization, which is a more challenging task than matching.

3. SUMMARIZATION WITH KNOWLEDGE INJECTION

3.1. Problem Statement

Prior knowledge

Given a document $\boldsymbol{d} = \{\boldsymbol{s_1}, \boldsymbol{s_2} \ldots, \boldsymbol{s_n}\}$ that includes $\boldsymbol{n}$ sentences, the prior knowledge of the document contains indicators that show the importance of each sentence in $\boldsymbol{d}$. The prior knowledge is defined as a matrix $\boldsymbol{A} = [\boldsymbol{n} \times \boldsymbol{n}]$ created from $\boldsymbol{d}$ by using some similarity computing algorithms. $\boldsymbol{A}[\boldsymbol{i}, \boldsymbol{j}]$ shows the correlation between two sentences $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$.

The summarization task

Given a document $\boldsymbol{d} = \{\boldsymbol{s_1}, \boldsymbol{s_2} \ldots, \boldsymbol{s_n}\}$, the extractive summarization task is formulated as a classification task in which the summarization model has to assign a label (important or unimportant) for each sentence. This study focuses on using prior knowledge for extractive summarization. Formally, suppose $\mathfrak{D}$ is a set of training documents, given a sentence $\boldsymbol{s_i} \in \boldsymbol{d}$, the probability of $\boldsymbol{s_i}$ to be included into the final summary is the conditional probability $\hat{y}_i$ [1] computed as $\boldsymbol{p}(\hat{y}_i|\boldsymbol{\theta}, \boldsymbol{A}, \mathfrak{D})$. The final summary $\boldsymbol{S}$ includes some sentences $\boldsymbol{s_i}$ that are predicted as important. The hyper-parameter $\boldsymbol{\theta}$ can be learned from the training set $\mathfrak{D}$ with or without using the prior knowledge $\boldsymbol{A}$.

3.2. The Proposed Model

There exists correlation among sentences in a document. The correlation allows us to estimate the importance of a sentence compared to others. Based on the correlation, there are several unsupervised methods for extractive summarization such as LexRank [8]. We argue that the correlation among sentences can be a useful indicator that can guide the learning process to focus more on important sentences. To encode the correlation into the learning process, we introduce a new extractive summarization model. The model uses BERT [22] as the main backbone and the correlation indicator is injected into BERT via attention computation.

Figure 1 shows the proposed model for extractive summarization by using knowledge injection. Given a document d with **n** sentences, the model first creates the knowledge of **d** in the

[1] We use $\hat{y}_i$ as the predicted label to distinguish the gold label y_i.

form of a matrix $\boldsymbol{A}$. The model then adds [CLS] and [SEP] tokens to concatenate n sentences to form a new single input sequence.

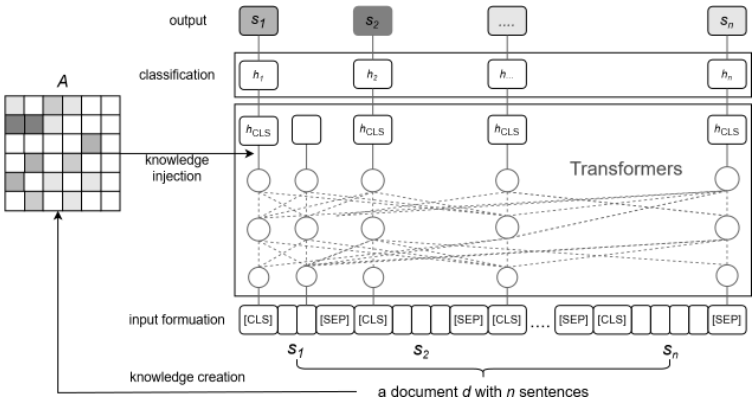


Figure 1. The extractive summarization model using prior knowledge. The rectangles of each sentence represent its tokens. Sentences with darker colour in the output layer are more important than others.

