Building Vector Representation for Modeling Similarity of Structured Documents

TRAN Duc Vu

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

Building Vector Representation for Modeling Similarity of Structured Documents

TRAN Duc Vu

Supervisor : Associate Professor NGUYEN Le Minh

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science
September, 2019

Abstract

Representing texts into vector space is revolutionary to Natural Language Processing, which brings the ability to apply deep learning, the very popular and very powerful machine learning technique, on texts which was previously infeasible. Remarkable works have been done on this topic, namely, *word2vec*, *GloVe*, *doc2vec*, *deepwalk*, and dependency-based word embeddings. Theirs models represent texts into vector space which enabling the computability or compatibility of texts with well-known deep learning models which were previously only applicable for digital data such as images, speeches, etc. *word2vec*/*GloVe* models context distribution of each word via the concept of surround context when a word is used in various text sequences. *dependency-based* word embedding is another work that builds the surround context by traversing through the dependency tree of a sentence, hence, takes care about positionally distant dependencies.

While it is convenient to have word vectors, it is usually not straightforward to compose a document vector from its word vectors. Based on specific tasks, document vectors are learned with certain algorithms or deep learning architecture specialized for the said tasks. *doc2vec* leverages this problem by introducing document-context presence into each word-context and learning the vector representations altogether. However, the implementation does not cover internal structures of the document. Besides, *deepwalk* is another work on context-based vector representation by learning node vectors of a given graph. Similar to *dependency-based* word embedding, *deepwalk* focuses on building the surround contexts of each node by performing random walks through the node.

Document structures can contain relationships including (but not limited to) hierarchy (sections, paragraphs, sentences), discourse (relationships between text-pairs such as agreement, contradiction, or equivalence), and cross-references, though, the previous works only cover a part or none of structural properties of documents.

We aim to build document embedding frameworks that can capture the dependencies within a document in multiple levels of hierarchy: words, sentences, and so on. We develop several methods for capturing those dependencies including context expansion on document hierarchy, *pq*-gram on dependency trees, rhetorical structure, and multi-level contextual features for encoded summarization.

We applied our methods successfully on tasks related to sentence pair modeling and information retrieval.

Keywords: **Deep Learning**, **Representation Learning**, **Information Retrieval**, **Legal Domain**, **Case Law**.

Acknowledgment

To whom I owe so much, I would like to express my sincerest gratefulness.

I would like to convey my faithful gratitude to my supervisor, Associate Professor Minh Le Nguyen, for his guidance and trust. I've been learning from him in every moment.

I would like to thank Professor Satoshi Tojo, Associate Professor Kiyoaki Shirai, Professor Ken Satoh, Professor Akira Shimazu, and Associate Professor Shinobu Hasegawa for their constructive questions and suggestions for the improvement of my research.

I am grateful to Associate Professor Cuong Pham who showed me the science world and encourages me to become a scientist.

I am glad that I have been working with my excellent colleagues. Their innovative discussions and dedicated collaboration are for my inspiration.

It is a great honor for me to work under JST CREST Grant Number JPMJCR1513, Japan for the chance to experience intensive research in various areas, which widens my research view and inspires me a lot.

I would like to thank JAIST for providing us fantastic research condition and infrastructure during the time I'm doing this dissertation. The excellent research environment here allows me to conduct my research effectively.

Thank you, my friends who are always at my side and cheer me up.

I want to send my love to my father, my mother and my sister as their love for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Direction

Representing texts into vector space is revolutionary to Natural Language Processing, which brings the ability to apply deep learning, the very popular and very powerful machine learning technique, on texts which was previously infeasible. Remarkable works have been done on this topic, namely, *word2vec*, *GloVe*, *doc2vec*, *deepwalk*, and *dependency-tree*-based word embeddings. Theirs models represent texts into vector space which enabling the computability or compatibility of texts with well-known deep learning models which were previously only applicable for digital data such as images, speeches, and so on.

While it is convenient to have word vectors, it is usually not straightforward to compose a document vector from its word vectors. Based on specific tasks, document vectors are learned with certain algorithms or deep learning architecture specialized for the said tasks. *doc2vec* leverages this problem by introducing document-context presence into each word-context and learning the vector representations altogether. However, the implementation does not cover internal structures of the document. Besides, *deepwalk* is another work on context-based vector representation by learning node vectors of a given graph. Similar to *dependency-tree*-based word embeddings, *deepwalk* focuses on building the surround contexts of each node by performing random walks through the nodes of a dependency tree.

We are interested in two parts of representation learning: input information and target embedding. For example, *word2vec* learns word representation with the input is contextual information bounded by a fixed window size and the target is to predict words in a given context. In our study, input contains document structure information such as document hierarchy, discourse relations, and references on top of the common sequential structure representing temporal nature of comprehending texts. The target of the representation learning is, then, defined to direct the representation to certain vector space embedding the semantics/properties of the target. In our study, the target can be either common, for example context prediction in *word2vec*, or task specific, for example text-pair modeling, and summarization. The representation can be learned and used directly in the originated task or be adapted to other tasks, for example, *doc2vec* is learned to predict document contexts and can be applied to measure document similarity and predict document relevance.

1.1.1 Structural Encoding

Document structures can contain relationships including (but not limited to) hierarchy (sections, paragraphs, sentences), discourse (relationships between text-pairs such as agreement, contradiction, or equivalence), and cross-references, though, the previous works only cover a part or none of structural properties of documents. We aim to build document embedding frameworks that can capture the dependencies within a document in multiple levels of hierarchy: words, sentences, and so on. We develop several methods for capturing those dependencies including context expansion on document hierarchy, *pq*-gram on dependency trees, and multi-level contextual features for encoded summarization.

Sentence encoding is one important step in common deep neural architecture based text modeling, which maps a lexical sentence into its embedding in computational real-value space. The sentence embeddings, based on tasks or learning problems, contain useful information for identifying themselves. Sentence embeddings are usually built from their content word embeddings with certain composition, for instance, sequential composition and tree composition.

Sequential composition captures the temporal structure of the sentence, the natural way of speaking/writing/listening/reading a sentence. This composition, however, does not explicitly capture syntactic constraints of how to compose the meaning of the sentence, thus minor sequence modification may lead to major shift in sentence embedding.

Tree composition captures explicit word/phrase relationship, for instance, syntactic trees, and dependency trees. The composition contains high level semantic dependency which is not trivial to neural networks even with the ability of automatically feature learning, where the unsupervised parsers are nowhere near supervised parsers [1, 2]. Hence, tree composition provides more information of how to interpret the sentence to learning models which do not know how to do so from initialization.

In unsupervised learning fashion, we present a method to represent a given document's paragraphs, sentences, words and itself into a vector space where contextual similarity can be measured. The method learns the vector representing a given text (documents, paragraphs, sentences, words) with its context expanded from its child contexts and its references. For example, the context of a paragraph is expanded to include the contexts of its sentences. Similarity can be measured at different context levels, and then supports text comparison at variant scope. As a case study, we conducted experiments on COLIEE 2017 dataset with external large text corpus. The results showed the effect of context expansion and common knowledge inclusion to retrieval. Since this is an unsupervised learning method, performance still has room for improvement though current result is not yet competitive to the state of the art.

In supervised learning fashion, we present our sentence encoding approach that uses local contexts constructed from *pq*-gram representations of a sentence's dependency tree. The context localization scope can be adjusted through parameters p, the dependency depth, and q, the dependency width, which allows controlling context-sensitivity. We show competitive results of using our sentence encoding approach for sentence-pair modeling tasks.

Aside from the composition approach, we can further build an intermediate composition step: local contexts. Local contexts are the surrounding words of a target word, where the meaning of the target word can be made clearer or less ambiguous. In other words, the composition process disambiguates each word before composing it

into sentence embeddings. The local contexts can be formed by extracting n-grams (in sequential composition), or head-children subtrees (tree composition). Besides, a sentence with a chosen composition structure, can be decomposed into a set of local contexts, which results in more relaxed representation which is less sensitive to word change and may benefit the task regarding similarity/relatedness.

We propose to represent a sentence as pq-grams of its dependency tree. This represents sentence local contexts with parameters p, the dependency depth, and q, the dependency width. Our approach utilizes tree composition but not with head-children fashion. In the head-children fashion, local contexts contain one head (target word) and all its children, whereas, our approach produces local contexts with ancestors of the target word and only a limited number of its children, described as parameters (p, q).

We apply the encoding approach to sentence-pair modeling tasks including recognizing textual entailment and measuring relatedness. We adapt convolutional neural networks (CNNs), which are successfully used in text modeling [3, 4, 5], to encode pq-gram representation into latent space. Then, we apply various sentence-pair encoding composition algorithms, namely, basic pooling, global comparison, local comparison with pq-alignment. The inclusion of multi-pq alignment, where we compute alignment of pq-gram encodings from different (p, q) values, shows good results in textual entailment and text relatedness tasks.

Sequential and tree composition approaches are used interchangeably or combined together in sentence encoding. In sequential composition approach , common architecture are based on long short-term memories (LSTMs) [6], and CNNs [3, 4]. LSTMs process a sentence from the beginning to the end where each word is encoded from its initial embedding and current temporal state composed from the previous words. Bidirectional-LSTMs (biLSTMs) process the sentence from both directions, then, produces richer representation. CNNs process a sentence by applying their shared filters to each n-gram region. In tree composition approach, LSTMs and CNNs are designed to operate over trees. Tree-LSTMs process a sentence's tree from leafs to root [7, 6, 8]. Tree-CNNs process a sentence's tree by applying their shared filters to each sub-tree consisting of one head word and its children [5].

We approach sentence encoding in a way that allows capturing clear semantic of the local contexts/regions of a sentence. Our approach benefits the tasks, for example, sentence-pair modeling tasks such as textual entailment and text relatedness, whose focus is on some parts of the sentences with various degrees of reception. One way that we can represent the reception degree is using CNNs with multiple window sizes which capture local context at multiple levels of granularity. The CNNs on the sequential representation of a sentence, however, also capture n-gram regions with unclear semantic. We sought out the method to not only capture local contexts but also capture regions with clear semantic. We, then, look at the dependency tree of a sentence, which represents the syntactic/semantic relations of each word-pair. Besides, the sub-trees (which are also dependency trees) of the dependency tree represent a clear semantic. Furthermore, a node is added to or removed from a dependency tree, the specificness of the semantic represented by that tree is, then, increased/reduced. This is where we adopt pq-gram model as a method for representing local contexts with clear semantic from dependency trees and multiple pq values for multiple granularity. We then use CNNs to encode pq-grams into latent space where pq-grams can be compared and aggregated to later provide sentence-pair relationship for tackling the sentence-pair

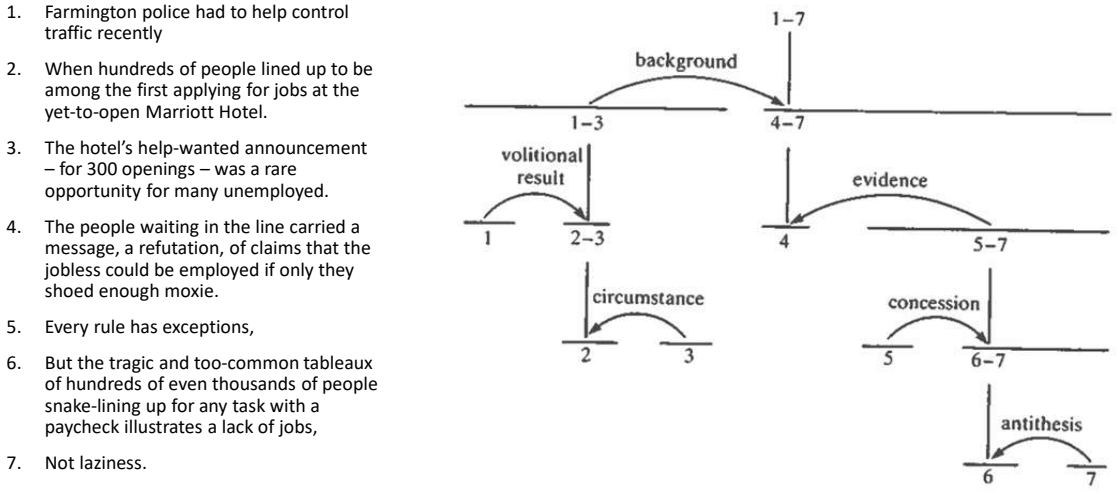


Figure 1.1.1: An example of parsing a text into rhetorical structure, copied from [9].

modeling tasks.

1.1.2 Rhetorical Information in Legal Case Documents

While sentence encoding relates to capturing internal sentence information, discourse analysis relates to capturing sentence inter-relationship, a higher level in the document hierarchy. On one hand, discourse analysis has high-impact applications in Natural Language Processing, for instances, text summarization, sentiment analysis, question answering. The output structures of the analysis contain high-level relationship of between discourses and so provides valuable information to such the tasks. On the other hand, deep learning has been shown effective towards Natural Language Processing tasks including discourse analysis. Recent studies reveal challenging problems regarding text-level discourse parsing, implicit discourse relations, etc. Further investigation on linguistic features combined with auto-generated features from appropriate deep learning architectures may help improve the efficiency of discourse paring models. The discourse structure can be used to compose the graph structure of a document and then benefits the learning of the encoded representation. Moreover, the applicability of deep learning to discourse parsing can help the learning of the encoded representation by providing not only the final discourse structure but also the parameters of the parsing model.

We follow the idea of rhetorical structure theory [9]. The ultimate goal is to build a system for automatic discourse structure parsing given a text block, which will result in a rhetorical structure graph illustrated in Fig. 1.1.1. We approach this challenge in two steps. The first step is to recognizing the potential rhetorical role or status of each sentence in a text block. The second step is to link the sentences with the known potential rhetorical role into the rhetorical structure graph. In this thesis, we tackle the first step by constructing a sentence rhetorical role recognition system, then, we further incorporate this system into a legal case retrieval system. Hachey et al. [10] describe a task of determining the rhetorical status of each sentence in a given document from a corpus of judgments of the UK House of Lords. The corpus is annotated with rhetorical statuses described in Table 3.1.1 with 7 types: FACT, PROCEEDINGS, BACKGROUND, FRAMING, DISPOSAL, TEXTUAL, and OTHER. Success-

Table 1.1.1: Example of rhetorical status annotated sentences in the corpus used by Hachey et al.

Rhetorical Status	**Sentence**
TEXTUAL	LORD NICHOLLS OF BIRKENHEAD
TEXTUAL	My Lords ,
...	...
FRAMING	This has long been axiomatic in this area of the law .
FRAMING	The matters the court may take into account are bounded only by the need for them to be relevant , that is , they must be such that , to a greater or lesser extent , they will assist the court in deciding which course is in the child 's best interests .
DISPOSAL	I can see no reason of legal policy why , in principle , any other limitation should be placed on the matters the judge may take into account when making this decision .
FRAMING	If authority is needed for this conclusion I need refer only to the wide , all embracing language of Lord MacDermott in J v C [1970] AC 668 , 710 - 711 .
BACKGROUND	Section 1 of the Guardianship of Infants Act 1925 required the court , in proceedings where the upbringing of an infant was in question , to regard the welfare of the infant ' as the first and paramount consideration ' .
...	...

fully identifying those statuses can help automatic information processing systems to comprehend or organize information in a court document more effectively. As shown in Table 1.1.1, with the rhetorical statuses, a system can identify the main statement in this segment is the DISPOSAL sentence supported by the surrounding FRAMING and BACKGROUND sentences.

1.1.3 Encoded Summarization & Legal Case Retrieval

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, 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]).

