Building Legal Case Retrieval Systems with Lexical Matching and Summarization using A Pre-Trained Phrase Scoring Model

Vu Tran
vu.tran@jaist.ac.jp
Japan Advanced Institute of Science and Technology

Minh Le Nguyen
nguyenml@jaist.ac.jp
Japan Advanced Institute of Science and Technology

Ken Satoh
ksatoh@nii.ac.jp
National Institute of Informatics
Japan

ABSTRACT

We present our method for tackling the legal case retrieval task of the Competition on Legal Information Extraction/Entailment 2019. Our approach is based on the idea that summarization is important for retrieval. On one hand, we adopt a summarization based model called encoded summarization which encodes a given document into continuous vector space which embeds the summary properties of the document. We utilize the resource of COLIEE 2018 on which we train the document representation model. On the other hand, we extract lexical features on different parts of a given query and its candidates. We observe that by comparing different parts of the query and its candidates, we can achieve better performance. Furthermore, the combination of the lexical features with latent features by the summarization-based method achieves even better performance. We have achieved the state-of-the-art result for the task on the benchmark of the competition.

CCS CONCEPTS

• **Information systems → Document structure**; **Content analysis and feature selection**; **Learning to rank**; **Summarization**.

KEYWORDS

legal texts, deep learning, document representation, structure analysis, information retrieval

ACM Reference Format:
Vu Tran, Minh Le Nguyen, and Ken Satoh. 2019. Building Legal Case Retrieval Systems with Lexical Matching and Summarization using A Pre-Trained Phrase Scoring Model. In *Seventeenth International Conference on Artificial Intelligence and Law (ICAIL '19), June 17–21, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3322640.3326740

1 INTRODUCTION

Automatic legal document processing systems can speed up significantly the work of experts, which, otherwise, requires significant time and efforts. One crucial kind of such systems, automatic information retrieval whose systems, in place of experts, process over enormous amount of documents, for example, legal case reports and statute law documents, which are accumulated rapidly over time. One of the challenging tasks is legal case retrieval task, where the corresponding systems, in place of experts, process over enormous amount of legal case documents which are accumulated rapidly over time (the number of filings in the U.S. district courts for civil cases and criminal defendants is 344,787 in 2017 [1]).

Legal case retrieval or retrieval of prior cases is an important research topic for decades where approaches to solve the corresponding task involve performing linguistics analysis, logical analysis, common lexical matching, and distributed vector representation with both common and legal expertise knowledge[1]. In [5], they build a system called "History Assistant" which extracts rulings from court opinions and retrieves relevant prior cases from a citator database by combining partial parsing techniques with domain knowledge and discourse analysis to extract information from the free text of court opinions. In [24], they develop a knowledge representation model for the intelligent retrieval of legal cases involving decomposing issues into sub-issues, and categorizing factors into pro-claimant, pro-responder and neutral factors. In [18], they overcome the problem of keyword-based search due to synonymy and ambivalence of words by developing an ontological framework to enhance the user's query and ensure efficient retrieval by enabling inferences based on domain knowledge. Other works related to building legal ontology are [4, 22, 23]. Aside of linguistics approaches which are expensive to develop because of the required expertise knowledge, other approaches utilizes the emerging effectiveness of neural networks for natural language processing with the pioneer method of mapping texts to continuous vector space[10, 14]. In [13], the authors measure legal document similarity considering structural information of the document including paragraphs, summary and utilizing various representation methods including lexical features: TF-IDF, and topic modeling, and distributed vector representational features: *word2vec*, and *doc2vec*.

The Competition on Legal Information Extraction/Entailment, for the first time in 2018, organized a legal case retrieval task [8]. The task was tackled in various approaches in both lexical matching and deep learning methods. The authors of the best system [21] have tackled the legal case retrieval task of COLIEE 2018 by developing a document encoding method using expert summary for training a phrase scoring model utilizing deep neural networks. In COLIEE 2019, we would like to utilize the resource of COLIEE 2018 and adopt the phrase scoring model pre-trained on COLIEE 2018 dataset to COLIEE 2019 dataset, and experiment the complementary benefits of using the encoding method with lexical matching for better retrieval system.