The new sequence is fed into the transformer architecture to obtain contextual vectors of each token. The knowledge from the matrix $\boldsymbol{A}$ is injected into attention layers of BERT during the fine-tuning process that conditions the representation of sentences. Finally, the model uses the vector from [CLS] tokens for classification that estimates the importance of each sentence. Sentences with high confidence of importance are selected to form the final summary. The model shares the idea of fine-tuning BERT for summarization [24]. However, instead of purely using BERT, we introduce prior knowledge that encodes the correlation among sentences and injects the knowledge to BERT. The injection forces the attention of BERT to focus more on some important sentences that help to improve the quality of summarization.

3.2.1. Prior knowledge creation

As mentioned, there exists a correlation among sentences in a document. The correlation is beneficial for the estimation of the importance of sentences. This section shows the creation of prior knowledge in three algorithms: Cosine similarity, LexRank, and matrix factorization.

Cosine similarity

Given a document $\boldsymbol{d} = \{\boldsymbol{s_1}, \boldsymbol{s_2} \ldots, \boldsymbol{s_n}\}$, a matrix **A** = **[n×n]** is constructed. The matrix **A** is used to represent the correlation among n sentences, in which each cell **A[i,j]** represents the correlation of the two sentences $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$. Given two sentences $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$, the first step is to project these sentences into two vectors in a multi-dimensional space. The projection uses SentenceBERT [29] for mapping input sentences into corresponding vectors. After mapping, the correlation between two sentences was computed by using the Cosine similarity.

$$cos(a, b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}} \quad (1)$$

LexRank

Lexrank is an algorithm for computing the similarity between two or more pieces of text based on the concept of eigenvector centrality in a graph of sentences [8]. It has been widely

applied to extractive summarization [30]. For using LexRank, an adjacency matrix $\boldsymbol{A}$ based on the Cosine similarity among all sentences was created. Only similarity scores above or equal to 0.1 were beneficial for the graph structure.

$$A[i,j] = \begin{cases} 1 & if\ cos(A[i,j]) \geq 0.1 \\ 0 & otherwise \end{cases} \quad (2)$$

Once the adjacency matrix $\boldsymbol{A}$ has been created, LexRank computes the connectivity matrix $\boldsymbol{C}$. The connectivity matrix was created by normalizing the row-wise of the adjacency matrix $\boldsymbol{A}$.

$$C[i,j] = \frac{A[i,j]}{\sum(A[i,:])} \quad (3)$$

Matrix factorization

Non-negative matrix factorization (NMF) [31] is a dimensionality reduction technique used for analyzing multivariate data. It decomposes a non-negative matrix into the product of two non-negative matrix, typically a data matrix and a parts-based representation matrix. NMF has been shown as an effective method for extractive summarization [10].

Let X be a non-negative matrix with the size of $m \times n$, where m represents the number of features (variables) and n represents the number of samples (data points). NMF aims to factorize this matrix into two non-negative matrices, W and H by the following approximation.

$$X \approx W.H \quad (4)$$

To apply NNF, a Cosine similarity matrix $\boldsymbol{X}$ (similar to the matrix **A**) was first created. After that, NMF factorizes $\boldsymbol{X}$ into two matrices $\boldsymbol{W}$ and $\boldsymbol{H}$. Then the matrix $\boldsymbol{W}$ was fed into the multi-head attention of BERT.

3.2.2. Input representation

Input formulation

Given a document $\boldsymbol{d} = \{\boldsymbol{s_1}, \boldsymbol{s_2} \ldots, \boldsymbol{s_n}\}$ that includes **n** sentences, the first step is to create an input sequence for embedding mapping. To do that, two special tokens: [CLS] and [SEP] were used for concatenation. More precisely, the [CLS] token was inserted in the head of each sentence to represent the meaning of the sentence. The [SEP] token was inserted into the tail of each sentence for the separation of two sentences. The concatenation of **n** sentences creates a new input sequence for embedding by using the Transformer architecture. Input creation is shown in the input formulation part in Figure 1.