We study the legal case retrieval task which involves reading a new case, and then extracting cases supporting the decision of the new case. A case document contains a large volume of contents as the case may last days or even years. This one problem

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

challenges the construction of an effective automatic legal case retrieval system. One approach is to identify the gist of the documents, specifically, catchphrases for legal case documents. "Catchphrases have an indicative function rather than informative, they present all the legal point considered instead that just summarizing the key points of a decision" [11]. Catchphrases give a quick impression on what the case is about: "the function of catchwords is to give a summary classification of the matters dealt with in a case. [...] Their purpose is to tell the researcher whether there is likely to be anything in the case relevant to the research topic" [12]. On one hand, catchphrases help lawyers/researchers quickly grasp the points of a case, without having to read the entire document, which saves significant time and effort for finding/studying relevant cases. On the other hand, catchphrases help improves the performance of automatic case retrieval systems.

Despite of the benefits, catchphrases are not always available in legal case documents, and are drafted by legal experts, which requires huge efforts when considering the enormous number of legal case documents. It is, therefore, crucial to build automatic catchphrase generation systems for both old documents not having drafted catchphrases and new documents.

Approaches for generating catchphrases are based on phrase scoring derived from common model for retrieval: lexical matching with term frequency-inverse document frequency [13, 11, 14]. The approaches are bounded by the limit of lexical matching, and corpus-wide statistical information. The limit of lexical matching can be lifted by moving to distributed vector space, for instance, distributed word embeddings in which common models are *Word2Vec* [15] and *GloVe* [16]. Corpus-wide statistical information has limit capability to identify catchphrases which are not really specific to some document but commonly used in several others.

We present our work on developing a legal case summarization system and on top of its core component - phrase scoring framework, building a legal case retrieval system.

First, we build a learning model to extract catchphrases for new documents with the knowledge from previously seen documents and the expert drafted catchphrases thereof. Our system utilizes deep neural networks which have been widely used in natural language processing [17] to learn the direct relationship between gold catchphrases and document phrases. This results in our phrase scoring framework which is used to identify important phrases from a given legal case document.

On top of the phrase scoring framework, we develop our legal case document representation method which summarizes the document into continuous vector space. The representation is used for constructing case relevance ranking model, the core component of the retrieval system.

We also explore the benefits of employing various types of similarity measurement belonging to lexical similarity (keyword matching) and semantic similarity (meaning matching).

On one hand, the lexical similarity and semantic similarity differ from each other and can potentially complement each other as well. The lexical similarity is obtained with approaches where the texts are compared by the direct surface forms with probably some transformations such as stemming, lemmatization, stopword removal, etc. High lexical similarity can present high matching, but low lexical similarity does not say much.

On the other hand, semantic similarity can provide the measurement where the surface forms are mismatched, for example, by paraphrasing. Semantic similarity can

be learned in unsupervised fashion where common approaches are using statistical methods and benefits from huge available corpora (e.g. Wikipedia, GoogleNews, etc.) [18, 19, 15, 16]. Those methods treat a document as bag/sequence of words equally. Other information in the documents such as important words or phrases, or the document hierarchy when considered may provide significant information.

1.2 Contributions

The main contributions of this dissertation are:

- A study of using structural information for encoding structured documents into vector spaces. The study shows the effectiveness of exploiting the present structures of a document for obtaining useful information for constructing its representation. One of the most important structures is document hierarchy which we base on through out the dissertation.

- Encoded summarization: summary oriented document encoding which can be used for document similarity measure. The developed method represents a document into a vector space where the summary properties of the document is embedded in the way such that key contents of the document make significant impact on locating the document in the vector space.

- Results of various document encoding methods for building legal case retrieval systems. We have achieved potential results with our approach for building the systems. Compared to methods on a related task, we currently achieve state-of-the-art performance. The top performance is obtained by one of our systems that combines several aspects to model query case-candidate case relationship including lexical matching from coarse-grain to fine-grain, and encoded summarization with structural encoding, and rhetorical information.

1.3 Dissertation Outline

We have introduced our research direction in this Chapter. In the following chapters, we dwell into the details of the implementations of our methods. In Chapter 2, we present our approach for encoding structure information of a document. There, we show our study of using document hierarchy, sentence internal dependency and rhetorical analysis with deep neural networks. In Chapter 4, we present our approach for encoding the gist of a document into a vector space. In Chapter 5, we present the application of our document encoding methods for building legal case retrieval systems. Finally, Chapter 6 concludes with the summary of our findings throughout the dissertation and potential future directions of our research.

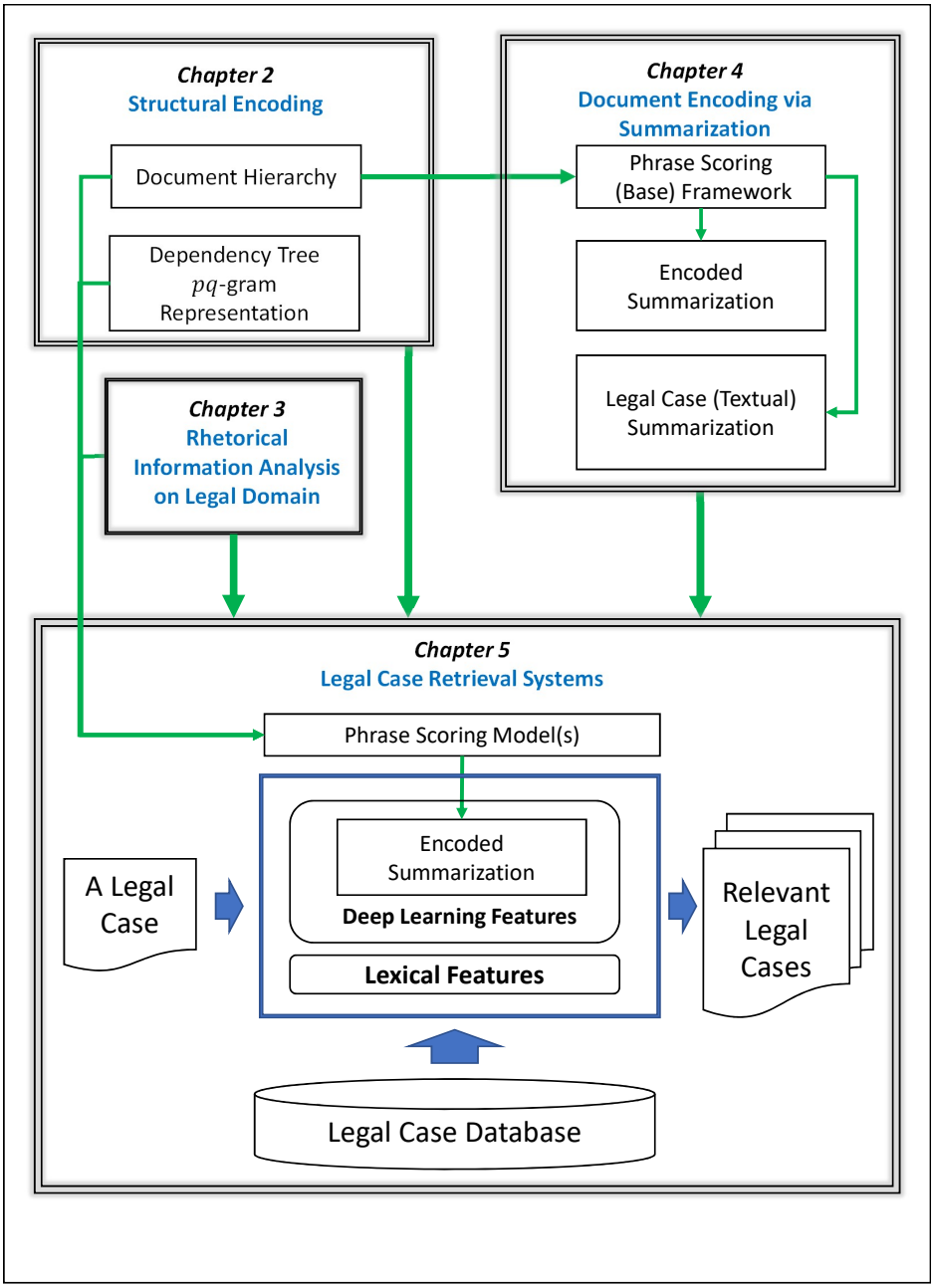


Figure 1.3.1: Dissertation Outline

Chapter 2

Structural Encoding

2.1 Introduction

Structural encoding refers to the approach of explicitly accounting the present structures of input texts to provide vector encoding of such texts. In this chapter, we present our study on the benefits of using structural information namely document hierarchy, sentence dependency tree.

2.2 Similarity with Document Components Vector Representations by Context Expansion from Document Structures

We present a method to represent a given document's paragraphs, sentences, words and itself into a vector space where contextual similarity can be measured. The method learns the vector representing a given text (documents, paragraphs, sentences, words) with its context expanded from its child contexts and its references. For example, the context of a paragraph is expanded to include the contexts of its sentences. Similarity can be measured at different context levels, and then supports text comparison at variant scope. As a case study, we conducted experiments on COLIEE 2017 dataset with external large text corpus. The results showed the effect of context expansion and common knowledge inclusion to retrieval. Since this is an unsupervised learning method, performance still has room for improvement though current result is not yet competitive to the state of the art.

2.2.1 Context Expansion from Document Structures

We present a method to represent a given document's paragraphs, sentences, words and itself into a vector space where contextual similarity can be measured. The method learns the vector representing a given text (documents, paragraphs, sentences, words) with its context expanded from its child contexts and its references. For example, the context of a paragraph is expanded to include the contexts of its sentences. Similarity can be measured at different context levels, and then supports text comparison at variant scope. As a case study, we conducted experiments on COLIEE 2017 dataset with external large text corpus. The results showed the effect of context expansion and common knowledge inclusion to retrieval. Since this is an unsupervised learning

method, performance still has room for improvement though current result is not yet competitive to the state of the art.

Document Similarity is a problem involving the task of measuring the similarity of a given pair of documents. Similarity measurement can be lexical similarity (keyword matching), semantic similarity (meaning matching). Document Similarity can be used as an important factor to determine the relevance of such pair of documents.

Similarity can be learned in supervised or unsupervised fashions. In supervised fashion, training data is required. In some special domains where labeled data is limited and expensive to make, trained models suffer from over-fitting, then couldn't learn the regularity of a given task from such data. In unsupervised fashion, common approaches are using statistical methods and benefits from huge available copora (e.g. Wikipedia, GoogleNews, etc.) [18, 19, 15, 16]. While the structural information of documents can be available, those methods have not yet take full advantage of the information.

We propose an approach to learn vector representations of a document's components (words, sentences, paragraphs) so as to measure similarity, with the following main points:

- Presenting a method to learn vector representations for document components which are hierarchically structured and contains cross-references among the components.
- Unsupervised method.
- Including common knowledge from general text into the learning procedure.
- Showing experimental results in legal information retrieval task on COLIEE 2017 dataset which has the above properties.

Representing texts into vector space is revolutionary to Natural Language Processing, which brings the ability to apply deep learning, the very popular and very powerful machine learning technique, on texts which was previously infeasible. Remarkable works have been done on this topic, namely, *word2vec* [15], *GloVe* [16], *doc2vec* [18], *deepwalk* [20], and *dependency-based word embeddings* [19]. Theirs models represent texts into vector space which enabling the computability or compatibility of texts with well-known deep learning models which were previously only applicable for digital data such as images, speeches, etc. *word2vec/GloVe* models context distribution of each word via the concept of surround context when a word is used in various text sequences. *dependency-based word embeddings* is another work that builds the surround context by traversing through the dependency tree of a sentence, hence, takes care about positionally distant dependencies.

While it is convenient to have word vectors, it is usually not straight forward to compose a sentence vector from its word vectors. Based on specific tasks, sentence vectors are learned with certain algorithms or deep learning architecture specialized for the said tasks. *doc2vec* leverages this problem by introducing document-context presence into each word-context and learning the vector representations altogether. However, the implementation does not cover internal structures of the document. Besides, *deepwalk* is another work on context-based vector representation by learning node vectors of a given graph. Similar to *dependency-based word embeddings*, *deepwalk* focuses on

building the surround contexts of each node by performing random walks through the node.

Document structures can contain relationships including (but not limited to) hierarchy (sections, paragraphs, sentences), discourse (relationships between text-pairs such as agreement, contradiction, or equivalence), and cross-references, though, the previous works only cover a part or none of structural properties of documents.

Document hierarchy is a way of organizing document in a structured format where multiple level of text components are presented hierarchically, for instances, from lower to higher levels: sentences, paragraphs, articles, subsections, sections, and so on (Figure 2.2.1). In the document hierarchy, the higher level presents the broader scope with more general context; the lower level presents the narrower scope with more specific context. Especially, in legal text, this presents the scopes of law.

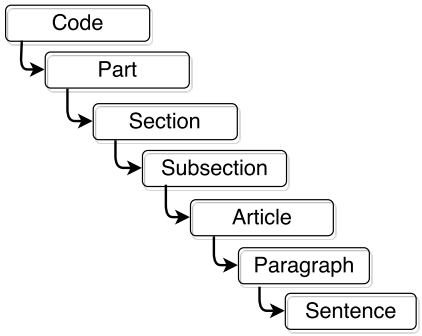


Figure 2.2.1: Hierarchy in Japanese Civil Code

Cross-referencing is a method with which a text refers to some text by placing a named entity pointing to that referred text. Instead of redefining terms or including repeated texts, cross-references make documents highly organized. This creates links between different locations inside a document and connects contexts which are on different branches in the hierarchy. However, this causes methods that assumes non-dependency when segmenting texts to suffer from missing information.

2.2.1.1 Context

The meaning or sense of words can only be inferred when using in certain context. Capturing or extracting contexts from texts is significant to learning word representations, and moreover, text representations.

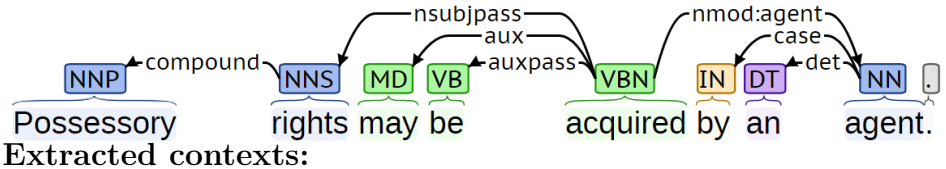
2.2.1.2 Context Extraction

To extract contexts from an input sentence, the system travels from root to each leaf of the sentence's dependency tree (Fig. 2.2.3). It results in the number of extracted contexts being the number of root-leaf paths. Sincere the root repeatedly appears in

> **(Rescission of Ruling for Commencement of Guardianship)**
> **Article 10**
> When the cause set forth in **Article 7** ceases to exist ...

Figure 2.2.2: An example of cross-references in Japanese Civil Code where Article 10 refers to Article 7.

(Possession by Agents)
Article 181
Possessory rights may be acquired by an agent.



Extracted contexts:

- acquired → may
- acquired → be
- acquired → rights → possessory
- acquired → agent → by
- acquired → agent → an

Figure 2.2.3: Contexts extracted from a dependency graph of a sentence.

all extracted contexts, it impacts significantly to the representation learning of these contexts. The idea of using dependency tree for context extraction is inspired from [19], where we relax dependency types to lessen the impact of strict syntactic and focus more on term presence. Dependency trees can capture positionally distant dependencies without introducing probably irrelevant contexts. Especially in a complex legal sentence, two terms having quite a long distance can be in direct dependency which is presented in the sentence's dependency tree.

2.2.1.3 Context Expansion

Context expansion is the process of extracting contexts for high level components, for instances, sentences, paragraphs, subsections, sections, etc. With document hierarchy, a higher level component has its contexts expanded by including its child components' contexts (Fig. 2.2.4). With cross-references, the component has its contexts expanded by including the referred components' contexts (Fig. 2.2.5).