[1]http://www.uscourts.gov/statistics-reports/judicial-business-2017

In this paper, we describe our method for tackling the legal case retrieval task in Competition on Legal Information Extraction/Entailment (COLIEE), 2019 with the following main points:

- We showed the effectiveness of representing documents in continuous vector space in which the summary properties of the documents are embedded. The representation method is based on a phrase scoring model which is trained to assign high scores for phrases representing contexts that are similar to contexts found in the summary.
- The method is used to generate not only the latent features but also surface texts as predicted summary for lexical matching.
- We utilized the resource of COLIEE 2018 on which we train the document representation model.
- We extracted lexical features on different parts of queries and candidates.
- We have achieved the state-of-the-art result for the task on the benchmark of the competition.

2 METHOD

2.1 Overview of Legal Case Retrieval Task

The legal case retrieval task involves reading a new case q, and extracting supporting cases $c_1^*, c_2^*, ..., c_n^*$ for the decision of q from a given list of candidate cases. The candidate cases that support for the decision of a new case are called 'noticed cases'. The legal cases are sampled from a database of predominantly Federal Court of Canada case laws, provided by Compass Law.

In COLIEE 2018, when dealing with this task, Tran et al. [21] observed several obstacles. First, the candidate cases are ≈5.7K-token long in average (Table 1). This poses the problem of understanding the reason of selecting the cases as supporting cases. They, then, chose another approach which is comparing the summaries of each query and its candidate cases. They, however, found that the summary of the query is not necessarily lexically similar to the summary of the candidate cases. Moreover, some candidate cases do not have summary at all. They obtained the summary for each and every candidate cases, and furthermore, the summary should be comparable with the summary of the corresponding query. They came up with the idea of encoding the entire document into vector space embedding the properties of summarization, and called it encoded summarization. They realized the approach successfully for COLIEE 2018, and achieved the state of the art.

Table 1: Statistics of candidate case documents in COLIEE 2018 training data.

Property	Max	Avg.
#words/doc	85,551	5,690
#paragraphs/doc	1,117	43
#summary-words/doc	8,827	589

Table 2: Statistics of candidate case documents in COLIEE 2019 training data. (*) Only count documents having an expert summary.

Property	Max	Avg.
#words/doc	9,666	2,665
#paragraphs/doc	119	22
#summary-words/doc*	3,085	242

In COLIEE 2019, we observed the similar and different challenges. First, the candidate cases are ≈2.7K-token long in average (Table 2). The difficulty of reading too long texts still emerges. We may pursue the idea that using summary as the main source of information. However, the dataset of COLIEE 2019 is different from the one of COLIEE 2018. While in COLIEE 2018, most of the candidate cases have a summary, in COLIEE 2019, more than ≈47K in a total of 57K candidate cases are confirmed to have no summary (indicated with the note "This case is unedited, therefore contains no summary"). This means that summarization over candidate case requires additional effort so that we can compare a query's summary with a candidate's summary.

2.2 Encoded Summarization

The summary of a document contains the highlights of the document, which is an important factor in relevance analysis. The summary, however, is short and may not cover enough information. In this section, we describe a process of summarization for both the query and the candidates to include more points from the whole document. For this purpose, we train a phrase scoring model for identifying important phrases which discuss contexts similar to those in the summary.

In [20], the authors present a way of obtaining catchphrases of legal case documents via phrase scoring using deep neural networks. Catchphrases represent important legal points of a legal case document and are usually drafted by experts, thus play important roles for understanding the case. To generate catchphrases, they built a scoring model to estimate phrasal scores of a legal case document and used the phrasal scores to construct predicted catchphrases for a new legal case. The scoring model is trained given expert drafted catchphrases of the document and optimized with the objective that catchphrases are expected to have higher scores than other contents in the document. In other words, we can train such a scoring model given training documents with indicated important contents which are document summaries in our task. Inspired from the work's learning process, Tran et al. [21] proposed to obtain document embedding directed by document summary, applied successfully the method for COLIEE 2018, and achieved the state of the art.

The method, encoded summarization, is based on a phrase scoring model which is trained to assign high scores for phrases representing contexts that are similar to contexts found in the summary. Given a document, its the phrasal scores are estimated given its summary (extracted using indicators '*Summary:*' and '*Present:*') and paragraphs. The phrase scoring model is trained with the objective that summary contents are expected to have higher scores than paragraph contents. We trained the model on COLIEE 2018 dataset only in which most of the candidate cases have an expert summary. Fine-tuning of the phrase scoring model over COLIEE 2019 dataset is left for future work. After obtaining the trained model, the final document representation in continuous vector space is composed from latent representations of the model. The encoded summary represents the summarization nature of the document. From the phrase scoring model, we also follow the phrase extraction in [20] to generate text summary for lexical matching with summary of the query since most of the candidate cases in COLIEE 2019 dataset do not have a summary.