Transformers

Transformer is a modern deep learning architecture that adopts the mechanism of self-attention, differentially weighting the significance of each part of input sequences [17]. It consists of an encoder and a decoder, each composes of a stack of N identical layers. The encoder processes the input, while the decoder processes the encoder's output vectors to obtain output tokens. Each encoder layer contains two sub-layers self-attention and feed-forward, and each decoder layer contains three sub-layers: sef-attention, encoder-decoder attention, and feed-forward. Transformation functions compute attention by using query (Q), key (K), and value (V) matrices as follows.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (5)$$

where d_k is the dimension of keys, and d_v is the dimension of values. Moreover, the Transformer uses "multi-head attention" in parallel as the following:

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^O \quad (6)$$

Where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ in which $W_i^K \in \mathbb{R}^{d_{modle} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. While any PLMs can be used, this study uses BERT [22] for input mapping.

Input mapping

Suppose *S* is the newly formed sequence after concatenating all sentences using [CLS] and [SEP] tokens. The sequence *S* can be represented as follows.

$$S = [CLS]\ s_1\ [SEP][CLS]\ s_2\ [SEP]...[CLS]\ s_n\ [SEP]$$

The vector of the [CLS] token outputted from BERT was used as the representation of each sentence.

$$\boldsymbol{H} = Encoder(S \quad (7)$$

More precisely, the contextual vector $\boldsymbol{H}$ can be interpreted as a set of hidden vectors of all *n* [CLS] tokens corresponding to *n* sentences in the sequence *S*.

$$\boldsymbol{H} = \{h_1,h_2,...,h_n\} \quad (8)$$

3.2.3. Knowledge injection

For knowledge creation, the SentenceBERT [29] was used to compute sentence embeddings and then formed one of the three matrices (the Cosine matrix, the LexRank matrix, and the NMF matrix) $\boldsymbol{A}$. The input embeddings are the sum of three parts: token embedding, position embedding, and segment embedding. BERT's attention functions can be described as a mapping from the query vector *Q* and a set of key-value vector pairs (*K*, *V*) to an output vector - the attention strengths. Figure 2 summarizes the injection into attention layers. The dot products of the query with all keys were computed as follows.

$$scores = QK^T \quad (9)$$

The prior knowledge was injected into the model by calculating the element-wise product of the scores with one of three matrices (Cosine similarity matrix, LexRank matrix, non-negative matrix factorization) $\boldsymbol{A}$ to make the model pay more attention to the sentence pairs with higher similarities in the document. After that the calculation was scaled down by $\sqrt{d_k}$ and applied a softmax function to obtain the weights on the values.

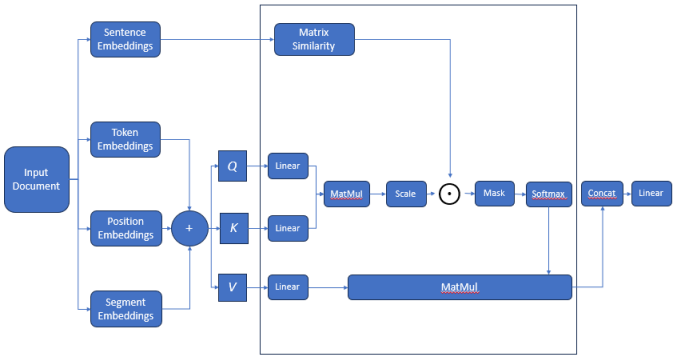


Figure 2. Knowledge injection into BERT's multi-head attention.

$$scores = QK^T \odot A + MASK \tag{10}$$

$$Attention(Q, K, V) = softmax(\frac{scores}{\sqrt{d_k}}) V \tag{11}$$

where A represents the prior knowledge in form of a correlation matrix. Based on the investigation in Section 5.2, the prior knowledge was only injected into the first and two first attention layers of BERT.

After injection in *Figure 2*, the hidden representation of the input sequence S is represented as follows.

$$\widehat{\boldsymbol{H}} = \{\hat{h}_1, \hat{h}_2, \dots, \hat{h}_n\} \tag{12}$$

The representation of $\widehat{\boldsymbol{H}}$ is the input for classification that estimates the importance of sentences corresponding to each hidden vector $\widehat{\boldsymbol{h}}_{\boldsymbol{i}}$. The final output layer is a sigmoid classifier:

$$\hat{y}_i = \sigma(W_o \hat{h}_i + b_o) \tag{13}$$

where $\hat{y}_i$ is the predicted probability of the sentence s_i that shows the importance of the sentence. The probability was used for sentence selection.