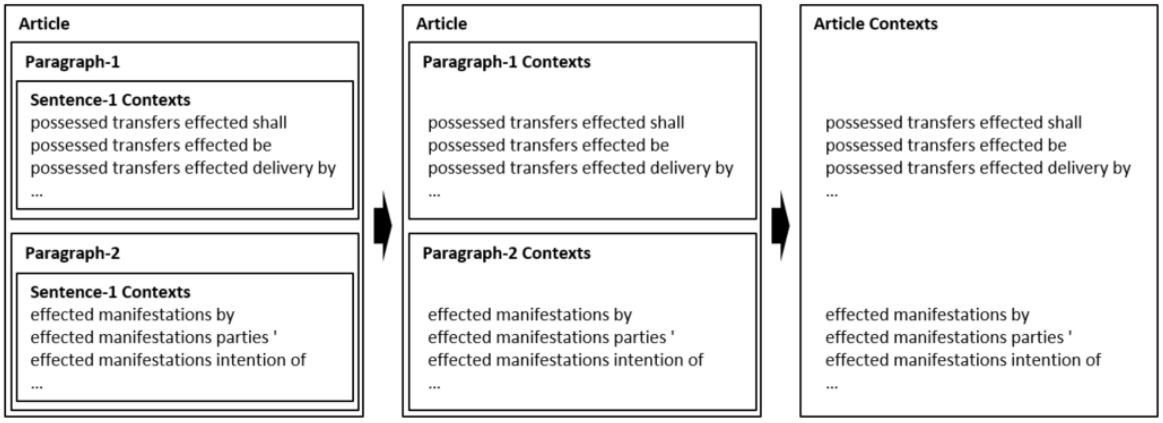


Figure 2.2.4: Context expansion with document hierarchy. Higher level components' context collections are expanded to included lower level components' contexts.

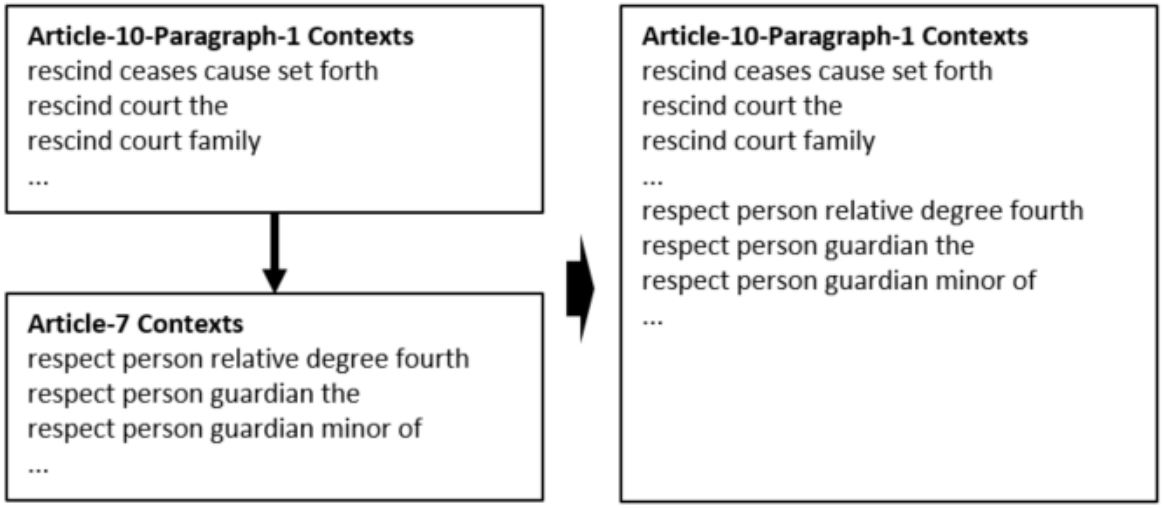


Figure 2.2.5: Context expansion with cross-references. Article 10 Paragraph 1 context collection is expanded to include referred Articled 7 contexts.

2.2.2 Document Components Vector Representations

We adopt Paragraph Vector - A Distributed Memory Model (PV-DM) [18], also known as *doc2vec* (Figure 2.2.6), for training vector representations for document components. From the extracted contexts in Section 4.1, sub-sampled contexts are generated with a specified window size (the number of words in a sub-sample context) which include the document component identity and sampled words. The vectors are trained in order to maximizing the probability of a word appearing in the correct context and minimizing the probability in the wrong context.

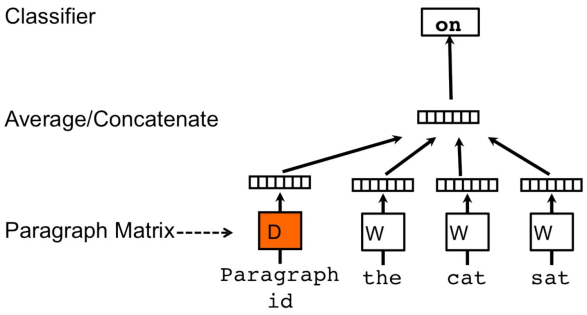


Figure 2.2.6: *doc2vec* model architecture

2.2.3 Experiments on Statute Law Retrieval Task

2.2.3.1 Dataset

In accordance with measure the effectiveness of our method, we conducted experiments on COLIEE 2017 dataset [21], a dataset for legal information retrieval task. The task is that given a juridical question, retrieve relevant civil articles from about 1,000 articles in Japanese Civil Code. The training set contains 580 examples including questions with their relevant articles.

To provide external contexts from common world knowledge, we include 1-billion-word-dataset [22] into training vector representations. The data has been widely used for language model benchmarking and monolingual language model training in machine translation tasks.

Dependency structures of sentences are parsed by Stanford Core NLP tool [23].

The representation training setup is configured with window size of 5 for context subsampling and vector size of 100.

Comparing with dependency tree based approach, word sequence based approach was employed with the above same setting.

2.2.3.2 Retrieval

For this task, relevant scores are determined by cosine similarity of each document component vector pairs as follows.

$$RelScore(Q, A) = \max_{q \in Q, a \in A} cosine(v_q, v_a) \tag{2.1}$$

where $Q - A$ is a question-article pair, q, a are components of question Q and answer A respectively, and v_q, v_a are corresponding component vectors. The highest scored article is then selected.

2.2.3.3 Evaluation Metric

Performance for the task was evaluated using precision, recall and F-score (F1) defined in [21].

Table 2.2.1: Experimental results of statute law retrieval task

Models		**Precison**	**Recall**	**F1**
iLis7-1 (state-of-the-art)		0.734	0.554	0.632
This work				
cross-reference	**external corpus**	**dependency tree based contexts**		
No	No	0.397	0.282	0.330
Yes	No	0.346	0.245	0.287
No	Yes	0.487	0.345	0.404
Yes	Yes	0.513	0.364	0.426
cross-reference	**external corpus**	**word sequence based contexts**		
No	No	0.064	0.046	0.053
Yes	No	0.026	0.018	0.021
No	Yes	0.268	0.191	0.223
Yes	Yes	0.256	0.181	0.213

Experimental results (Table 2.2.1) show the effect of this method on legal information retrieval task. Performance increases when including common corpus into training vector representations, and further with cross-references in the cases of using dependency tree structures to construct contexts. Besides, dependency tree based contexts show advantage over word sequence based contexts.

2.2.4 Summary

We have presented an approach of learning vector representation of document components via context expansion with document hierarchy and cross-references and apply the method to legal information retrieval task. Results show that dependency trees show potential of better representing local contexts which are used to learn text vectors for similarity measure. In this section, dependency trees are used in unsupervised learning fashion though, in the next section, we present the use of dependency trees in supervised learning fashion where the contribution of dependency-tree based contexts are discussed.

2.3 Encoding Local Contexts of Sentences with Convolutions on pq-Gram Representations of Dependency Trees

We present our sentence encoding approach that uses local contexts constructed from pq-gram representations of a sentence's dependency tree. The context localization scope can be adjusted through parameters p, the dependency depth, and q, the dependency width, which allows controlling context-sensitivity. We show competitive results of using our sentence encoding approach for sentence-pair modeling tasks.

Sentence encoding is one important step in common deep neural architecture based text modeling, which maps a lexical sentence into its embedding in computational real-value space. The sentence embeddings, based on tasks or learning problems, contain useful information for identifying themselves. Sentence embeddings are usually built from their content word embeddings with certain composition, for instance, sequential composition and tree composition.

Sequential composition captures the temporal structure of the sentence, the natural way of speaking/writing/listening/reading a sentence. This composition, however, does not explicitly capture syntactic constraints of how to compose the meaning of the sentence, thus minor sequence modification may lead to major shift in sentence embedding.

Tree composition captures explicit word/phrase relationship, for instance, syntactic trees, and dependency trees. The composition contains high level semantic dependency which is not trivial to neural networks even with the ability of automatically feature learning, where the unsupervised parsers are nowhere near supervised parsers [1, 2]. Hence, tree composition provides more information of how to interpret the sentence to learning models which do not know how to do so from initialization.

Aside from the composition approach, we can further build an intermediate composition step: local contexts. Local contexts are the surrounding words of a target word, where the meaning of the target word can be made clearer or less ambiguous. In other words, the composition process disambiguates each word before composing it into sentence embeddings. The local contexts can be formed by extracting n-grams (in sequential composition), or head-children subtrees (tree composition). Besides, a sentence with a chosen composition structure, can be decomposed into a set of local contexts, which results in more relaxed representation which is less sensitive to word change and may benefit the task regarding similarity/relatedness.

We propose to represent a sentence as pq-grams of its dependency tree. This represents sentence local contexts with parameters p, the dependency depth, and q, the

dependency width. Our approach utilizes tree composition but not with head-children fashion. In the head-children fashion, local contexts contain one head (target word) and all its children, whereas, our approach produces local contexts with ancestors of the target word and only a limited number of its children, described as parameters (p, q).

We apply the encoding approach to sentence-pair modeling tasks including recognizing textual entailment and measuring relatedness. We adapt convolutional neural networks (CNNs), which are successfully used in text modeling [3, 4, 5], to encode pq-gram representation into latent space. Then, we apply various sentence-pair encoding composition algorithms, namely, basic pooling, global comparison, local comparison with pq-alignment. The inclusion of multi-pq alignment, where we compute alignment of pq-gram encodings from different (p, q) values, shows good results in textual entailment and text relatedness tasks.

Sequential and tree composition approaches are used interchangeably or combined together in sentence encoding. In sequential composition approach , common architecture are based on long short-term memories (LSTMs) [6], and CNNs [3, 4]. LSTMs process a sentence from the beginning to the end where each word is encoded from its initial embedding and current temporal state composed from the previous words. Bidirectional-LSTMs (biLSTMs) process the sentence from both directions, then, produces richer representation. CNNs process a sentence by applying their shared filters to each n-gram region. In tree composition approach, LSTMs and CNNs are designed to operate over trees. Tree-LSTMs process a sentence's tree from leafs to root [7, 6, 8]. Tree-CNNs process a sentence's tree by applying their shared filters to each sub-tree consisting of one head word and its children [5].

We approach sentence encoding in a way that allows capturing clear semantic of the local contexts/regions of a sentence. Our approach benefits the tasks, for example, sentence-pair modeling tasks such as textual entailment and text relatedness, whose focus is on some parts of the sentences with various degrees of reception. One way that we can represent the reception degree is using CNNs with multiple window sizes which capture local context at multiple levels of granularity. The CNNs on the sequential representation of a sentence, however, also capture n-gram regions with unclear semantic. We sought out the method to not only capture local contexts but also capture regions with clear semantic. We, then, look at the dependency tree of a sentence, which represents the syntactic/semantic relations of each word-pair. Besides, the sub-trees (which are also dependency trees) of the dependency tree represent a clear semantic. Furthermore, a node is added to or removed from a dependency tree, the specificness of the semantic represented by that tree is, then, increased/reduced. This is where we adopt pq-gram model as a method for representing local contexts with clear semantic from dependency trees and multiple pq values for multiple granularity. We then use CNNs to encode pq-grams into latent space where pq-grams can be compared and aggregated to later provide sentence-pair relationship for tackling the sentence-pair modeling tasks.

2.3.1 Local Contexts of Sentences from pq-Gram Representations of Dependency Trees

The dependency tree of a sentence represents the syntactic/semantic relations of each and every word-pair (Fig. 2.3.1). The relations not only define the semantic composition of the sentence but also exists in other sentences. While a sub-tree of the dependency tree represents a clear semantics, that semantics can also be found in other sentences.

In other words, a sub-tree of the dependency tree can be seen as a local context, and a sentence is composed from local contexts.

pq-Gram model is the method of representing an ordered tree for approximately matching hierarchical data[24]. Given an ordered tree, a *pq*-gram is one subtree where contiguous q nodes share the same set of p contiguously connected ancestors. In the case of dependency trees, in which the siblings are unordered, we restrict the order to be word-order. Each *pq*-gram of a dependency tree represents a local context with dependency depth p and dependency width q. With small p, q, the *pq*-grams more commonly exists in other sentences, the semantics are broader. As p, q increase, the existence becomes less common, the semantics are, thus, narrower.

We define a *pq*-gram local context as:

$$(w_p^{\mathfrak{a}}, ..., w_2^{\mathfrak{a}}, w_1^{\mathfrak{a}}, w_1^{\mathfrak{c}}, w_2^{\mathfrak{c}}, ..., w_q^{\mathfrak{c}}) \tag{2.2}$$

where $w_i^{\mathfrak{a}}$ are ancestors of $w_i^{\mathfrak{c}}$, and each $w_i^{\mathfrak{a}}$ is a direct parent of $w_{i-1}^{\mathfrak{a}}$. $w_1^{\mathfrak{c}}, w_2^{\mathfrak{c}}, ..., w_q^{\mathfrak{c}}$ are ordered by word-order. $w_1^{\mathfrak{a}}$ is the anchor node of the *pq*-gram. The *pq*-gram representation of the sentence (dependency tree) is defined as the sequence of all *pq*-grams ordered by the anchor nodes and siblings (Fig. 2.3.1).

2.3.2 Encoding Local *pq*-Gram Contexts with Convolutional Operations

We employ CNNs for mapping *pq*-grams into latent space with the following convolutional operation:

$$\mathbf{c}_i = ReLU(\mathbf{W}^c \left[\mathbf{v}(w_p^{\mathfrak{a}}); ...; \mathbf{v}(w_2^{\mathfrak{a}}); \mathbf{v}(w_1^{\mathfrak{a}}); \mathbf{v}(w_1^{\mathfrak{c}}); \mathbf{v}(w_2^{\mathfrak{c}}); ...; \mathbf{v}(w_q^{\mathfrak{c}})\right] + \mathbf{b}^c) \tag{2.3}$$

where, $\mathbf{v}(\cdot) : \mapsto \mathbb{R}^d$: word embedding vector lookup map, $[\cdot] \in \mathbb{R}^{d(p+q)}$: concatenated embedding vector, $\mathbf{W}^c \in \mathbb{R}^{k \times d(p+q)}$: convolution kernel matrix with k filters, $\mathbf{b}^c \in \mathbb{R}^k$: bias vector, $\mathbf{c}_i$: encoding vector of the i^{th} *pq*-gram, $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N] \in \mathbb{R}^{k \times N}$: encoding matrix of a sentence, N: total number of *pq*-grams, $ReLU$: rectified linear unit activation.

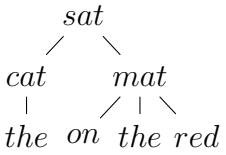
2.3.3 Experiments on Text-Pair Modeling Tasks

A sentence-pair modeling task is, given a sentence-pair consisting of two sentence namely a and b, to predict/measure the relationship of a and b. Instances of the tasks are entailment (entailment, contradiction, and neutral), and relatedness. Entailment tasks can be treated as multi-class classification, and relatedness tasks can be treated as scoring regression.