2.2.1 Phrase Scoring Model. We describe the phrase scoring model of [20] for this task.

Given a document, we denote $w_j^{s_i}$ as word j^{th} of sentence i^{th}. The features of an n-gram phrase $p_j = \{w_j, w_{j+1}, ..., w_{j+l-1}\}$ of a sentence are captured using convolutional neural layer as follows:

$$\mathbf{f}_{\mathbf{p_j}} = ReLU\left(\mathbf{W}^c \begin{bmatrix} \mathbf{v}(w_j) \\ \mathbf{v}(w_{j+1}) \\ ... \\ \mathbf{v}(w_{j+l-1}) \end{bmatrix}\right) \tag{1}$$

where, $\mathbf{v}(\cdot) : \mapsto \mathbb{R}^d$: word embedding vector lookup map, l: corresponding to the window size containing l contiguous words, $[\cdot] \in \mathbb{R}^{dl}$: concatenated embedding vector, $\mathbf{W}^c \in \mathbb{R}^{c \times dl}$: convolution kernel matrix with c filters, $\mathbf{f}_{p_j} \in \mathbb{R}^c$: phrase feature vector, $ReLU$: rectified linear unit activation.

Convolutional neural layers or convolutional neural networks have been applied widely in natural language processing, for example, text classification [6, 7, 9], machine translation [3], text pair modeling (question answering [19], and textual entailment [15]), and text summarization [11, 16]. The networks are designed to capture local contextual information by applying feature extraction over limited sub-regions of the input. With the assumption that words next to each other have relationship and contribute to the way of interpreting each of the words, convolutional neural networks could possibly capture phrasal writing phenomena in a given corpus. The assumption is also used in obtaining well-known word embeddings (*Google word2vec* [14], *Stanford GloVe* [17]).

Sentence features $\mathbf{f}_{s_i}$ are, then, captured by applying max pooling over the whole sentence.

$$\mathbf{f}_{s_i} = \text{max-pooling}_j(\mathbf{f}_{p_j^{s_i}}) \tag{2}$$

where max-pooling are operated over each dimension of vectors $\mathbf{f}_{p_j^{s_i}}$.

With the same max-pooling operation as above, we compute document features as:

$$\mathbf{f}_d = \text{max-pooling}_i(\mathbf{f}_{s_i}) \tag{3}$$

Finally, we apply a multilayer perceptron (MLP) with one hidden and one output layer

$$MLP(\mathbf{x}) = \text{sigmoid}(\mathbf{W}_2 \cdot \tanh(\mathbf{W}_1 \cdot \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \tag{4}$$

to compute the score of each phrase $p_j^{s_i}$ as

$$P\left(p_j^{s_i}, s_i, d\right) = MLP\left(\begin{bmatrix} \mathbf{f}_{p_j^{s_i}} \\ \mathbf{f}_{s_i} \\ \mathbf{f}_d \end{bmatrix}\right) \tag{5}$$

where the hidden layer computes the phrase representative features respecting to its local use (limited word window), its enclosing sentence, and its document. The phrase representative features are fed to the output layer to compute phrase score (ranging from 0.0 to 1.0).

We now describe how to train the phrase scoring model. The main objective is the trained model should assign summary phrases with higher scores than document phrases if the summary belongs to the document and otherwise, assign summary phrases with lower scores than document phrases if the summary does not belong to the document. We denote mean E and standard deviation std of phrase scores P for each document d in the following equations, which we will use to describe our objective as set of constraints, then formulate into loss function to be optimized.

$$E_c = E[P(p_c, c, d)] \text{ where } p_c \in c, c \in d \tag{6}$$

$$std_c = std[P(p_c, c, d)] \text{ where } p_c \in c, c \in d \tag{7}$$

$$E_s = E[P(p_s, s, d)] \text{ where } p_s \in s, s \in d \tag{8}$$

$$std_s = std[P(p_s, s, d)] \text{ where } p_s \in s, s \in d \tag{9}$$

$$E_{c,d'} = E[P(p_c, c, d')] \text{ where } p_c \in c, c \notin d' \tag{10}$$

Where p, c, s, d stand for phrase, summary sentence, document sentence, and the whole document respectively. $c \notin d'$ means c is not a summary of document d'.