3.2.4. Sentence selection

After training, the model was applied to test sets to extract summaries of documents. After ranking predicted sentences based on their importance (scores), the model uses the Maximal Marginal Relevance (MMR) algorithm [32] to form final summaries. MMR iteratively constructs a summary by including the highest-scoring sentence with the following formula:

$$S_{next} = max_{s \in S - S_{sum}}(0.7 * f(s) - 0.3 * sim(s, S_{sum})) \tag{14}$$

where S is the set of all the sentences in the document, S_{sum} contains current sentences in the summary, $f(s)$ is the sentence score from the model, $sim()$ is the Cosine similarity of the sentence to S_{sum}. When extracting a summary of a new document, MMR first uses the model to obtain the score for each sentence. It then ranks these sentences by their scores to create a sorted list (decreasing) and selects the top-m ranked sentences as the summary. During sentence selection, MMR uses trigram blocking to reduce redundancy.

3.2.5. Training and inference

For training, the model receives an input document and injects knowledge from the document into the training process. The representation of contextual vectors of [CLS] tokens is used for prediction that decides whether a sentence is important or not, utilizing a sigmoid function. The loss function of the model is the binary cross-entropy loss, which measures the dissimilarity between the predicted probability $\hat{y}_i$ and the target label y_i over N training samples.

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \, log(\hat{y}_i) + (1 - y_i) \, log(1 - \hat{y}_i)] \tag{15}$$

For inference, given an input document, the trained model estimates the importance of each sentence by using knowledge injection. After prediction, m important sentences (which have the highest probabilities) are selected to form the final summary by using the MMR algorithm.

4. EXPERIMENTAL SETTINGS

4.1. Datasets

The evaluation used three benchmark datasets of text summarization in two languages: English and Vietnamese.

Table 1. Three benchmark summarization datasets.

DATA	Train	Dev	Test	Domain	Language
CNN-DailyMail	286,817	13,368	11,487	News	English
BillSum (US)	18,949	----	1,132	Bill	English
BillSum (CA)	----	----	1,237	Bill	English
VNDS	105,418	22,642	22,644	News	Vietnamese

CNN-DailyMail is a benchmark English-language dataset for single-document summarization [33]. This corpus contains 286,817 training samples, 13,368 validation samples, and 11,487 test samples collected from CNN and Daily Mail[2]. BillSum [34] is a dataset collected from the Govinfo service provided by the United States Government Publishing Office (GPO) for the summarization of US Congressional and California state bills. It consists of three parts: US training bills, US test bills (US), and California test bills (CA). VNDS is a benchmark dataset for Vietnamese summarization [30]. The dataset was collected from news providers in Vietnam. The dataset consists of 150,704 documents and was divided into three sets in the following ratio: 70 % for training, 15 % for development, and 15 % for testing. Each document contains a title, a gold summary, and sentences.

Table 1 summarizes the information of the datasets. It shows that datasets are in different domains and languages, and it can create challenges for summarization models.

4.2. Implementation

We used PyTorch and the "bert-base-uncased" version of BERT to implement the model. The English version of BERT (BERT-base) [22] was used for CNN/DailyMail and BillSum and the Vietnamese version of BERT (phoBERT-base) [35] was used for VNDS. For pre-processing, we utilized BERT's tokenizer for English [22] and PhoBERT's tokenizer for Vietnamese [35]. SentenceBert [29] was used to tokenize each sentence for creating prior knowledge from the Cosine similarity, LexRank, and non-negative factorization algorithms. All extractive models were trained in 50,000 steps with the learning rate of $2e^{-3}$ on an A100 GPU. We use the Adam optimizer was used with β_1 = 0.9 and β_2 = 0.999. The learning rate follows the warming-up method with a value of 10000.

$$lr = 2e^{-3}.min(step^{-0.5}, step.warmup^{-1.5}) \quad (17)$$

Model checkpoints were saved and evaluated on the validation set after every 1,000 steps. Top-3 checkpoints were selected based on the evaluation loss on the validation set and reported the averaged results on the test set. For the CNN/DailyMail dataset, after ranking, the top-3 sentences having the highest scores were extracted to form a summary [18–20]. For the BillSum dataset, when predicting a summary for a new document, the model uses the MMR algorithm to select important sentences until reach the length limit of 2000 characters [34]. For VNDS, the model selects top-2 sentences to form a summary [30].