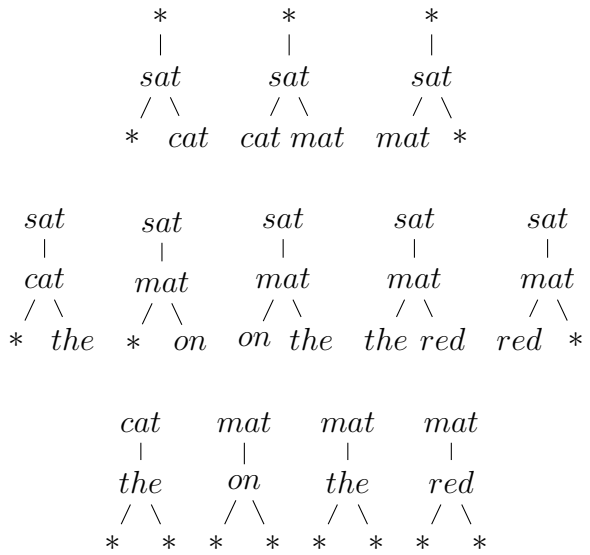
The relatedness and entailment of a sentence-pair (a, b) is shown through the examples (from SICK dataset[25]) in Tables 2.3.1, and 2.3.2.

We build on top of *pq*-encoding module with different sentence encoding composition described as the following 3 architectures (Fig.2.3.2):

- *Encoding-Base-Prediction*: this model encodes input *pq*-grams, performs pooling to obtain sentence encodings, concatenates the encodings, feeds them to prediction module, and finally outputs the class/score.



(a) Dependency tree



(b) $2,2$*-Grams*

$w_2^{\mathfrak{a}}$	$w_1^{\mathfrak{a}}$	$w_1^{\mathfrak{c}}$	$w_2^{\mathfrak{c}}$
cat	the	*	*
sat	cat	*	the
*	sat	*	cat
*	sat	cat	mat
*	sat	mat	*
mat	on	*	*
mat	the	*	a
mat	red	*	a
sat	mat	*	on
sat	mat	on	the
sat	mat	the	red
sat	mat	red	*

(c) $2,2$*-Gram representation*

Figure 2.3.1: Example of the pq-gram representation of a dependency tree of sentence "the cat sat on the red mat", with $(p, q) = (2, 2)$. Dummy nodes (*) are for border padding.

Table 2.3.1: Example sentence pairs in relatedness task (on a 5-point rating scale).

Relatedness score	Example
1.6	*a*: A man is jumping into an empty pool *b*: There is no biker jumping in the air
2.9	*a*: Two children are lying in the snow and are making snow angels *b*: Two angels are making snow on the lying children
3.6	*a*: The young boys are playing outdoors and the man is smiling nearby *b*: There is no boy playing outdoors and there is no man smiling
4.9	*a*: A person in a black jacket is doing tricks on a motorbike *b*: A man in a black jacket is doing tricks on a motorbike

Table 2.3.2: Example sentence pairs in entailment task.

Entailment label	Example
entailment	*a*: Two teams are competing in a football match *b*: Two groups of people are playing football
contradiction	*a*: The brown horse is near a red barrel at the rodeo *b*: The brown horse is far from a red barrel at the rodeo
neutral	*a*: A man in a black jacket is doing tricks on a motorbike *b*: A person is riding the bicycle on one wheel

- *Encoding-GlobalComparison-Prediction*: this model operates similar to "Encoding-Base-Prediction" except that instead of only feeding the concatenated sentence encodings, it feeds additionally the sentence encodings comparison to the prediction module.
- *Encoding-LocalComparison-Prediction*: this model operates differently from the other two. Before compute the sentence encodings, it aligns the *pq*-gram encoding, which models sentence-pair association in the form of soft alignment among *pq*-grams.

We now describe each module in our architectures.

2.3.3.1 Encoding

Given a sentence pair, we apply convolution operations (Section 2.3.1) to obtain two encoding matrices $\mathbf{C}^a$ and $\mathbf{C}^b$ corresponding to sentences a and b respectively.

We can, then, obtain the global sentence encoding $\mathbf{s}^a, \mathbf{s}^b \in \mathbb{R}^{2k}$ by pooling as:

$$\mathbf{s} = [\max\{\mathbf{C}\}; \text{avg}\{\mathbf{C}\}] \tag{2.4}$$

where max, average (avg) are operated over the whole sequence for each dimension of vectors $\mathbf{c}_i^a$, $\mathbf{c}_j^b$. The max pooling confirms the existence of patterns. In case of *pq*-gram encoding, it checks if the kernel W^c matches any *pq*-gram of a given sentence and reports the highest matching values. The average pooling, differently, captures the repetition of such patterns as how much the kernel W^c matches all of the *pq*-grams.

In the base architecture, the encodings are concatenated to obtain the feature vector $\mathbf{z} \in \mathbb{R}^{4k}$:

$$\mathbf{z} = [\mathbf{s}^a; \mathbf{s}^b] \tag{2.5}$$

which is forwarded to the prediction module.

2.3.3.2 Sentence-Pair Comparison

The sentence-pair comparison can be done at two different levels: global and local. Global comparison associates the composed encoding of the sentences (sentence level encoding via pooling) which have no information of individual sentence components. Local comparison looks at every encoded *pq*-gram of sentence a and b, and produces the association among them.

Global comparison: Sentence encoding comparison

Differing from the base architecture, after pooling operation in Eq. 2.4, we compute the similarity and dissimilarity of the sentence encodings via element wise multiplication $(. \circ .)$ and subtraction $(. - .)$ to obtain the feature vector $\mathbf{z} \in \mathbb{R}^{8k}$:

$$\mathbf{z} = [\mathbf{s}^a; \mathbf{s}^b; \mathbf{s}^a \circ \mathbf{s}^b; \mathbf{s}^a - \mathbf{s}^b] \tag{2.6}$$

This is seen as comparing the encoding features produced by the same convolution filter on each sentence of the pair. Two similar *pq*-grams found in each of the pair would result in high multiplication and low subtraction.

Local comparison: *pq*-gram encoding alignment

Soft aligning each *pq*-gram of sentence a with all *pq*-grams of sentence b and vice versa.

$$e_{i,j} = \langle \mathbf{c}_i^a, \mathbf{c}_j^b \rangle \tag{2.7}$$

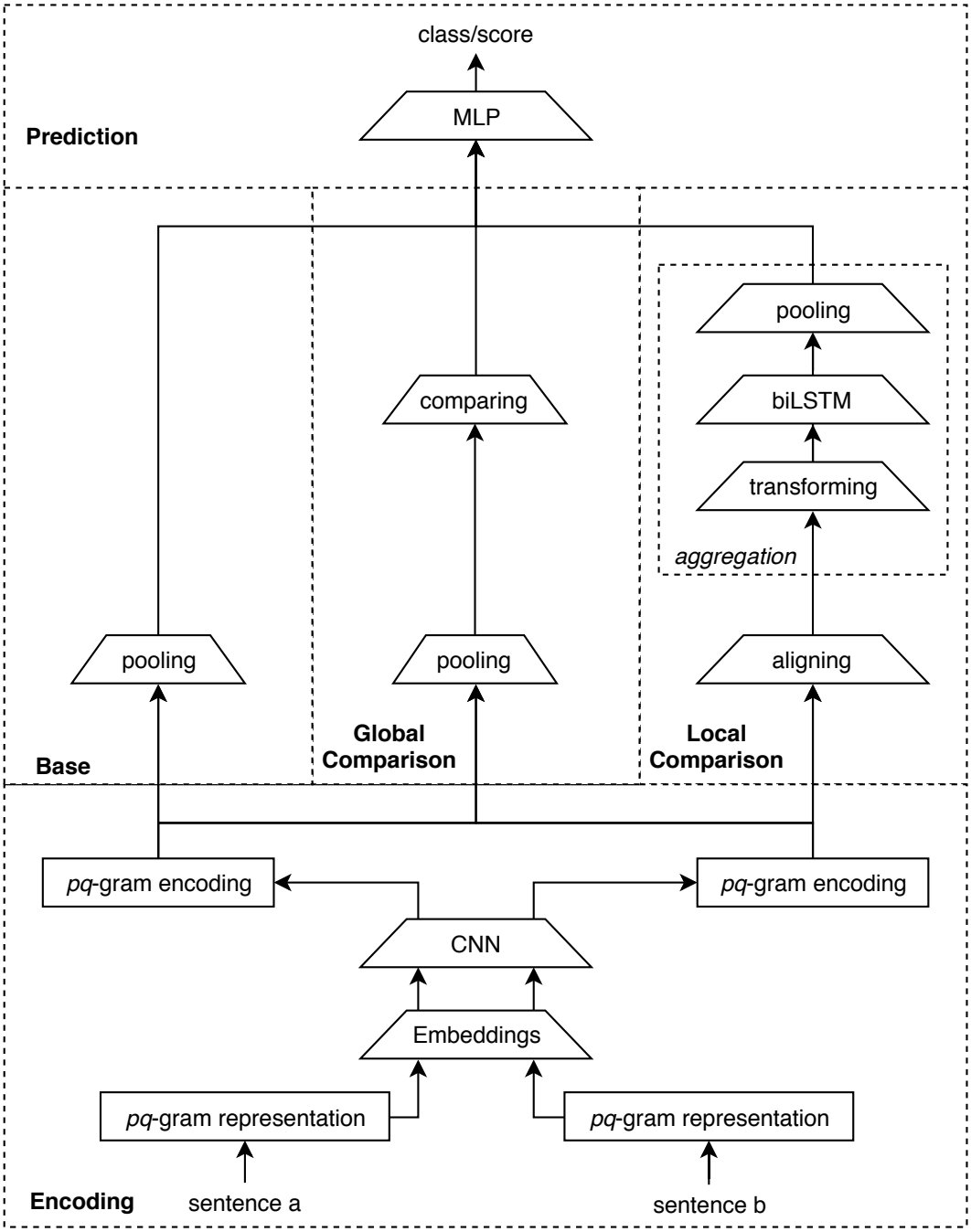


Figure 2.3.2: Sentence-pair modeling pipelines in our approach.

Soft alignment normalized weights:

$$\bar{e}^a_{i,j} = \frac{exp(e_{i,j})}{\sum_j exp(e_{i,j})} \qquad \bar{e}^b_{i,j} = \frac{exp(e_{i,j})}{\sum_i exp(e_{i,j})} \tag{2.8}$$

Soft alignment for pq-gram i^{th} of sentence a, and pq-gram j^{th} of sentence b:

$$\bar{\mathbf{c}}^a_i = \sum_j \bar{e}^a_{i,j}\mathbf{c}^b_j \qquad \bar{\mathbf{c}}^b_j = \sum_i \bar{e}^b_{i,j}\mathbf{c}^a_i \tag{2.9}$$

The alignment features $\bar{\mathbf{c}}^a_i$ is seen as the composition of pq-grams in sentence b those are compatible with pq-gram i^{th} of sentence a, and vice versa.

Then we compose the encoding of each pq-gram consisting of convolutional features, alignment features, similarity and dissimilarity between them as:

$$\hat{\mathbf{c}}_i = [\mathbf{c}_i; \hat{\mathbf{c}}_i; \mathbf{c}_i \circ \bar{\mathbf{c}}_i; \mathbf{c}_i - \bar{\mathbf{c}}_i] \tag{2.10}$$

At this step, we can perform multi-pq alignment: aligning pq-grams having different (p, q) values. Multi-pq alignment allows us to compare local contexts with similar semantics but different lexical granularity.

Feature aggregation to produce sentence encoding with three steps: transformation, temporal reading, and pooling.

Transformation:

$$\tilde{\mathbf{c}}_i = ReLU(\mathbf{W}^d\hat{\mathbf{c}}_i + \mathbf{b}^d) \tag{2.11}$$

Temporal reading with bidirectional LSTM:

$$\dot{\mathbf{c}}_i = [\overrightarrow{LSTM}(\tilde{\mathbf{c}}_i); \overleftarrow{LSTM}(\tilde{\mathbf{c}}_i)] \tag{2.12}$$

Pooling:

$$\mathbf{s} = [\max\{\dot{\mathbf{c}}_i\}; \text{avg}\{\dot{\mathbf{c}}_i\}] \tag{2.13}$$

where max and average (avg) are the same operators as in Eq. 2.4.

Finally, the feature vector $\mathbf{z} = [\mathbf{s}^a; \mathbf{s}^b]$ is obtained.

2.3.3.3 Prediction

Multilayer Perceptron with one hidden layer and one output layer, receives the input feature vector $\mathbf{z}$ and output corresponding label depending on the given task.

$$\mathbf{h} = \tanh(\mathbf{W}^h\mathbf{z} + \mathbf{b}^h) \tag{2.14}$$

$$\mathbf{y} = f(\mathbf{W}^o\mathbf{h} + \mathbf{b}^o) \tag{2.15}$$

where f is *softmax* if the task is recognizing textual entailment, *linear* if the task is relatedness.

2.3.4 Experimental Settings

2.3.4.1 Datasets

- **SNLI** [26]: a natural language inference dataset which contains over 550K train, 10K development, and 10K test examples built on image captions. Given an image caption, three hypothesis sentences which are, in turn, entailed by, contradicted to, neutral with the caption, are generated.
- **SICK** [25, 27]: a sentence-pair dataset with about 10K pairs split into 4.5K train, 0.5K development, 5K test examples. The dataset is built from image and video descriptions, contains sentences that describe the same picture or video, thus, are lean on named entities and rich in generic terms. The process of creating the dataset encourages the development of approaches focusing on building compositional semantics step to understand sentence-pair relations. The dataset contains annotated labels for two tasks: recognizing textual entailment task (SICK-E), and relatedness task (SICK-R).

Table 2.3.3: Information of sentences' dependency trees for average tree depth, average number of children per tree node.

Datasets	Tree Depth	#Children/Node
SNLI		
premise	4.6	2.3
hypothesis	3.5	2.0
SICK		
premise	3.7	2.2
hypothesis	3.6	2.2

2.3.4.2 Model Settings

Our model hyper-parameters are described in table 2.3.4. We configure pq parameters as:

- We select up to $p = 2$ ancestors where the one ancestor is the direct parent to q siblings, and the other ancestor is the direct parent of the prior ancestor, so the pq window is anchored by one node where the other nodes are directly connected.
- We select up to $q = 2$ siblings as the average number of children per tree node in Table 2.3.3.

Table 2.3.4: *pq*-CNN model settings

Parameter	Information
pq	$(p, q) \in \{1, 2\} \times \{1, 2\}$ (4 options of single *pq*). Multi-*pq* setting combines all 4 options.
Word embedding	Pre-trained GloVe [16] with size of 300
$\mathbf{W}^c$	number of filters $k = 300$
$\mathbf{W}^d$	output size of 300
LSTMs	output size of 300 (total size of 600 for bidirectional LSTM)
$\mathbf{W}^h$	output size of 300
$\mathbf{W}^o$	output size of number of classes (recognizing textual entailment), 1 for relatedness.

We optimize our model parameters θ by minimizing categorical cross-entropy loss:

$$\mathfrak{L}(\mathbf{y}^*, \mathbf{y}|\theta) = -\sum \langle \mathbf{y}^*{}_i, \log(\mathbf{y}_i) \rangle \tag{2.16}$$

for recognizing textual entailment task, and mean square error for relatedness task

$$\mathfrak{L}(\mathbf{y}^*, \mathbf{y}|\theta) = \sum ||\mathbf{y}^*{}_i - \mathbf{y}_i||_2^2 \tag{2.17}$$

where $\mathbf{y}^*{}_i$, $\mathbf{y}_i$ are gold label and prediction respectively to training examples i, using Adam optimizer with initial learning rate of 0.0004 and gradient clipping with max norm of 10.0.

We evaluate the model performance by accuracy on recognizing textual entailment tasks, Pearson correlation + mean square error for relatedness tasks.