The main objective is realized by comparing the mean scores of summary phrases and document phrases:

- **(o1)** The mean score of summary phrases is higher than the mean score of document phrases: $E_c > E_s$.
- **(o2)** The mean score of summary phrases is lower than document phrases when comparing a summary with a document that the summary does not belong to: $E_{c,d'} < E_{s'}$. This is the negative constraint as opposed to the constraint **o1**.

The above two constraints are straightforward as the positive and negative factors of the objective. However, the comparison of the mean values does not guarantee to obtain to good scoring model as the score boundaries are not considered yet.

- **(o3)** The maximum score of summary phrases is higher than the maximum score of document phrases. It is expected that there exists concise summary phrases which is typical and representative for the document but could not found in the document. Such summary phrases should get higher scores than document phrases. The estimation $E + std$ is used for representing max instead of hard max, whereby the constraint is realized as $(E_c + std_c) > (E_s + std_s)$.

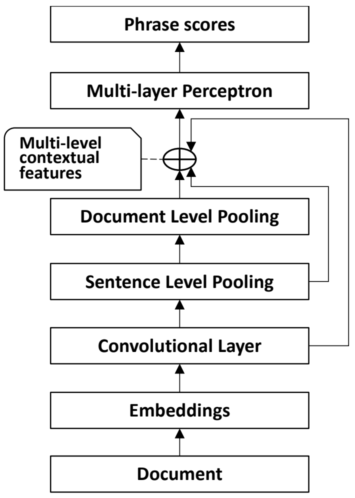

Figure 1: Phrase scoring model architecture.

(o4) The minimum score of summary phrases is higher than the mean score of document phrases. Once again, to emphasize the importance of summary phrases, all summary phrases should get higher score than the average score of document phrases. The estimation $E - std$ is used for representing min instead of hard min, whereby the constraint is realized as $(E_c - std_c) > E_s$.

The loss function, hence, is composed from the constraints **(o1-4)** as follows.

$$\begin{aligned}\mathfrak{L} = \sum_d \max(0, m - (a_1(E_c - E_s) \\ + a_2(\frac{1}{|\{d'\}|} \sum_{d' \neq d} E_{s'} - E_{c,d'}) \\ + b_1((E_c + std_c) - (E_s + std_s)) \\ + b_2((E_c - std_c) - E_s)))\end{aligned} \quad (11)$$

2.2.2 Document Vector Composition. We present our method of composing document vectors from the phrase scoring model.

First, given a document, we obtain its phrase scores and its internal representations: phrase level, sentence level and document level encodings.

Then, we compose the document vector as:

$$\mathbf{g}(d) = \frac{\sum_{i,j} P\left(p_j^{s_i}, s_i, d\right) \times \left[\mathbf{f}_d; \mathbf{f}_{s_i}; \mathbf{f}_{p_j^{s_i}}\right]}{\sum_{i,j} P\left(p_j^{s_i}, s_i, d\right)} \quad (12)$$

This composition resembles summarization where we weight the document internal representations by its summary. Thus, we call this composition encoded summarization.

2.2.3 Query-Candidate Relevance Vector. Each dimension in one document vector is obtained from composition of the same kernel representing one feature. By comparing each dimension independently, we can estimate the compatibility of a query and a candidate over the dimension. Thus, we compute the query-candidate relevance vector as the element-wise product of query vector and candidate vector.

First, we obtain query vector $\mathbf{g}(q)$ and candidate vector $\mathbf{g}(c)$ using the document vector composition in Equation 12. Then, we compute the relevance vector of query q and candidate c by the following element-wise product.

$$\mathbf{h}(q, c) = \mathbf{g}(q) \odot \mathbf{g}(c) \quad (13)$$

2.2.4 Generating Text Summary. In this phase, we generate a summary given a document by selecting and joining document phrases scored by the phrase scoring model. The process is as follows.

- Rank document phrases by they phrasal scores.
- Select phrases with scores from high to low.
- Join overlapping phrases into a longer phrase.
- Stop when the summary length exceeds length-threshold t.

The result summary is a list of phrases. The shortest phrases contains l words (l is the window size of the convolutional neural layer). The longest phrases are the sentences themselves.

2.3 Lexical Matching

We perform multi-aspect lexical matching where we compare a query case with its candidates via different views. We represent each query as 2 parts: summary and paragraphs, and each candidate as 3 parts: summary, lead sentences (of each paragraph), and paragraphs. This time, most of the candidates do not have a summary, thus, we generate a summary for each candidate by utilizing our encoded summarization method with the summary generation process described in Section 2.2.4.