4.3. Evaluation Metrics

[2] http://nlpprogress.com/english/summarization.html

ROUGE-scores [36] were used for evaluation. For evaluation, gold summaries and extracted sentences were compared for score calculation as follows.

$$ROUGE - N = \frac{\sum_{s \in S_{ref}} \sum_{gram_n \in s} Count_{match}(gram_n)}{\sum_{s \in S_{ref}} \sum_{gram_n \in s} Count(gram_n)} \quad (18)$$

where n is the length of n-gram, $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and the references, $Count(gram_n)$ is the number of n-grams in the references. In practice, we employed ROUGE-1.5.5 by using pyrouge[3] with parameters *"-c 95 -2 -1 -U -r 1000 -n 2 -w 1.2 -a -s -f B -m"*. The packages provide several ROUGE-scores and the F-score of ROUGE-1, ROUGE-2, and ROUGE-L was used as major metrics of text summarization [4,18,19,21,34].

5. RESULTS AND DISCUSSION

This section summarizes the comparison of the proposed model and strong methods for extractive summarization. It first shows the performance comparison and then reports the ablation study. It finishes by showing the observation of outputs from the model.

5.1. Performance Comparison

5.1.1. Knowledge injection vs. no knowledge injection

The first scenario is to confirm the contribution of knowledge injection to the model. To do that, the model was trained with two settings. The first setting uses the whole architecture in Figure 1 with knowledge injection.

Table 2. Extractive summarization performance. bold is the best with $p \leq 0.05$ with the pair-wise t-test.

Data	Method	ROUGE-1	ROUGE-2	ROUGE-L
CNN-DailyMail	No injection	42.80	19.19	39.37
	Injection (Cosine)	**43.13**	**20.19**	**39.54**
	Injection (LexRank)	42.96	20.05	39.53
	Injection (NMF)	42.87	20.08	39.45
BillSum (US)	No injection	41.56	21.23	38.35
	Injection (Cosine)	**42.09**	**21.87**	38.54
	Injection (LexRank)	41.85	21.56	**38.69**
	Injection (NMF)	41.85	21.56	38.55
BillSum (CA)	No injection	44.61	19.24	41.13
	Injection (Cosine)	44.69	19.28	41.33
	Injection (LexRank)	**44.96**	**19.84**	**41.51**
	Injection (NMF)	44.72	19.45	41.26
VNDS	No injection	52.80	23.96	37.13
	Injection (Cosine)	**53.43**	**24.35**	**37.55**
	Injection (LexRank)	52.89	24.01	37.02
	Injection (NMF)	52.91	23.99	36.46

The second setting removes the injection. The ROUGE-scores in Table 2 show that knowledge injection benefits the proposed model for extractive summarization. The ROUGE-

[3] https://github.com/andersjo/pyrouge

scores of the proposed model are consistently better than the baseline that does not use knowledge injection. Among the three knowledge injection methods, matrices created by Cosine similarity of LexRank are more appropriate. There exist tiny gaps between the two models: using knowledge injection and without using knowledge injection. It shows that the backbone model using BERT is a strong method for extractive summarization [20]. By using training samples for fine-tuning, it can achieve competitive results.

5.1.2. Comparison with SOTA models

The second scenario is to challenge the proposed model to strong extractive summarization methods on the four test sets. This scenario confirms the gap between the proposed model and SOTA methods which include the following methods: **HIBERT** [18], **PNBERT** [19], **BERTSum** [20], **DiscoBERT** [21], **MatchSum** [4], **DOC** [34], **SUM** [34], **DOC+SUM** [34], **SumBasic** [30], **SVR** [30], **CNN** [30], **LSTM** [30]. All the results were extracted from the original papers.