2.3.5 Experimental Results

At first, we observe that the performance difference with and without encoding comparison of our approach is larger than TBCNN [5] on SNLI dataset (Table 2.3.5). In our approach, the accuracy increases 6.7% when using global comparison and 11.7% when using local comparison on multi-*pq* representation. In TBCNN, however, the increment is 2.8%. Our approach achieves similar results as TBCNN with sentence encoding comparison but much lower without the comparison. We suspect this is because of the amount of local information provided by *pq*-grams. In TBCNN, the number of intermediary (subtree) encodings are $O(n)$ where n is the number of nodes, where in our approach, it is $O(nc)$, where c is the number of children per node. Besides, in TBCNN, the encodings are more specific to the sentence than ours, which is by the nature of *pq*-grams. Thus, the upper stream modules have to work with larger search space in our cases. The comparisons reduce the search space by producing the direct related features: comparing signal from the same filter over two sentences in global comparison; aligning related *pq*-grams in local comparison.

We also observe that using multi-*pq*-gram can yield better results than single *pq*-gram.

Table 2.3.5: Experimental results on text pair modeling related tasks. acc. : accuracy (%), r: pearson correlation, mse: mean square error. ESIM*: re-implemented.

	SNLI	SICK-E	SICK-R	
Model	acc.%	acc.%	r	mse
BiLSTM [28]	81.5	-	-	-
ESIM [6]	88.6	-	-	-
ESIM [28]	86.7	-	-	-
ESIM*	87.0	73.7	0.716	0.502
TBCNN [5] w/o encoding comparison	79.3	-	-	-
TBCNN [5] w/ encoding comparison	82.1	-	-	-
ECNU [27]	-	83.6	0.828	0.325
Illinois-LH [27]	-	84.6	0.799	0.369
***pq*-grams base (w/o sentence-pair comparison)**				
$(p, q) = (1, 1)$	73.9	62.2	0.248	0.977
$(p, q) = (1, 2)$	73.3	62.4	0.217	0.976
$(p, q) = (2, 1)$	73.1	60.7	0.265	0.969
$(p, q) = (2, 2)$	72.8	61.2	0.237	0.964
multi-*pq*	74.6	61.3	0.269	0.967
***pq*-grams w/ global comparison**				
$(p, q) = (1, 1)$	81.8	81.9	0.780	0.401
$(p, q) = (1, 2)$	81.6	82.2	0.797	0.373
$(p, q) = (2, 1)$	80.2	82.0	0.792	0.383
$(p, q) = (2, 2)$	79.4	81.5	0.799	0.370
multi-*pq*	81.3	83.7	0.812	0.347
***pq*-grams w/ local comparison**				
$(p, q) = (1, 1)$	85.9	81.7	0.802	0.366
$(p, q) = (1, 2)$	85.8	75.6	0.805	0.361
$(p, q) = (2, 1)$	84.3	78.6	0.790	0.388
$(p, q) = (2, 2)$	85.0	75.8	0.789	0.387
multi-*pq*	86.3	81.1	0.824	0.331

Using the similar attention mechanism (phrase-by-phrase attention in ESIM, *pq*-alignment in ours), our approach performs comparatively with ESIM in SNLI and MultiNLI datasets and outperforms them in SICK dataset. We also achieve comparable performance with top competitors in SemEval2014[27].

2.3.6 Summary

We have presented our approach for encoding sentences with convolutional operations on *pq*-gram representations of dependency trees, and our application on sentence-pair modeling. The representations are flexible with adjusting *pq* values while keeping the semantic dependency from dependency trees. Besides, the representations provide more local information in the form of intermediary encodings than n-gram and subtree (containing all children) representations by the average number of children per node, which, however, may require feature comparison/alignment to reduce search space. Our approach achieve competitive performance with related methods using tree composition in sentence encoding.

2.4 Chapter Summary

In this chapter, we have been going through several structures existing in documents and our approaches to model these structures in certain tasks. The structures are shown to be useful in analyzing texts. The possibility of using the structures in neural networks create the potential of building a robust document encoding model from neural networks for document processing tasks. As the next step, we will introduce our approach of exploiting the structures for tackling legal case retrieval task which is described in Chapter 5.

Chapter 3

Rhetorical Information Analysis in Legal Case Documents

3.1 Introduction

Discourse analysis has high-impact applications in natural language processing, for instances, text summarization [29, 30], sentiment analysis [31], and question answering [32]. The output structures of the analysis contain high-level relationship of between discourses and so provides valuable information to such the tasks. Figure 3.1.1 shows an example of rhetorical structure of a text analyzed according to the rhetorical structure theory by Mann et al. [9], where the sentences 5-7 are evident for claiming the statement in sentence 4.

Despite of a wide range of applications and the necessity for automatic court document processing, automatic rhetorical structure analysis has not been well noticed in legal domain.

Hachey et al. [10] describe a task of determining the rhetorical status of each sentence in a given document from a corpus of judgments of the UK House of Lords. The corpus is annotated with rhetorical statuses described in Table 3.1.1 with 7 types: FACT, PROCEEDINGS, BACKGROUND, FRAMING, DISPOSAL, TEXTUAL, and OTHER. Successfully identifying those statuses can help automatic information processing systems to comprehend or organize information in a court document more effectively. As shown in Table 3.1.2, with the rhetorical statuses, a system can identify the main statement in this segment is the DISPOSAL sentence supported by the surrounding FRAMING and BACKGROUND sentences.

Hachey et al. tackled this task with a variety of linguistic features. They perform POS tagging, Lemmatization, Named entity recognition, Chunking and clause identification and extract features of location, thematic words, sentence length, quotation, entities, and cue phrases. They, then, train classification models with the learning algorithms of decision tree, naive bayes, winnow, support vector machine.

In this paper, we describe our approach for solving the task by applying well-known deep neural networks. Deep learning has been shown effective towards natural language processing tasks including discourse analysis. We have achieved promising results for the task, which suggests the applicability of artificial neural module embedding rhetorical information for other tasks, for example, summarization and information retrieval.

Table 3.1.1: Description of rhetorical statuses [10]. The second column indicates the frequency of each rhetorical status presented in the data published by Hachey et al..

Label	**Freq.**	**Description**
FACT	862 (8.5%)	The sentence recounts the events or circumstances which gave rise to legal proceedings. E.g. On analysis the package was found to contain 152 milligrams of heroin at 100% purity.
PROCEEDINGS	2434 (24%)	The sentence describes legal proceedings taken in the lower courts. E.g. After hearing much evidence, Her Honour Judge Sander, sitting at Plymouth County Court, made findings of fact on 1 November 2000.
BACKGROUND	2813 (27.5%)	The sentence is a direct quotation or citation of source of law material. E.g. Article 5 provides in paragraph 1 that a group of producers may apply for registration . . .
FRAMING	2309 (23%)	The sentence is part of the law lord's argumentation. E.g. In my opinion, however, the present case cannot be brought within the principle applied by the majority in the Wells case.
DISPOSAL	935 (9%)	A sentence which either credits or discredits a claim or previous ruling. E.g. I would allow the appeal and restore the order of the Divisional Court.
TEXTUAL	768 (7.5%)	A sentence which has to do with the structure of the document or with things unrelated to a case. E.g. First, I should refer to the facts that have given rise to this litigation.
OTHER	48 (0.5%)	A sentence which does not fit any of the above categories. E.g. Here, as a matter of legal policy, the position seems to me straightforward.

1. Farmington police had to help control traffic recently
2. When hundreds of people lined up to be among the first applying for jobs at the yet-to-open Marriott Hotel.
3. The hotel's help-wanted announcement – for 300 openings – was a rare opportunity for many unemployed.
4. The people waiting in the line carried a message, a refutation, of claims that the jobless could be employed if only they shoed enough moxie.
5. Every rule has exceptions,
6. But the tragic and too-common tableaux of hundreds of even thousands of people snake-lining up for any task with a paycheck illustrates a lack of jobs,
7. Not laziness.

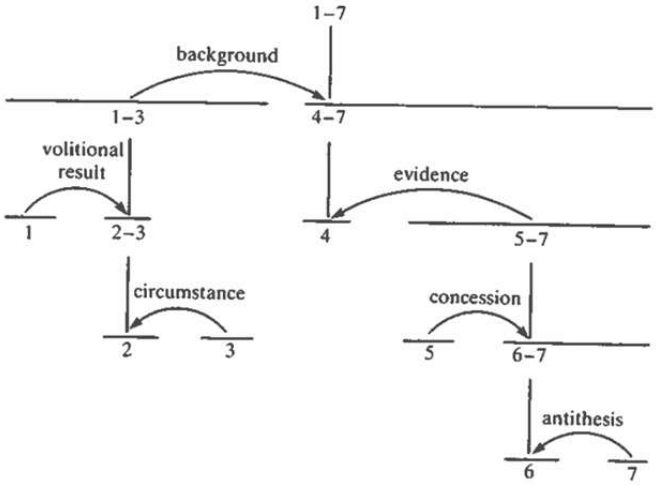


Figure 3.1.1: An example of parsing a text into rhetorical structure by Mann et al. [9].

Table 3.1.2: Example of rhetorical status annotated sentences in the corpus used by Hachey et al.

Rhetorical Status	**Sentence**
TEXTUAL	LORD NICHOLLS OF BIRKENHEAD
TEXTUAL	My Lords ,
...	...
FRAMING	This has long been axiomatic in this area of the law .
FRAMING	The matters the court may take into account are bounded only by the need for them to be relevant , that is , they must be such that , to a greater or lesser extent , they will assist the court in deciding which course is in the child 's best interests .
DISPOSAL	I can see no reason of legal policy why , in principle , any other limitation should be placed on the matters the judge may take into account when making this decision .
FRAMING	If authority is needed for this conclusion I need refer only to the wide , all embracing language of Lord MacDermott in J v C [1970] AC 668 , 710 - 711 .
BACKGROUND	Section 1 of the Guardianship of Infants Act 1925 required the court , in proceedings where the upbringing of an infant was in question , to regard the welfare of the infant ' as the first and paramount consideration ' .
...	...

3.2 Method

3.2.1 Problem Formulation

We treat the task of recognizing rhetorical statuses of the sentences in a court document as sequential labeling problem. We assume that the rhetorical role of a sentence not only depends on the contents it conveys but also the relation with other sentences. To model the dependency among sentences, we take the most simple approach: applying inter-sentence relationship modeling and inter-status relationship modeling. For the inter-sentence relationship modeling, we use a recurrent neural network described later in Section 3.2.2. For the inter-status relationship, we utilize conditional random field (CRF) [33] as the algorithm for capturing such dependency. Hence, the problem to solve is formulated as follows.

Given a document D which contains a list of sentences $S = \{s_1, s_2, ..., s_N\}$, the system should predict the rhetorical roles of the sentences as a list $R = \{r_1, r_2, ..., r_N\}$, where the system is trained to maximizing likelihood by minimizing the negative log likelihood given a training dataset $\mathcal{D} = \{(R, S)\}$:

$$L(D, \lambda) = -\sum_{(R,S)\in D} log\left(P(R|S, \lambda)\right) \tag{3.1}$$

where

$$P(R|S, \lambda) = \frac{1}{Z(S)} exp\left(\sum_{i=1}^{|S|} f(S, i, r_{i-1}, r_i, \lambda)\right) \tag{3.2}$$

where

$$Z(S) = \sum_{R'} exp\left(\sum_{i=1}^{|S|} f(S, i, r'_{i-1}, r'_i, \lambda)\right) \tag{3.3}$$

where λ is the set of parameters to be optimized, and f is the rhetorical status score or label score of sentences S given labels R, which is computed by combining rhetorical status transition scores and sentence features from a deep neural network described in the later section. The computation of the label scores is also described in[34], which is:

$$f(S, i, r_{i-1}, r_i, \lambda) = \mathbf{A}_{r_{i-1}, r_i} + F(S, i, r_i, \lambda_F) \tag{3.4}$$

where $F(S, i, r_i, \lambda_F)$ is the score of sentence i labeled with status r_i, and is output by the deep neural network, the set of parameters λ is $\{A_{i,j}\} \cup \lambda_F$. In this work, we use $F(.)$ as a linear function:

$$F(S, i, r_i, \lambda_F) = \langle \mathbf{w}_{r_i}, \hat{s}_i \rangle + b_{r_i} \tag{3.5}$$

where $\hat{s}_i$ is the encoding vector of sentence i output by the later described text encoding.

We also study the removal of the dependency modeling. With inter-sentence relationship modeling, we remove the recurrent neural network. With inter-status relationship modeling, we replace the CRF with a commonly used fully-connected neural layer with softmax activation.

$$\text{Fully-Connected}(S) = \text{softmax}(\mathbf{W}\hat{S} + \mathbf{b}) \tag{3.6}$$

where $\hat{S}$ is the encoding matrix of sentences S output by the later described text encoding.

3.2.2 Text Encoding Mechanisms

We employ two well-known techniques for text encoding, namely, Bidirectional Long Short-term Memory (BiLSTM) [35, 36] and Convolutional Neural Network (CNN) [37]. The BiLSTM is also used to model the inter-sentence relationship.

- **BiLSTM** captures the temporal information from both left-to-right and right-to-left directions of a sequence. LSTM provides its mechanism to gate information flow through each step during encoding the text sequence. Given a sequence $X = (\mathbf{x}_1, ..., \mathbf{x}_t, ..., \mathbf{x}_N)$, each step t of the sequence is encoded using an LSTM as follows.

$$\mathbf{f}_t = \text{sigmoid}(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{3.7}$$

$$\mathbf{i}_t = \text{sigmoid}(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{3.8}$$

$$\mathbf{o}_t = \text{sigmoid}(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{3.9}$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{3.10}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \text{sigmoid}(\mathbf{c}_t) \tag{3.11}$$

  Thus, we get the encoding $\mathbf{h}_t$ of the step t of the sequence. Finally, we collect all the encodings ($\mathbf{h}_t$):

$$\text{LSTM}(\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)) = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N) \tag{3.12}$$

  A BiLSTM is, straightforwards, composed of two LSTM layers running in two directions: left-to-right and right-to-left.

$$\text{BiLSTM}(\mathbf{X}) = [\overrightarrow{\text{LSTM}}(\mathbf{X}); \overleftarrow{\text{LSTM}}(\mathbf{X})] \tag{3.13}$$

- **CNN** captures the local information in a predefined local region of a sequence. Given a sequence $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$, a local region $\mathbf{X}_{t-l:t+r} = (\mathbf{x}_{t-l}, ..., \mathbf{x}_{t+r})$ is encoded using a CNN as follows.

$$\mathbf{h}_t = \text{ReLU}(\mathbf{W}\mathbf{X}_{t-l:t+r} + \mathbf{b}) \tag{3.14}$$

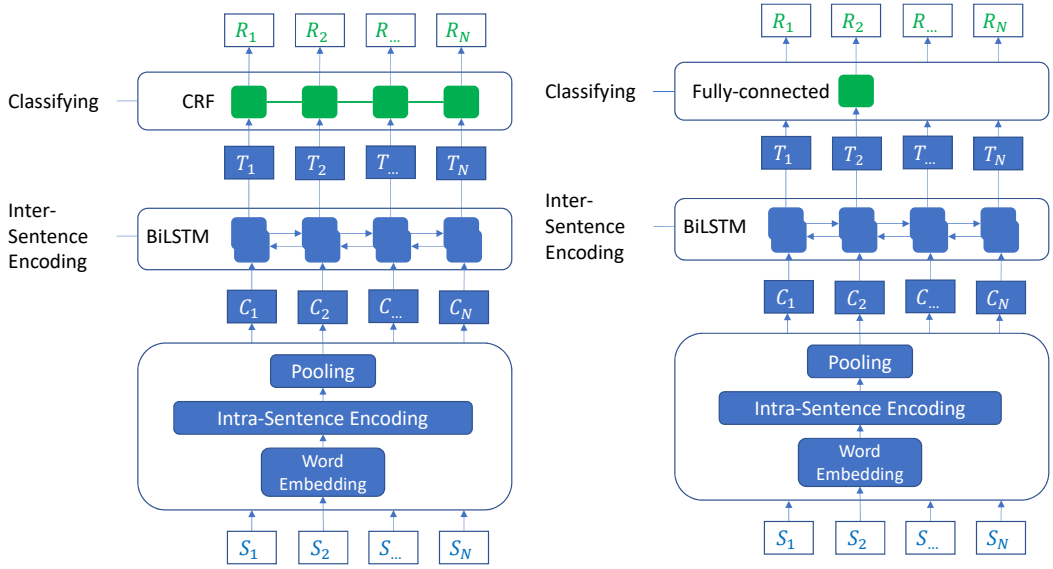
  For steps t such that $1 \leq t < l + 1$ or $N \geq t > N - r$, we use dummy zero-tokens $\mathbf{x}_d = 0$ for out-of-bound steps. Finally, we collect all the encodings ($\mathbf{x}_t$):

$$\text{CNN}(\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)) = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N) \tag{3.15}$$

3.2.3 Rhetorical Status Classification Models

The system is built upon a deep neural network, as shown in Figure 3.2.1, consisting of the following layers:

- Word Embedding: maps a word into a continuous vector space. We employ GloVe [16] which models contextual similarity among words. In other words, if two words are used in common contexts frequently, they are near each other in the vector space.