To have more precise comparison between a query versus a candidate, we compute 6 matching options:

- Summary vs. Summary: Matching the summary of the query case with the summary of candidates.
- Summary vs. Lead-sentences: Matching the summary of the query case with the opening of each paragraph of candidates.
- Summary vs. Paragraphs: Matching the summary of the query case with the details of candidates.
- Paragraphs vs. Summary: Matching the details of the query case with the summary of candidates.
- Paragraphs vs. Lead-sentences: Matching the details of the query case with the opening of each paragraph of candidates.
- Paragraphs vs. Paragraphs: Matching the details of the query case with candidates.

We perform various kinds of text matching including: n-gram, skip-gram, subsequence matching formulas.

- N-gram matching: measuring n-gram overlapping between a query and a candidate case. We employ unigram and bigram models.
- Skip-bigram matching: measuring the co-occurrence of all word pairs in their sentence order. This allows the same noncontinuous word pairs could be found in both query and candidate.

- We also employ the unigram + skip-gram model which balances the unigram matching and skip-gram matching.
- Longest common subsequence: measuring the strictly ordered overlapping scattering over the texts. We employ 2 variants: standard version and distance weighted version. The distance weighted version favors subsequences with less distances among words.

For each matching formula, we compute the matching scores by 3 different factors:

- Recall: normalized by query, measuring the percentage of the query contents found in the candidate.
- Precision: normalized by candidate, measuring the percentage of the candidate contents found in the query.
- F-measure: harmony score of the previous two.

$$f\text{-}measure = \frac{2 \times precision \times recall}{precision + recall}$$

For the computation of lexical matching features, we used publicly available ROUGE library[2].

In total, we collect lexical features from 6 matching options, 6 matching formulas and 3 matching factors, which results in lexical feature space with 108 dimensions.

2.4 Learning to Rank Candidates

We formulate the task as ranking problem and devise the learning to ranking method to solve it using lexical matching features from Sections 2.3 and query-candidate relevance features from Section 2.2.

We utilize pair-wise ranking strategy: pairing each supporting case with an irrelevant case from the candidate list. We adopt Linear-SVM as the learning algorithm for solving the optimization problem. The final ranking model should assign high scores to supporting cases and low scores to non-noticed cases.

After obtaining the scored candidates as a ranked list, we proceed to select top k candidates as the predicted noticed cases.

3 EXPERIMENTS

3.1 Experimental Settings

For the phrase scoring model, we copied the model settings from [20], except the loss coefficients, as shown in Table 3. The loss coefficients were tuned using random search on COLIEE 2018 dataset. For word embeddings, we use the pre-trained GloVe[3] which was trained on data from "Common Crawl"[4], an open repository of web crawl data.

The phrase scoring model was trained on COLIEE 2018 dataset, and adopted to generate encoded summarization vectors for case documents, and text summaries for the candidate cases in COLIEE 2019 dataset. For generating the text summaries, the summary length threshold t (Section 2.2.4) is set to $t = 20\%$ document-length. The average length of summaries is $\approx$ 10% document-length for COLIEE 2018 dataset (Table 1), and $\approx$ 9% document-length for COLIEE 2019 dataset (Table 2). Thus, with a threshold $t = 20\%$ document-length, we could expect to cover potential information with good recall rate while keeping an acceptable summary length.

Table 3: Phrase scoring model parameters.

Parameter	Description
Embeddings (vector size d)	GloVe [17] $d = 300$
CNN filters c	300
CNN window size l	5
MLP hidden size	300
Optimizer	Adam[2]
Learning rate	0.0001
Gradient clipping max norm	5.0
Loss coefficients (a_1, a_2, b_1, b_2)	$(1.0, 1.7, 0.3, 0.7)$
Size of negative set $\lvert\{d'\}\rvert$	2
loss margin m	0.5

For text pre-processing, we use the default word and sentence tokenization from NLTK [12][5]. We use ROUGE library with Porter stemmer and stopword removal enabled[6].

We reported our system's validation results with the following metrics:

- MRR: Mean reciprocal rank. This metric measures the rank of the first correct answer or the first (true) supporting case given a query case. In other words, this measures how far users have to read to find a relevant case when looking from the top of the retrieved list.
- MAP: Mean average precision. Since we pursue the task with learning to rank method, we use this common ranking evaluation metric.
- Prec, Rec, F1: Precision, Recall, F-1 whose values are averaged by query. This is straightforward as we average the results of all folds in the leave-one-out validation.