Table 3. Extractive summarization performance. bold is the best. The results of CNN-DailyMail were extracted from the NLP progress page.

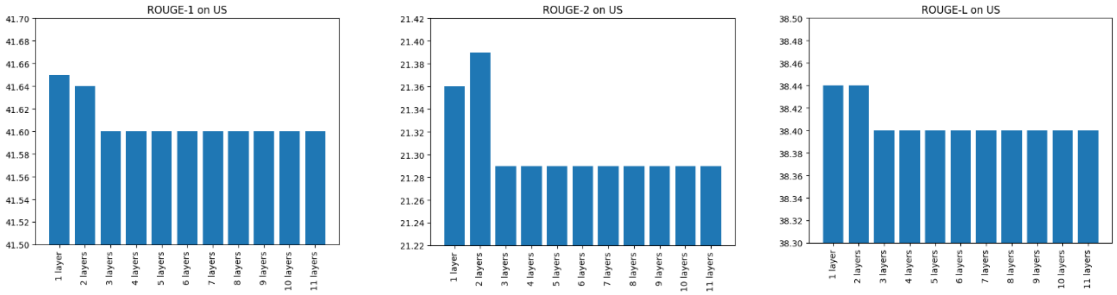
Data	**Method**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
CNN-DailyMail	HIBERT [18]	42.37	19.95	38.83
	PNBERT [19]	42.69	19.60	38.83
	BERTSum [20]	43.85	20.34	39.90
	DiscoBERT [21]	43.77	20.85	40.67
	MatchSum [4]	**44.41**	**20.86**	**40.55**
	Our model (Cosine)	43.13	20.19	39.54
	Our model (LexRank)	42.96	20.05	39.53
	Our model (NMF)	42.87	20.08	39.45
BillSum (US)	DOC [34]	38.51	21.38	31.49
	SUM [34]	40.69	23.88	33.65
	DOC+SUM [34]	40.80	23.83	33.73
	Our model (Cosine)	**42.09**	**21.87**	38.54
	Our model (LexRank)	41.85	21.56	**38.69**
	Our model (NMF)	41.85	21.56	38.55
BillSum (CA)	DOC [34]	38.35	19.76 20.79	32.89
	SUM [34]	38.90	**21.14**	33.20
	DOC+SUM [34]	39.65		34.05
	Our model (Cosine)	44.69	19.28	41.33
	Our model (LexRank)	**44.96**	19.84	**41.51**
	Our model (NMF)	44.72	19.45	41.26
VNDS	SumBasic [30]	52.65	19.13	26.32
	SVR [30]	50.41	23.67	35.02
	CNN [30]	48.17	21.93	33.73
	LSTM [30]	46.56	20.29	32.49
	Our model (Cosine)	**53.43**	**24.35**	**37.55**
	Our model (LexRank)	52.89	24.01	37.02
	Our model (NMF)	52.91	23.99	36.46

The ROUGE-scores in Table 3 are consistent with the scores in *Table 2* in which the proposed model obtains promising results. For CNN-DailyMail, MatchSum [4] is the best followed by DiscoBERT [21], BERTSum [20], and our model. This is because MatchSum uses

semantic matching by using SiameseBERT that can correctly match a gold summary with candidates. The DiscoBERT model encodes the discourse structure into the summarization process by using a discourse graph encoder. In contrast, the proposed model with quite simple architecture can be competitive with two complicated methods. However, the gap between the proposed model and the two strong methods is small. Compared to HIBERT and PNBERT, the proposed model is better. For BillSum and VNDS, the proposed model exhibits a significant improvement over the baseline methods. These results confirm the effectiveness of our proposed method, which uses knowledge injection to improve the quality of sentence selection.

5.2. Ablation Study

This section investigates the behavior of attention layers when injecting prior knowledge. The behavior of the model when using knowledge injection was examined to assess the impact of prior knowledge on attention layers. To do that, prior knowledge was injected into each layer ($1 \leq L \leq 11$). For example, with $L = 1$, the knowledge from LexRank was injected into the first layer of BERT. With $L = 2$, the knowledge from LexRank was injected into the first and second layers of BERT. After that, the model was retrained on the BillSum dataset and evaluated on the test sets of US bills.