(a) Classifying layer using CRF (b) Classifying layer using Fully-Connected

Figure 3.2.1: Rhetorical Status Classification Models. We can place CNN or BiLSTM into the intra-sentence encoding layer, and use either fully-connected or CRF for the classifying layer.

- Intra-Sentence Encoding: we can encode the local information of a sentence in two ways:
  - BiLSTM: encoding the temporal information of the word sequence in both left-to-right and right-to-left directions.
  - CNN: applies convolutional operations on contiguous word gram within a limited window size, which captures the local contexts in a sentence.
- Pooling: pools out the most salient features, and transforms the variable length representation of a sentence into a fixed vector. For this layer, we apply max-over-time pooling as described by Kim et al.[37]. We apply the operation on the feature matrix $\mathbf{X}$ of a sentence, which is an array of vectors $\mathbf{x}_i$ each for word i.

$$\text{pooling}(\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)) = \begin{bmatrix} max(x_{11}, ..., x_{N1}) \\ ... \\ max(x_{1d}, ..., x_{Nd}) \end{bmatrix} \tag{3.16}$$

  where $\mathbf{x}_i \in \mathbb{R}^d$. We, then, receive a fixed vector $\in \mathbb{R}^d$ representing the one sentence.
- Inter-Sentence Encoding with BiLSTM: constraints the temporal dependency of the input sentence sequence. With bi-direction, information from the beginning and the end of the sequence can be taken into account by the decision time.
- Classifying: given the input sentence sequence represented by the above neural layers, we employ two options (Figure 3.2.1 (a) and (b)) for classifying layer:

– CRF: predicts sequentially dependent labels. The use of CRF on top of BiLSTM has shown effectiveness in natural language processing tasks, especially sequential tagging tasks [34, 38]. While BiLSTM binds one sentence with others with temporal order, thus puts dependency to the features of each sentence, CRF puts another dependency for the output rhetorical statuses. In other words, using BiLSTM + CRF, we can obtain two layers of dependency in both features and outputs.

– Fully-Connected (FC): predicts labels independently. Even though the prediction of the rhetorical status of one sentence does not directly depend on the predicted statuses of other sentences, the prediction of one sentence is still affected by the information from other sentences when we use FC on top of inter-sentence encoding which binds one sentence with others in temporal order. Complete independent prediction is done when the inter-sentence encoding is removed.

3.3 Experiments

3.3.1 Experimental Settings

The dataset used in our experiments is also used by Hachey et al. [10], and is a collection of 47 judgments of the House of Lord[1] from 2001 to 2003. As also described by Hachey et al., each judgment shows decision as the opinions of the Law Lords, which often starts with a statement of how the case came before the court, and sometimes will move to a recapitulation of the facts, on to discuss one or more points of law and then offer a ruling. The dataset contains 10,169 sentences annotated by 2 annotators with agreement of 0.83 kappa co-efficient.

For the deep neural model, we use the following setting:

- Embedding layer: GloVe embeddings with vector size of 300 [2],
- Encoding layers: we set the CNN local region parameters $l = 2, r = 2$ (used in Eq. 3.14), and tune the hidden size of CNN layer from the set $\{300, 500, 1000\}$.

  Similarly, we also tune the hidden size of BiLSTM layers from the set $\{300, 500, 1000\}$ for each of BiLSTM of intra-sentence encoding and inter-sentence encoding layers.

  We also experiment with the removal of the inter-sentence encoding layer to confirm the importance of inter-sentence relationship for recognizing rhetorical statuses.
- Classifying layer: two choices are considered, Fully-Connected and CRF, for checking if inter-status can be captured by CRF.

We perform 10-fold cross-validation and evaluate the models with micro-averaged F-score (also used by Hachey et al.).

[1] https://www.parliament.uk/business/lords/

[2] Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 300d vectors, https://nlp.stanford.edu/projects/glove/

Table 3.3.1: Micro-averaged F-score results. BiLSTM(1000)-BiLSTM(1000)-FC: model using BiLSTM as intra and inter sentence encoding layers with hidden sizes of 1000, and Fully-Connected as classifying layer.

Method	F-score (%)
Hachey et al. [10]	
C4.5	59.7
NB	51.7
Winnow	41.4
SVM	60.6
Our model	
BiLSTM(1000)-BiLSTM(1000)-FC	68.6

3.3.2 Experimental Results

Among all of our experimented settings, the model (BiLSTM(1000)-BiLSTM(1000)-FC) using BiLSTM with hidden size of 1000 for both intra and inter sentence encoding layers, and Fully-Connected as classifying layer yields the best performance with F-score of 68.6% (Figure 3.3.1). As shown in Table 3.3.1, our approach outperforms the models built with linguistic-features (Hachey et al. performed POS tagging, Lemmatization, Named entity recognition, Chunking and clause identification and extract features of location, thematic words, sentence length, quotation, entities, and cue phrases). We, though, have not yet incorporate any linguistic information other than word embedding which provide statistical features of contextual similarity, and the amount of training data is quite insufficient for training deep learning models.

The removal of inter-sentence encoding layer reduces performance substantially. This suggests the importance of capturing inter-sentence relationship for predicting the rhetorical status of a sentence.

In a majority of the experiments, regarding the use of classifying layer, models with Fully-Connected outperform models with CRF. Since CRF does not show significant contribution to the performance of the models, we hypothesize that CRF is not suitable for capturing the inter-status relationship between rhetorical statuses of an input document. Besides, when we remove the inter-sentence encoding layer, CRF equipped models achieve marginally better results than Fully-Connected equipped models. When inter-sentence relationship is not captured in the lower layer, CRF could capture some kind of dependency, but possibly too weak.

For the intra-sentence encoding, BiLSTM shows better performance than CNN. It shows that temporal encoding by BiLSTM provides more useful information for detecting rhetorical statuses than local context encoding by CNN.

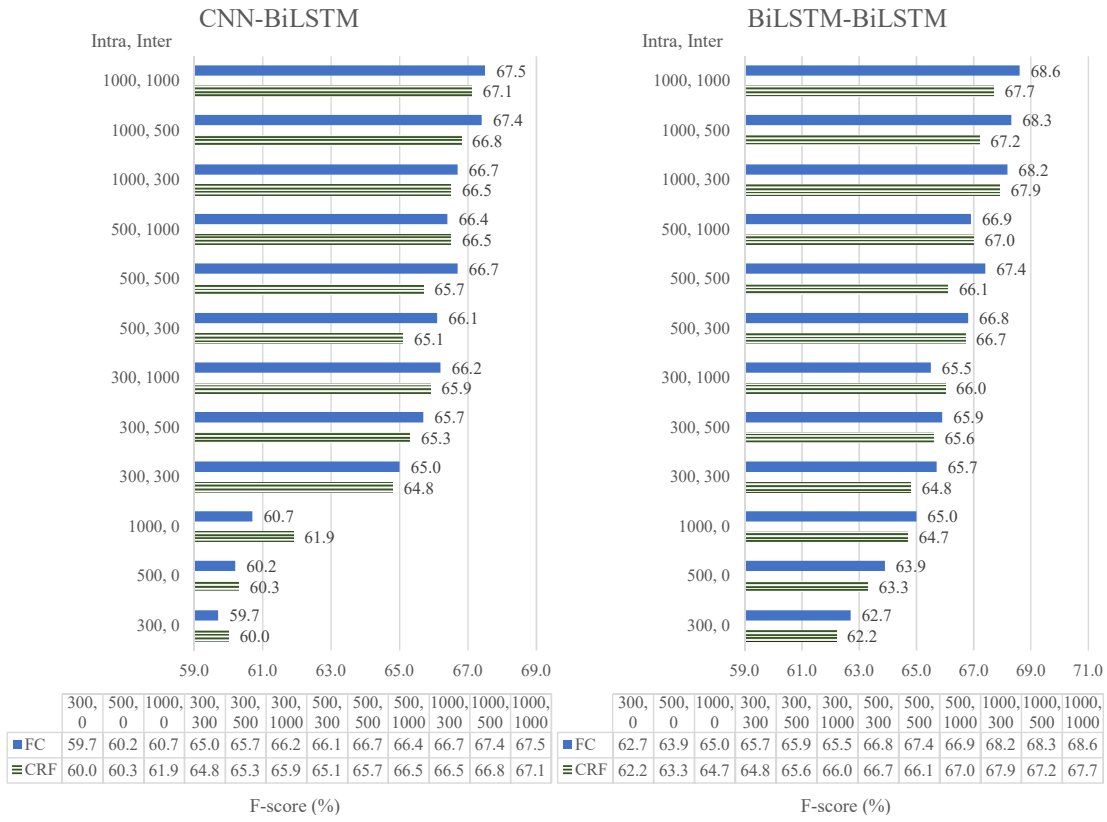


Figure 3.3.1: Results of tuning hidden sizes of neural layers. CNN and BiLSTM are in turn used as intra-sentence encoder and BiLSTM is used as inter-sentence encoder. The vertical axis is the intra-inter pair of hidden sizes for intra-sentence encoding and inter-sentence encoding layers, respectively. When **Inter=0**, we remove the inter-sentence encoding layer from the model's architecture. *FC: Fully-Connected, CRF: Conditional Random Field.*

Table 3.3.2: Performance of BiLSTM(1000)-BiLSTM(1000)-FC for each rhetorical status.

Rhetorical Status	**Prec(%)**	**Rec(%)**	**F-score(%)**
FACT	73.9	66.1	68.0
PROCEEDINGS	65.9	68.7	67.1
BACKGROUND	68.1	72.8	69.9
FRAMING	61.7	66.8	63.9
DISPOSAL	73.0	46.9	56.9
TEXTUAL	84.8	78.4	81.4
OTHER	10.0	3.8	5.5

While looking at the performance of our approach for each rhetorical status shown in Table 3.3.2, even though the dataset is limited, and the distribution of labels is quite imbalance, we get satisfactory performance across all rhetorical statuses. The TEXTUAL status gets the highest performance of 81.4% F-score despite of being the second lowest-frequency class (7.5% of sentences), while the OTHER status gets 5.5% F-score with being the lowest-frequency class (0.5% of sentences).

3.4 Summary

We have presented our approach for building rhetorical status recognition systems for processing court documents using deep learning models. We solve the recognition task as sequential labeling problem. We have observed the inter-sentence relationship captured by BiLSTM contributes significantly to improve performance. However, inter(-rhetorical)-status relationship is not effectively captured by CRF. The curent best model is BiLSTM-BiLSTM-FC, where we use BiLSTMs for intra and inter sentence encoding layers, and Fully-Connected as classifying layer. We have achieved encouraging results since, we have not yet utilized any linguistic features, but only statistical features by deep neural network from a considerably insufficient dataset. With the performance in F-score of 68.6%, we still have room for improvement, especially, with incorporating already designed linguistic features. The applicability of deep neural network to this task sets the step for incorporating rhetorical information into more sophisticated deep learning models of tasks such as summarization, information retrieval.

Chapter 4

Document Encoding via Summarization

4.1 Introduction

In human perspective, summarization means producing a concise text block for quick comprehension given a document which would, otherwise, require much more effort. In machine perspective, the above is also true in the regard of reducing computation cost though, summarization also means showing to machines how to weight the contents according to some pre-defined orientation, for example, human drafted summaries. Content weighting instead of content removal is preferable when the computation cost is negligible compared to information loss. This is also the direction of this chapter, providing a method for weighting document contents based on some pre-defined orientation, and then map documents into a vector space which embeds such orientation.

The method is applied to legal case retrieval task. The legal case retrieval task involves reading a new case, and then extracting cases supporting the decision of the new case. Automatic legal document processing systems can speed up significantly the work of experts, which, otherwise, requires significant time and efforts. The systems, in place of experts, process over enormous amount of documents, for example, legal case reports, 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]).

A case document contains a large volume of contents as the case may last days or even years. This one problem challenges the construction of an effective automatic legal case retrieval system. One approach is to identify the gist of the documents, specifically, catchphrases for legal case documents. "Catchphrases have an indicative function rather than informative, they present all the legal point considered instead that just summarizing the key points of a decision" [11]. Catchphrases give a quick impression on what the case is about: "the function of catchwords is to give a summary classification of the matters dealt with in a case. [...] Their purpose is to tell the researcher whether there is likely to be anything in the case relevant to the research topic" [12]. On one hand, catchphrases help lawyers/researchers quickly grasp the points of a case, without having to read the entire document, which saves significant time and effort for finding/studying relevant cases. On the other hand, catchphrases help improves the performance of automatic case retrieval systems.

Despite of the benefits, catchphrases are not always available in legal case documents,

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

and are drafted by legal experts, which requires huge efforts when considering the enormous number of legal case documents. It is, therefore, crucial to build automatic catchphrase generation systems for both old documents not having drafted catchphrases and new documents. Developing such systems, however, is challenging as the complexity of catchphrases shown in Table 4.1.1.

Approaches for generating catchphrases are based on phrase scoring derived from common model for retrieval: lexical matching with term frequency-inverse document frequency [13, 11, 14]. The approaches are bounded by the limit of lexical matching, and corpus-wide statistical information. The limit of lexical matching can be lifted by moving to distributed vector space, for instance, distributed word embeddings in which common models are *Word2Vec* [15] and *GloVe* [16]. Corpus-wide statistical information has limit capability to identify catchphrases which are not really specific to some document but commonly used in several others.

We present our work on developing a legal case summarization system and on top of its core component - phrase scoring framework, building a legal case retrieval system.

First, we build a learning model to extract catchphrases for new documents with the knowledge from previously seen documents and the expert drafted catchphrases thereof. Our system utilizes deep neural networks which have been widely used in natural language processing [17] to learn the direct relationship between gold catchphrases and document phrases. This results in our phrase scoring framework which is used to identify important phrases from a given legal case document.

On top of the phrase scoring framework, we develop our legal case document representation method which summarizes the document into continuous vector space. The representation is used for constructing case relevance ranking model, the core component of the retrieval system.

4.2 Summarization with Phrase Scoring Framework

We present our method of automatic summarization via phrase scoring with multi-level contextual features by deep neural networks. As case study, we build a system extracting catchphrases from legal case documents.

4.2.1 Phrase Scoring Framework

In this phase, we present our scoring model and how to train it using documents and their corresponding drafted summary.

4.2.1.1 Constructing our scoring model architecture

We score each phrase in a document based on its contexts: its words, enclosing sentence, and document. Our approach takes advantage of the core property of word embedding techniques by Google word2vec, GloVe, etc.: contextual similarity, the similarity of two words is measured as the amount of common contexts where they appear. Besides, the constructed context representation is multi-level contextual where the hierarchical structure presents.

Table 4.1.1: Example of catchphrases found in legal case reports[13].