For computing Prec, Rec, and F1, we proceed to perform top k selection from the ranked list output by the Linear-SVM Rank model. The selected value of k is 5, the average number of noticed cases in COLIEE 2019 dataset.

For presenting lexical features' impact analysis, we use a coding for representing subsets of the full lexical feature set. The coding is in the form of q-c, where

- q is a subset of query components including summary (s) and paragraphs (p),
- c is a subset of candidate components including paragraphs (p), lead sentences (l), and generated summary (e) (described in Section 2.2.4),
- each component in q is compared with each component in c.

For example, the lexical method sp-ple (q=sp, c=ple) means we perform all 6 matching options, and the lexical method s-p (q=s, c=p) means we only compare the summary of a query with the paragraphs of a candidate.

[2]https://github.com/andersjo/pyrouge
[3]https://nlp.stanford.edu/projects/glove/ Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors)
[4]http://commoncrawl.org/
[5]https://www.nltk.org/ version 3.3.0
[6]rouge parameters: -c 95 -2 -1 -U -r 1000 -n 2 -w 1.2 -a -d -m -s

Table 4: Validation results. The coding for lexical features is in the form of q-c, where q is a subset of query components including summary (s) and paragraphs (p), c is a subset of candidate components including paragraphs (p), lead sentences (l), and generated summary (e) (described in Section 2.2.4), and each component in q is compared with each component in c. For example, the lexical method sp-ple (q=sp, c=ple) means we perform all 6 matching options, and the lexical method s-p (q=s, c=p) means we only compare the summary of a query with the paragraphs of a candidate.

Model	MRR	MAP	Prec	Rec	F1
Lexical					
s-p	0.843	0.690	0.484	0.620	0.470
s-l	0.798	0.589	0.420	0.528	0.405
s-e	0.756	0.561	0.401	0.517	0.390
p-p	0.845	0.680	0.476	0.601	0.461
p-l	0.805	0.619	0.443	0.563	0.429
p-e	0.801	0.588	0.413	0.534	0.402
sp-p	0.871	0.712	0.490	0.635	0.480
sp-l	0.816	0.634	0.448	0.570	0.435
sp-e	0.793	0.602	0.429	0.553	0.416
sp-pl	0.857	0.713	0.493	0.639	0.483
sp-pe	0.864	0.709	0.485	0.633	0.476
sp-ple	0.859	0.715	0.495	0.641	0.485
ES	0.840	0.576	0.436	0.534	0.410
ES + sp-p	0.957	0.822	0.572	0.715	0.549
ES + sp-pl	*0.959*	*0.831*	**0.581**	**0.730**	**0.560**
ES + sp-pe	0.954	0.823	0.577	0.716	0.553
ES + sp-ple	**0.963**	**0.833**	*0.579*	*0.724*	*0.557*

3.2 Validation Results

The validation results are shown in Table 4. The best two are ES+sp-ple and ES+sp-pl. ES+sp-ple achieves the best MRR of 0.963 and the best MAP of 0.833. ES+sp-pl achieved the best Prec of 0.579, Rec of 0.724 and F1 of 0.557.

The validation results (Table 4) of lexical features with various combinations from the 6 matching options in Section 2.3 show that the combination of lexical matching options does have positive effect to improve the performance. sp-p, the combination of the two (s-p and p-p), achieves MAP of 0.712 and F1 of 0.480, which are higher than those of the two. The same effect appears on sp-l, sp-e, sp-pl, sp-ple. Even though with the exception of sp-pe: MAP and F1 are lower than sp-p, sp-ple is the best lexical combination, which wins over other lexical combinations over three of four evaluation metrics with MAP of 0.715, Prec of 0.495, Rec of 0.641, and F1 of 0.485.

The encoded summarization (ES) approach alone achieves MAP of 0.576 and F1 of 0.410, lower performance than the best lexical combination. The effect is different from the observation in [21] where the performance of encoded summarization is higher than lexical matching approach. Since the encoded summarization model is trained on only COLIEE 2018 dataset, some summary phenomena in COLIEE 2019 dataset may not be well captured.

The combination of encoded summarization and lexical features does improve performance significantly. In term of F1 score, the improvement is from +0.068 (ES+sp-pe) to +0.077 (ES+sp-pl). In term of MAP score, the improvement is from +0.108 (ES+sp-pe) to +0.118 (ES+sp-pl and ES+sp-ple). The improvement by the combination shows that, even though the encoded summarization does not perform well alone, it still provides useful information for identifying relevant cases. This effect is also observed in [21].