(a) ROUGE-1 (b) ROUGE-2 (a) ROUGE-L

Figure 3. Knowledge injection from LexRank on US BillSum for each layer.

The results depicted in *Figures 3* provide insights of the performance of the proposed model with knowledge injection from LexRank at different layers. The findings indicate that the model achieves the best results when knowledge is injected into the first layer or the first two layers of the BERT model. This suggests that introducing prior knowledge early in the model's architecture enables it to effectively capture the relationships and salient information between sentences in a document. After the two first layers, the performance of the model decreases and keeps stablely until the final layer. By summarizing the results, we can conclude that incorporating prior knowledge into the first layer or the two first layers of the BERT model enhances its ability to learn and understand the correlation among sentences within a document.

6. CONCLUSION AND FUTURE WORK

This paper introduces a model for extractive summarization. The model is designed to take into account context representation from a PLM, i.e., BERT and prior knowledge obtained from unsupervised learning methods. The prior knowledge is injected into attention layers of BERT, enabling the model to focus more on important sentences based on prior indicators. Experimental results on three benchmark datasets with four test sets in two languages confirm two important points. First, prior knowledge contributes to improve the performance of the

summarization model. Second, knowledge injection is more beneficial for some first layers rather than for all attention layers. The proposed model provides a framework for injecting various types of prior knowledge into the attention layers of PLMs such as BERT for extractive summarization.

Future work will confirm the ability of the proposed model on different genres. Also, the investigation of knowledge injection in other summarization scenarios, e.g., abstraction should be considered.

Acknowledgements. This work was supported by the Ministry of Education and Training, Vietnam, under grant number B2024-SKH-01.

CRediT authorship contribution statement. Ngoc-Thang Le and Minh-Tien Nguyen: wrote the main manuscript. Minh-Tien Nguyen: idea formulation and experimental supervision. Nhat-Minh Do: implementation and wrote the draft of the method part. Chi-Thanh Nguyen: idea discussion, editing, and proofreading. Quang-Minh Le: idea discussion, editing, and proofreading. All authors reviewed the manuscript.

REFERENCES

1. Nenkova A., McKeown K., *et al*. - Automatic summarization. Foundations and Trends® in Information Retrieval **5** (2 - 3) (2011) 103-233.
2. Luhn H. P. - The automatic creation of literature abstracts, IBM Journal of Research and Development **2** (2) (1958) 159-165. https://doi.org/10. 1147/rd.22.0159
3. Ermakova L., Cossu J. V., Mothe J. - A survey on evaluation of summarization methods. Information processing & management **56** (5) (2019) 1794-1814.
4. Zhong M., Liu P., Chen Y., Wang D., Qiu X., Huang X. J. - Extractive summarization as text matching, In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 6197-6208.
5. Ghadimi, A., Beigy H. - Sgcsumm: An extractive multi-document summarization method based on pre-trained language model, submodularity, and graph convolutional neural networks, Expert Systems with Applications **215** (2023) 119308.
6. Akiyama K., Tamura A., Ninomiya T. - Hie-bart: Document summarization with hierarchical bart, In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, 2021, pp. 159-165.
7. Qiu Y., Cohen S. B. - Abstractive summarization guided by latent hierarchical document structure, arXiv preprint arXiv:2211.09458 (2022).
8. Erkan G., Radev D. R. - Lexrank: Graph-based lexical centrality as salience in text summarization, Journal of artificial intelligence research **22** (2004) 457-479.
9. Nguyen M. T., Tran D. V., Nguyen L. M., Phan X. H. - Exploiting user posts for web document summarization, ACM Transactions on Knowledge Discovery from Data (TKDD) **12** (4) (2018) 1-28.
10. Nguyen M. T., Cuong T. V., Hoai N. X., Nguyen M. L. - Utilizing user posts to enrich web document summarization with matrix cofactorization, In: Proceedings of the 8th International Symposium on Information and Communication Technology, 2017. pp. 70-77.