COSTS - proper approach to admiralty and commercial litigation - goods transported under bill of lading incorporating Himalaya clause - shipper and consignee sued ship owner and stevedore for damage to cargo - stevedore successful in obtaining consent orders on motion dismissing proceedings against it based on Himalaya clause - stevedore not furnishing critical evidence or information until after motion filed - whether stevedore should have its costs - importance of parties cooperating to identify the real issues in dispute - duty to resolve uncontentious issues at an early stage of litigation - stevedore awarded 75% of its costs of the proceedings
CORPORATIONS - winding up - court-appointed liquidators - entry into agreement - able to subsist more than three months - no prior approval under s 477(2B) of Corporations Act 2001 (Cth) - application to extend "period" for approval under s 1322(4)(d) - no relevant period - s 1322(4)(d) not applicable - power of Court under s 479(3) to direct liquidator - liquidator directed to act on agreement as though approved - implied incidental powers of Court - prior to approve agreement - power under s 1322(4)(a) to declare entry into agreement and agreement not invalid - COURTS AND JUDGES - Federal Court - implied incidental power - inherent jurisdiction
MIGRATION - partner visa - appellant sought to prove domestic violence by the provision of statutory declarations made under State legislation - "statutory declaration" defined by the Migration Regulations 1994 (Cth) to mean a declaration "under" the Statutory Declarations Act 1959 (Cth) in Div 1.5 - contrary intention in reg 1.21 as to the inclusion of State declarations under s 27 of the Acts Interpretation Act - statutory declaration made under State legislation is not a statutory declaration "under" the Commonwealth Act - appeal dismissed

We adapt convolutional neural networks (CNNs), which are successfully used in text modeling [37, 4, 39, 40], to encode each local context into latent feature space. Specifically, document phrase (summary phrase) features are captured by applying convolutional operations with window size l covering l words.

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:

$$\mathbf{f}_{p_j} = ReLU\left(\mathbf{W}^c\left[\mathbf{v}(w_j); \mathbf{v}(w_{j+1}); ...; \mathbf{v}(w_{j+l-1})\right]\right) \tag{4.1}$$

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.

Sentence (catchphrase) features are, then, captured by applying max pooling over the whole sentence (catchphrase).

$$\mathbf{f}_{s_i} = \text{max-pooling}_j(\mathbf{f}_{p_j^{s_i}}) \tag{4.2}$$

$$\mathbf{f}_{c_i} = \text{max-pooling}_j(\mathbf{f}_{p_j^{c_i}}) \tag{4.3}$$

where max-pooling are operated over each dimension of vectors $\mathbf{f}_{p_{i,j}^s}$ ($\mathbf{f}_{p_{i,j}^c}$).

Document features are captured by applying max pooling over the document (not including summary). With the same max-pooling operation as above, we compute document features as:

$$\mathbf{f}_d = \text{max-pooling}_i(\mathbf{f}_{s_i}) \tag{4.4}$$

The document features depend on only the document sentence, thereby, independent from the gold summary which are obviously not available for new documents.

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

$$MLP(\mathbf{x}) = \text{sigmoid}(\mathbf{W}_2\ \tanh(\mathbf{W}_1\ \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \tag{4.5}$$

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

$$PS(p_s, s, d) = MLP\left(\left[\mathbf{f}_{p_j^{s_i}}; \mathbf{f}_{s_i}; \mathbf{f}_d\right]\right) \tag{4.6}$$

$$PS(p_c, c, d) = MLP\left(\left[\mathbf{f}_{p_j^{c_i}}; \mathbf{f}_{c_i}; \mathbf{f}_d\right]\right) \tag{4.7}$$

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

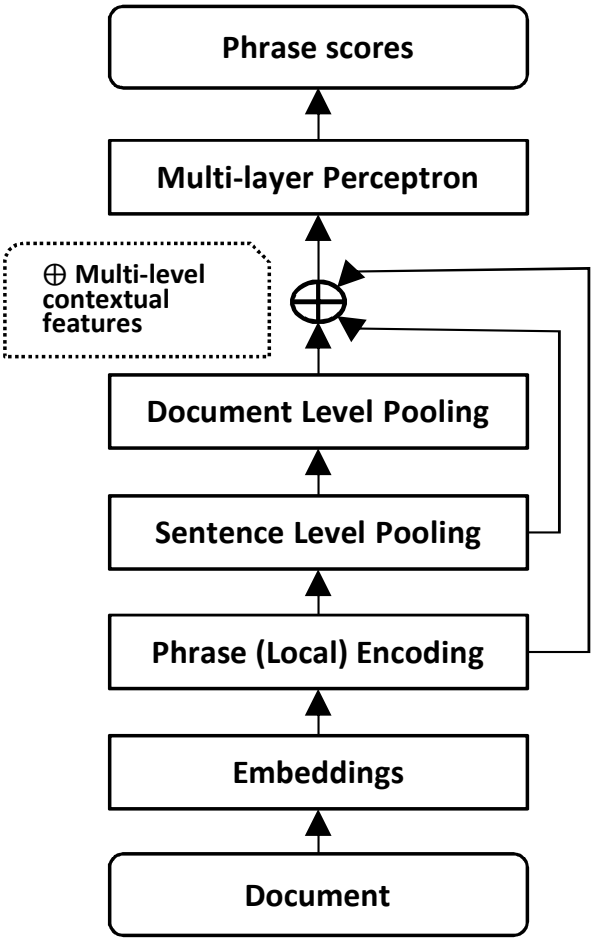


Figure 4.2.1: Phrase scoring framework

4.2.1.2 Training our scoring model

Main objective: given a document, summary phrases are "expected" to have higher score than document phrases.

First, we denote mean E and standard deviation std of word scores P for each document d in the following equations, which we will use to describe our objective as set of constraints, then formulated into loss function to be optimized.

$$E_c = E[PS(p_c, c, d)] \text{ where } p_c \in c, c \in d \tag{4.8}$$

$$std_c = std[PS(p_c, c, d)] \text{ where } p_c \in c, c \in d \tag{4.9}$$

$$E_s = E[PS(p_s, s, d)] \text{ where } p_s \in s, s \in d \tag{4.10}$$

$$std_s = std[PS(p_s, s, d)] \text{ where } p_s \in s, s \in d \tag{4.11}$$

$$E_{c,d'} = E[PS(p_c, c, d')] \text{ where } p_c \in c, c \notin d' \tag{4.12}$$

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 $(E_c + std_c) > (E_s + std_s)$.

(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 $(E_c - std_c) > E_s$.

We also add the following additional constraint to keep the scores from collapsing, which acts as regularization.

(o5) Scores should not have small variance: $std_c \not\approx 0, std_s \not\approx 0$.

The loss function, hence, is composed from the constraints **(o1-5)** as follows.

$$\begin{aligned}\mathfrak{L} = \sum_d \max(0, m - (&a_1(E_c - E_s) \\ &+a_2(\frac{1}{|\{d'\}|}\sum_{d' \neq d} E_{s'} - E_{c,d'}) \\ &+b_1((E_c + std_c) - (E_s + std_s)) \\ &+b_2((E_c - std_c) - E_s) \\ &-b_3(std_c) - b_4(std_s) \\ &))\end{aligned} \tag{4.13}$$

Note that rather imposing hard constraints, we compose the loss function with soft constraints. This means that some constraints may not be strictly satisfied after the training process. However, the violations of such constraints still incur certain losses and benefit the learning process.

4.2.2 Legal Case Summarization

4.2.2.1 Summary Generation by Phrase Concatenation

We generate a summary given a document by selecting and joining document phrases scored by the phrase scoring model. The process is as follows.

1. Rank document phrases by they phrasal scores.
2. Select phrases with scores from high to low.
3. Join overlapping phrases into a longer phrase.
4. 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.

We trained the phrase scoring model with settings shown in Table 4.2.1. We use two sets of loss coefficients: (i) the parameters used for COLIEE 2018 submission, (ii) the parameters used for COLIEE 2019 submission. While the parameter set (i) is copied from [41], the parameter set (ii) is obtained by random searching around the set (i) for better retrieval performance on COLIEE 2018 dataset.

We report the empirical evaluation of the phrase scoring model applied to case summarization. A predicted summary of a given case is composed according to Section 4.2.2. We evaluate the predicted summary with length-threshold t values from 10% to 50% of document length. The evaluation is performed with ROUGE metrics including: ROUGE-1, ROUGE-2, ROUGE-SU. Results of the evaluation are shown in 4.2.2.

Table 4.2.1: Phrase scoring model parameters. We use two sets of loss coefficients: (i) the parameters used for COLIEE 2018 submission, (ii) the parameters used for COLIEE 2019 submission. While the parameter set (i) is copied from [41], the parameter set (ii) is obtained by random searching around the set (i) for better retrieval performance on COLIEE 2018 dataset.

Parameter	**Description**
Embeddings (vector size d)	GloVe [16] $d = 300$
CNN filters c	300
CNN window size l	5
MLP hidden size	300
Optimizer	Adam[42]
Learning rate	0.0001
Gradient clipping max norm	5.0
Loss coefficients $(a_1, a_2, b_1, b_2, b_3, b_4)$	(i) $(1.0, 1.0, 0.5, 0.1, 0.01, 0.02)$ (ii) $(1.0, 1.7, 0.3, 0.7, 0, 0)$
Size of negative set $\lvert\{d'\}\rvert$	2

Table 4.2.2: Legal case summarization performance measured in ROUGE scores on dataset from COLIEE 2018 case law retrieval task. The phrase scoring model is trained with loss coefficients (i).

Length Threshold t	**ROUGE-1**			**ROUGE-2**			**ROUGE-SU6**		
	Pre	**Rec**	**F1**	**Pre**	**Rec**	**F1**	**Pre**	**Rec**	**F1**
10%	0.482	0.409	0.405	0.186	0.152	0.152	0.258	0.199	0.167
20%	0.377	0.592	**0.424**	0.155	0.244	**0.174**	0.169	0.388	**0.184**
30%	0.304	0.687	0.390	0.135	0.311	**0.174**	0.116	0.511	0.155
40%	0.253	0.745	0.352	0.121	0.364	0.169	0.084	0.592	0.125
50%	0.216	0.784	0.318	0.109	0.407	0.162	0.063	0.651	0.100

The phrase score statistics are shown in Fig. 4.2.2 and Fig. 4.2.3. Most of the sample cases, positive summaries have higher mean scores than document contents, and document contents have higher mean scores than negative summaries.

We measure the score distribution but over all data. Similar to per document, positive summaries have higher mean scores than document contents, and document contents have higher mean scores than negative summaries. High-score document contents are selected from top 50 highest phrases for each document. The phrases in high-score document contents affects much to the composition of document vectors, and could also be selected for summarizing documents.

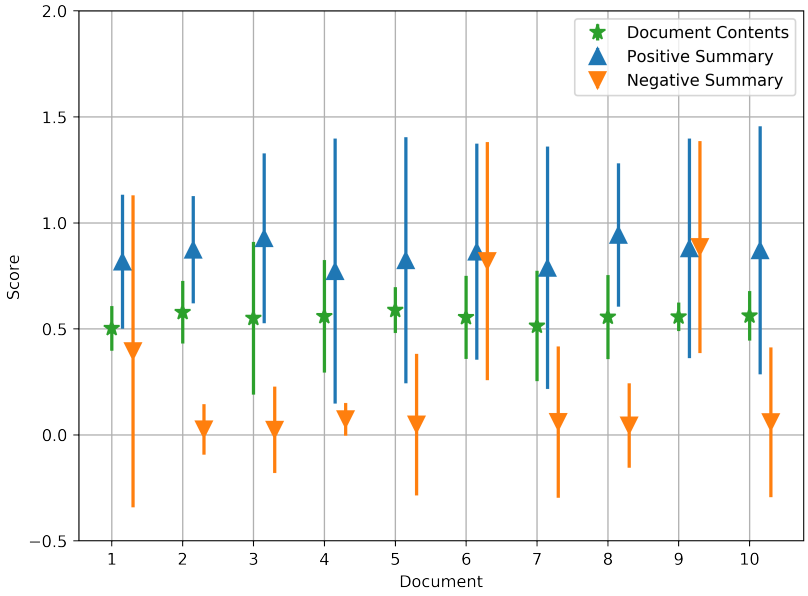


Figure 4.2.2: Visualization of score distribution (95% confidence) per document showing the comparison among scores of a document's contents with its summary (positive summary) and other random document's summary (negative summary). Most of the sample cases, positive summaries have higher mean scores than document contents, and document contents have higher mean scores than negative summaries.

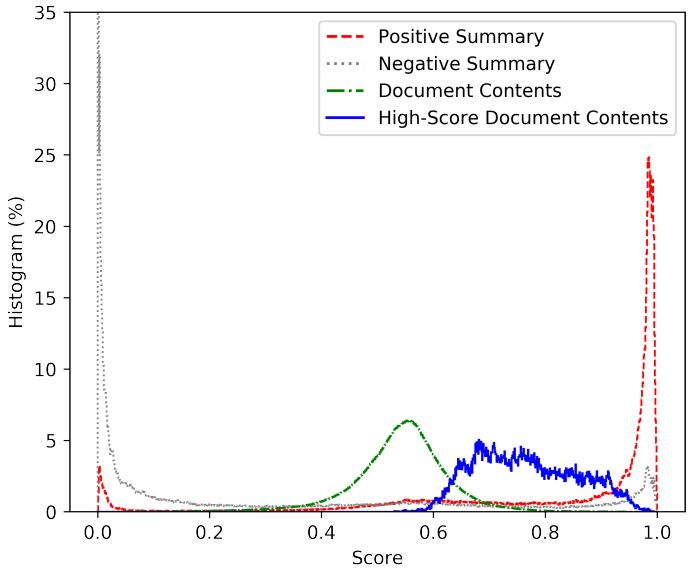


Figure 4.2.3: Visualization of score distribution over all data. Positive summaries have higher mean scores than document contents, and document contents have higher mean scores than negative summaries. High-score document contents are selected from top 50 highest phrases for each document. The phrases in high-score document contents affects much to the composition of document vectors, and could also be selected for summarizing documents.

4.2.2.2 Summary Generation by Sentence Selection

We evaluate on the dataset provided by Hachey et al. [10], which is a collection of 47 judgments of the House of Lord[2] (HOLJ) from 2001 to 2003. We compare the results of sentence selection with the methods of Hachey et al. [10] and Kim et al. [43].

- Hachey et al.: develop a sentence classification method using models trained on several labor linguistic features: cue phrase, location, entities, sentence length, quotations, and thematic words.
- Kim et al.: develop a graph-based algorithm which selects sentences towards the conclusion/decision of the case. The sentences are connected based on the embedding probability, the probability that a sentence is embedded in another.

Given a document $d = s$, we select top t sentences with highest scores computed by Equation 4.14. As we compute the sentence score as sum of its phrase scores, this obviously favors the long sentences with many high score phrases.

Since there are only 47 documents in HOLJ corpus, the phrase scoring model is trained on COLIEE 2018 corpus which is a collection of case documents from a database of predominantly Federal Court of Canada case laws, provided by Compass Law.

$$Score(s) = \sum_{p \in s} PS(p, s, d) \text{ where } s \in d \tag{4.14}$$

[2]https://www.parliament.uk/business/lords/

Table 4.2.3: Sentence selection results by selection F-score

Top t Sentences	**Pre**	**Rec**	**F1**
10%	0.197	0.136	0.155
20%	0.182	0.245	0.201
30%	0.168	0.344	0.219
40%	0.168	0.460	0.240
50%	0.171	0.579	0.258
Hachey et al.	0.317	0.307	0.312
Kim et al.	0.313	0.364	0.337

Table 4.2.4: Sentence selection results by ROUGE scores

Top t Sentences	**ROUGE-1**			**ROUGE-2**			**ROUGE-SU6**		
	Pre	**Rec**	**F1**	**Pre**	**Rec**	**F1**	**Pre**	**Rec**	**F1**
10%	0.523	0.715	0.583	0.313	0.424	0.347	0.302	0.530	0.342
20%	0.365	0.846	0.494	0.258	0.592	0.348	0.155	0.739	0.236
30%	0.289	0.896	0.424	0.221	0.685	0.325	0.097	0.824	0.164
40%	0.247	0.931	0.380	0.205	0.770	0.315	0.071	0.882	0.126
50%	0.220	0.957	0.350	0.194	0.838	0.307	0.056	0.928	0.103

Even though, using the phrase scoring model, we can select sentences with high overlap with the gold sentences (Table 4.2.4), the accuracy of selecting the labeled sentences is low (Table 4.2.3). This is understandable since our phrase scoring model focuses on evaluating the importance of phrases, and is not directly learned to score sentences. Besides, there are two factors our phrase scoring model does not have during inference: 91) any explicit linguistic features other than word embedding, (2) statistical information: term frequency-inverse document frequency. Furthermore, the phrase scoring model is trained on a different corpus. The common of the training corpus (COLIEE 2018) with the test corpus (HOLJ) is essentially captured through the use of word embedding.