3.3 Submission Results

We submitted 3 runs to the competition: ES+sp-p, ES+sp-pl, and ES+sp-ple (Table 5). The best test performance is F1 of 0.5764 achieved by two models: ES+sp-pl and ES+sp-ple. Moreover, we also have achieved the best performance compared to other participants of COLIEE 2019 on the test set of the legal case retrieval task (Table 7).

Table 5: Submission results.

Model	Prec	Rec	F1
ES + sp-p	0.5934	0.5485	0.5701
ES + sp-pl	0.6000	0.5545	**0.5764**
ES + sp-ple	0.6000	0.5545	**0.5764**

Table 6: Difference in test submission of ES+sp-pl and ES+sp-ple.

	ES + sp-pl		ES + sp-ple	
Query	Candidate	Relevant?	Candidate	Relevant?
002	043	NO	181	NO
004	182	NO	067	NO
007	157	NO	023	NO
009	003	NO	107	NO
010	**002**	**NO**	**164**	**YES**
013	187	NO	132	NO
014	022	NO	020	NO
016	168	NO	115	NO
017	126	NO	073	NO
028	**113**	**YES**	**039**	**NO**
030	**116**	**YES**	**044**	**YES**
033	126	NO	028	NO
035	**001**	**YES**	**066**	**NO**
037	116	NO	188	NO
039	070	NO	095	NO
042	001	NO	032	NO
048	**163**	**NO**	**121**	**YES**
051	090	NO	175	NO
056	079	NO	101	NO

Even though ES+sp-pl and ES+sp-ple share the same test performance as shown in Table 5, the test submissions are different. Over 61 test queries, 19 queries are submitted differently for each of the two models. The noticeable differences are of queries 010, 028, 030, 035, and 048, where the performance changes. In the cases of queries 010 and 048, ES+sp-ple has better outputs, discards irrelevant candidates and selects relevant ones. In the cases of queries

Table 7: Participants' results. We submitted 3 runs to the competition: ES+sp-p, ES+sp-pl, and ES+sp-ple, corresponding to JNLP.task_1.p, 'JNLP.task_1.pl, and JNLP.task_1.ple, respectively. We achieved the best performance of 0.5764 F1 score on the test set of the legal case retrieval task.

Team	Run name	Prec	Rec	F1
CACJ	submit_task1_CACJ01	0.2119	0.5848	0.3110
CLArg	CLarg	0.9266	0.3061	0.4601
HUKB	task1.HUKB	0.7021	0.4000	0.5097
IITP	task1.IITPBM25	0.6256	0.3848	0.4765
IITP	task1.IITPd2v	0.4653	0.3455	0.3965
IITP	task1.IITPdocBM	0.6368	0.3879	0.4821
ILPS	BERT_Score_0.946	0.6810	0.4333	0.5296
ILPS	BERT_Score_0.96	0.8188	0.3424	0.4829
ILPS	BM25_Rank_6	0.4672	0.5182	0.4914
JNLP	*JNLP.task_1.p*	0.5934	0.5485	*0.5701*
JNLP	**JNLP.task_1.pl**	0.6000	0.5545	**0.5764**
JNLP	**JNLP.task_1.ple**	0.6000	0.5545	**0.5764**
UA	UA_0.52	0.3513	0.3364	0.3437
UA	UA_0.54	0.3639	0.3242	0.3429
UA	UA_0.57	0.3560	0.3333	0.3443

028 and 035, ES+sp-ple has worse outputs, discards relevant candidates and selects irrelevant ones. In the last case of query 030, ES+sp-pl and ES+sp-ple choose different but all relevant candidates. The two models differ in the addition of comparing a query's summary and paragraphs to a candidate's generated summary.

4 CONCLUSION

We have presented our method for legal case retrieval task in Competition on Legal Information Extraction/Entailment, 2019. We have applied the approach of combining lexical features and latent features embedding summary properties which we call encoded summarization. The combination of encoded summarization and lexical features does improve performance significantly. Besides, the inclusion of lexical matching with lead sentences of each paragraph and text summaries generated via the phrase scoring model shows positive effect in improving retrieval performance. For validation results, we achieved best MAP score of 0.833 and second best F1 score of 0.557 (the best is 0.560) with combination of encoded summarization with all of the described lexical features including comparing a query's summary and paragraphs to a candidate's paragraphs, lead sentences and summary generated via the phrase scoring model. We have showed that the phrase scoring model trained from COLIEE 2018 dataset can provide useful features for representing documents in COLIEE 2019 dataset.