11. Lin H., Bilmes J. - A class of submodular functions for document summarization, In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp. 510-520.
12. Liu Y. - Fine-tune BERT for Extractive Summarization, 2019.
13. Shen D., Sun J. T., Li H., Yang Q., Chen Z. - Document summarization using conditional random fields, IJCAI **7** (2007) 2862-2867.
14. Cao Z., Wei F., Dong L., Li S., Zhou M. - Ranking with recursive neural networks and its application to multi-document summarization, In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, 2015.
15. Cao Z., Wei F., Li S., Li W., Zhou M., Wang H. - Learning summary prior representation for extractive summarization. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Vol. 2, 2015, pp. 829-833.
16. Lewis M., Liu Y., Goyal N., Ghazvininejad M., Mohamed A., Levy O., Stoyanov V., Zettlemoyer L. - Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871-7880.
17. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. - Attention Is All You Need, 2017.
18. Zhang X., Wei F., Zhou M. - Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization, In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5059-5069.
19. Zhong M., Liu P., Wang D., Qiu X., Huang X. J. - Searching for effective neural extractive summarization: What works and what's next, In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 1049-1058.
20. Liu Y., Lapata M. - Text Summarization with Pretrained Encoders, 2019.
21. Xu J., Gan Z., Cheng Y., Liu J. - Discourse-aware neural extractive text summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 5021-5031.
22. Devlin J., Chang M. W., Lee K., Toutanova K. - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.
23. Xia T., Wang Y., Tian Y., Chang Y. - Using prior knowledge to guide bert's attention in semantic textual matching tasks, In: Proceedings of the Web Conference 2021, 2021, pp. 2466-2475.
24. Christian H., Agus M., Suhartono D. - Single document automatic text summarization using term frequency-inverse document frequency (tf-idf), ComTech: Computer, Mathematics and Engineering Applications **7** (285) (2016). https://doi.org/10.21512/comtech.v7i4.3746
25. Km S., Soumya R. - Text summarization using clustering technique and svm technique 10, (2015) 25511–25519
26. Steinberger J., Jezek K. - Using latent semantic analysis in text summarization and summary evaluation, 2004.

27. Yadav A., Maurya A. K., Ranvijay R., Yadav R. - Extractive text summarization using recent approaches: A survey, Ing´enierie des syst`emes d information 26, 2021, pp. 109-121. https://doi.org/10.18280/isi.260112
28. Sutskever I., Vinyals O., Le Q. V. - Sequence to sequence learning with neural networks, Advances in neural information processing systems **27** (2014)
29. Reimers N., Gurevych I. - Sentence-bert: Sentence embeddings using siamese bert-networks, In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3982-3992.
30. Nguyen V. H., Nguyen T. C., Nguyen M. T., Hoai N. X. - Vnds: A vietnamese dataset for summarization, In: 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), 2019, pp. 375-380. https://doi.org/10.1109/NICS48868.2019.9023886
31. Lee D., Seung H. S. - Algorithms for non-negative matrix factorization, Advances in neural information processing systems **13** (2000)
32. Carbonell J., Goldstein J. - The use of mmr, diversity-based reranking for reordering documents and producing summaries, In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998, pp. 335-336.
33. Nallapati R., Zhou B., dos santos C. N., Gulcehre C., Xiang B. - Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond, 2016.
34. Kornilova A., Eidelman V. - BillSum: A corpus for automatic summarization of US legislation, In: Proceedings of the 2nd Workshop on New Frontiers in Summarization, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 48-56. https://doi.org/10.18653/ v1/D19-5406. https://aclanthology.org/D19-5406
35. Nguyen D. Q., Nguyen A. T. - Phobert: Pre-trained language models for Vietnamese, In: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 1037-1042.
36. Lin C. Y., Hovy E. - Automatic evaluation of summaries using n-gram co-occurrence statistics, In: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2003, pp. 150-157.
37. Nenkova A., Vanderwende L. - The impact of frequency on summarization, Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005 101, 2005.