There are several directions for improving the performance of legal case retrieval systems. One is that we can use the documents having a summary in COLIEE 2019 dataset for fine-tuning the phrase scoring model. Besides, the lexical matching has not yet considered the statistical information of terms in the corpus, which can be modeled by term frequency-inverse document frequency for example. Including such information may improve the matching by recognizing the statistically typical words for each document.

ACKNOWLEDGMENTS

This work was supported by JST CREST Grant Number JPMJCR1513, Japan.

REFERENCES

[1] Trevor Bench-Capon, Michał Araszkiewicz, Kevin Ashley, Katie Atkinson, Floris Bex, Filipe Borges, Daniele Bourcier, Paul Bourgine, Jack G Conrad, Enrico Francesconi, et al. 2012. A history of AI and Law in 50 papers: 25 years of the international conference on AI and Law. *Artificial Intelligence and Law* 20, 3 (2012), 215–319.

[2] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[3] Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A Convolutional Encoder Model for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 123–135.

[4] Anatoly P Getman and Volodymyr V Karasiuk. 2014. A crowdsourcing approach to building a legal ontology from text. *Artificial intelligence and law* 22, 3 (2014), 313–335.

[5] Peter Jackson, Khalid Al-Kofahi, Alex Tyrrell, and Arun Vachher. 2003. Information extraction from case law and retrieval of prior cases. *Artificial Intelligence* 150, 1-2 (2003), 239–290.

[6] Rie Johnson and Tong Zhang. 2015. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, 103–112. http://www.aclweb.org/anthology/N15-1011

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 655–665. http://www.aclweb.org/anthology/P14-1062

[8] Yoshinobu Kano, Mi-Young Kim, Masaharu Yoshioka, Yao Lu, Juliano Rabelo, Naoki Kiyota, Randy Goebel, and Ken Satoh. 2018. COLIEE-2018: Evaluation of the Competition on Legal Information Extraction and Entailment. Twelfth International Workshop on Juris-informatics (JURISIN), COLIEE.

[9] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1746–1751. http://www.aclweb.org/anthology/D14-1181

[10] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.

[11] Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling Length in Abstractive Summarization Using a Convolutional Neural Network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4110–4119.

[12] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1 (ETMTNLP '02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 63–70. https://doi.org/10.3115/1118108.1118117

[13] Arpan Mandal, Raktim Chaki, Sarbajit Saha, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh. 2017. Measuring Similarity among Legal Court Case Documents. In *Proceedings of the 10th Annual ACM India Compute Conference on ZZZ*. ACM, 1–9.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[15] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *The 54th Annual Meeting of the Association for Computational Linguistics*. 130.

[16] Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1797–1807.

[17] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[18] M. Saravanan, B. Ravindran, and S. Raman. 2009. Improving legal information retrieval using an ontological framework. *Artificial Intelligence and Law* 17, 2 (01 Jun 2009), 101–124. https://doi.org/10.1007/s10506-009-9075-y

[19] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 373–382.

[20] Vu D. Tran, Minh L. Nguyen, and Ken Satoh. 2018. Automatic Catchphrase Extraction from Legal Case Documents via Scoring using Deep Neural Networks. In *Workshop on MIning and REasoning with Legal texts*. https://www.researchgate.net/publication/327645790_Automatic_Catchphrase_Extraction_from_Legal_Case_Documents_via_Scoring_using_Deep_Neural_Networks

[21] Vu Duc Tran, Son Truong Nguyen, and Minh Le Nguyen. 2018. JNLP Group: Legal Information Retrieval with Summary and Logical Structure Analysis. Twelfth International Workshop on Juris-informatics (JURISIN), COLIEE.

[22] Adam Wyner. 2008. An ontology in OWL for legal case-based reasoning. *Artificial Intelligence and Law* 16, 4 (2008), 361.

[23] Adam Wyner and Rinke Hoekstra. 2012. A legal case OWL ontology with an instantiation of Popov v. Hayashi. *Artificial Intelligence and Law* 20, 1 (2012), 83–107.

[24] Yiming Zeng, Ruili Wang, John Zeleznikow, and Elizabeth Kemp. 2005. Knowledge Representation for the Intelligent Legal Case Retrieval. In *Knowledge-Based Intelligent Information and Engineering Systems*, Rajiv Khosla, Robert J. Howlett, and Lakhmi C. Jain (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 339–345.