4.3 Encoded Summarization

We present our approach to obtain document representation in continuous vector space in which the summary properties of the document are embedded, which we call encoded summarization. The "encoded summarization" utilizes the phrase scoring framework with two main factors: output phrase scores and internal multi-level contextual features.

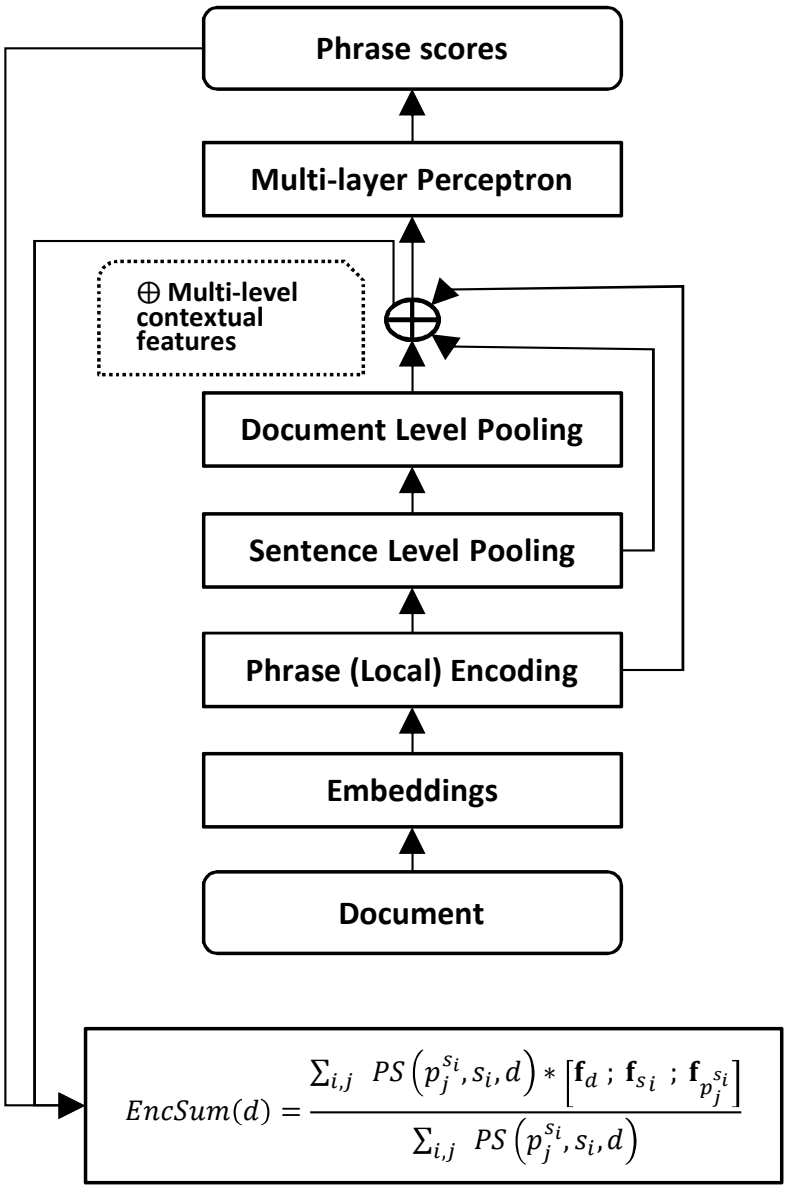


Figure 4.3.1: Encoded Summarization composition

4.3.1 Encoded Summarization Model

We present our method of composing document vectors from the phrase scoring model.

Given a document, we obtain its phrase scores and internal representations at three levels: phrase level, sentence level and document level. Then, we compose the document vector as:

$$\mathbf{g}(d) = \frac{\sum_{i,j} P\left(p_j^{s_i}, s_i, d\right) \times \left[\mathbf{f}_d; \mathbf{f}_{s_i}; \mathbf{f}_{p_j^{s_i}}\right]}{\sum_{i,j} P\left(p_j^{s_i}, s_i, d\right)} \tag{4.15}$$

Given a document, the composition weights the document contents based on their scores obtained from the phrase scoring framework. Important contents should have high contribution or affection to the final document vector. The component representations are multi-level contextual features which are the internal representations of the phrase scoring model. These internal representations contain the features which are learned to be used as base for scoring the surface contents. By using the multi-level contexts, the final document vector embeds the weighted multi-level contextual information including phrase level and sentence level contexts.

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

4.3.2 Experiments on Legal Case Retrieval Task

We describe the application of the Encoded Summarization method to building legal case retrieval systems in Chapter 5.

4.4 Chapter Summary

We have introduced encoded summarization: encoding document based on the contents we deem important via the phrase scoring mechanism. The experiments show the effectiveness of the phrase scoring mechanism, the core of encoded summarization, in modeling the target which, in this context, is the draft summary of each given legal case document. In the next chapter, we will present our approach of building legal case retrieval systems based on the study of structural encoding in Chapter 2 and document encoding target developed in this chapter.

Chapter 5

Legal Case Retrieval Systems

5.1 Introduction

In this chapter, we showcase our retrieval systems built from various implementations of structural encoding and encoded summarization methods.

We also explore the benefits of employing various types of similarity measurement belonging to lexical similarity (keyword matching) and semantic similarity (meaning matching).

On one hand, the lexical similarity and semantic similarity differ from each other and can potentially complement each other as well. The lexical similarity is obtained with approaches where the texts are compared by the direct surface forms with probably some transformations such as stemming, lemmatization, stopword removal, etc. High lexical similarity can present high matching, but low lexical similarity does not say much.

On the other hand, semantic similarity can provide the measurement where the surface forms are mismatched, for example, by paraphrasing. Semantic similarity can be learned in unsupervised fashion where common approaches are using statistical methods and benefits from huge available corpora (e.g. Wikipedia, GoogleNews, etc.) [18, 19, 15, 16]. Those methods treat a document as bag/sequence of words equally. Other information in the documents such as important words or phrases, or the document hierarchy when considered may provide significant information.

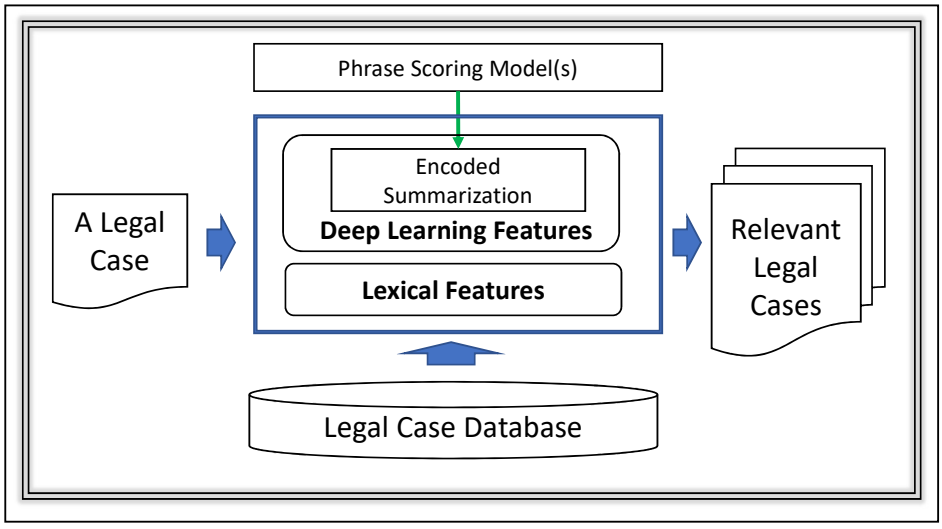


Figure 5.1.1: Legal Case Retrieval System Architecture

SUMMARY:
A human rights complaint alleged the federal government's underfunding of welfare services for on-reserve First Nations children resulted in a lower level of services for those children than for other Canadian children whose welfare services were provincially funded. /* ... */
The Federal Court held that, while the Tribunal had the power to decide this issue in advance of a full hearing on the merits, the process followed was not fair. /* ... */
Administrative Law - Topic 547
The hearing and decision - Decisions of the tribunal - Reasons for decision - When required - [See second Civil Rights - Topic 7046].
Administrative Law - Topic 2608
Natural justice - Evidence and proof - Extraneous or irrelevant considerations - [See first Civil Rights - Topic 7046].
/* ... */
PARAGRAPHS:
[**1**] Mactavish, J. : The Government of Canada funds child welfare services for First Nations children living on reserves. The provinces fund child welfare services for all other Aboriginal and non-Aboriginal children.
[**2**] The First Nations Child and Family Caring Society and the Assembly of First Nations filed a human rights complaint with the Canadian Human Rights Commission in which they allege that the Government of Canada under-funds child welfare services for on-reserve First Nations children. /* ... */
/* ... */
[**254**] In my view, the ordinary meaning of the phrase "differentiate adversely in relation to any individual" on a prohibited ground of discrimination is to treat someone differently than you might otherwise have done because of the individual's membership in a protected group. /* ... */
/* ... */
[**395**] As a result, the three applications for judicial review are granted.
/* ... */
[**396**] THIS COURT ORDERS AND ADJUDGES that /* ... */

Figure 5.1.2: Illustration of a legal case document from Federal Court of Canada. "/* ... */": omitted contents. Other information about citing cases, noticed cases, notices statutes, etc. are omitted.

In the next sections, we present our method for building legal case retrieval systems by exploring the benefits from combining lexical features and deep learning features generated with neural networks. Our experiments show that lexical features and deep learning features complement each other to improve the retrieval system performance. Furthermore, our experimental results suggest the importance of case summarization in different aspects: using provided summaries and performing encoded summarization.

5.2 Lexical Features

We estimate the lexical features by performing lexical matching between a query and a candidate case in different types of measure formulas and comparison aspects. The lexical similarity is obtained with approaches where the texts are compared by the direct surface forms with probably some transformations such as stemming, lemmatization, stopword removal, etc. High lexical similarity can present high matching, but low lexical

similarity does not say much.

We estimate the lexical features by performing lexical matching between a query and a candidate case in different types of n-grams, skip-grams, longest common subsequence to measure various degrees of lexical similarity.

- 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 non-continuous 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 two variants: standard version and distance-weighted version. The distance-weighted version favors subsequences with shorter 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[1].

To have more precise comparison between a query and a candidate, we apply the following 4 matching options:

- Summary-Summary: we compute the matching of the query's summary with the candidate's summary. This matching represents the comparison of the highlights between the query and the candidate.
- Paragraphs-Summary: we compute the matching of the query's paragraphs with the candidate's summary. This matching represents the ratio of the candidate summary mentioning relevant details.
- Summary-Paragraphs: we compute the matching of the query's summary with the candidate's paragraphs. This matching represents the ratio of the query's highlights mentioned in the candidate's details.

[1] `https://github.com/andersjo/pyrouge`

- Paragraphs-Paragraphs: we compute the matching of the query's paragraphs with the candidate's paragraphs. This matching represents the ratio of the query's details also occurred in the candidate's details.

For COLIEE 2019 dataset, since most of the candidate cases do not have a summary, we perform summary generation in two ways: using the lead sentence of each paragraph and the generated summary described in Section 4.2.2. This results in 6 matching options for COLIEE 2019 dataset.

The coding for lexical features is in the form of q-c described as follows.

- q is a subset of query components including its expert summary (s) and paragraphs (p).
- c is a subset of candidate components including its expert summary (s) and paragraphs (p). As the case of COLIEE 2019 dataset, we use the lead sentences (l) and the generated summary (e) instead of unavailable expert summary (s).
- Each component of q is compared with each component of c.

For example, the lexical method sp-sp (q=sp, c=sp) means we perform 4 matching options: Summary-Summary, Summary-Paragraphs, Paragraphs-Summary, Paragraphs-Paragraphs, and the lexical method s-p (q=s, c=p) means we only perform Summary-Paragraphs matching. We use this naming for presenting lexical features' impact analysis in our experiments.

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

In total, we collect lexical features from 6 matching formulas and 3 matching factors and 4 matching options, which results in 72 lexical features for measuring lexical matching between a query and each of its candidates. For COLIEE 2019 dataset, since most of the candidate cases do not have a summary, we perform summary generation in two ways: using the lead sentence of each paragraph and the generated summary described in Section 4.2.2. with the two additional matching options, we obtain 108 lexical features for COLIEE 2019 dataset.

5.3 Deep Learning Features

Semantic similarity can provide the measurement where the surface forms are mismatched, for example, by paraphrasing. Semantic similarity can be learned in unsupervised fashion where common approaches are using statistical methods and benefits from huge available. We utilize common methods for encoding documents based on *word-embedding* and *doc2vec*. Furthermore, we propose to use summaries and lead sentences as objectives to encode documents into continuous vector space.

We utilizes several approaches for encoding documents into continuous vector space as follows.

- *word-embedding*: From word vectors, we apply three kinds of vector compositions for producing document vectors: max-pooling, average-pooling, hierarchical-pooling. The word vectors can be obtained from word embedding models, for example, Google *word2vec* [15] or *GloVe* [16].

[2] `https://github.com/andersjo/pyrouge`

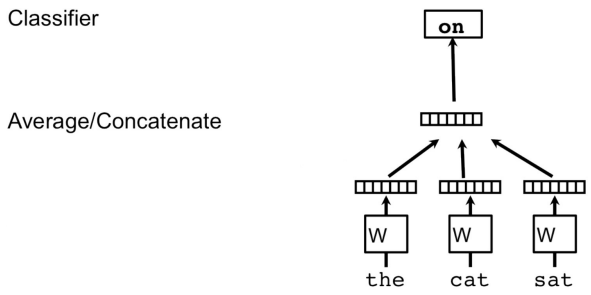


Figure 5.3.1: *word2vec* model architecture

- *doc2vec*[18]: This is a method for mapping text blocks into vector space. The method considers texts as sequences of tokens regardless of presented structures.
- Encoded summarization: We apply our method described in Section 4.3 with various implementations including:
  - Local contexts via seqential n-gram
  - Local contexts via tree pq-gram
  - Rhetoric information embedding
  - Phrase scoring with gold summaries
  - Phrase scoring with lead sentences

5.3.1 *word-embedding* and *doc2vec* Based Models

5.3.1.1 *word-embeddings* (WordEmb):

We employ simple document vector composition from the content word vectors. We devise 3 composition methods: average-pooling, max-pooling and hierarchical max-pooling.

Average-pooling: average of word vectors of all words w of document d.

$$\mathbf{g}(d) = \frac{\sum_w WordEmb(w)}{|\{w\}|} \tag{5.1}$$

This very simple composition averages out the document content equally among all words which results in keeping values of dimensions having mostly either low or high values and moderate values of high variant dimensions. In other words, common representative dimensions of the composition have either high or low values.

Max-pooling: maximum over each dimension of word vectors of all words w of document d.

$$\mathbf{g}(d) = \text{max-pooling}_w(WordEmb(w)) \tag{5.2}$$

This composition, different from average-pooling, takes out the highest value for each dimension, which represents the highest existence of such dimension among all words.

Hierarchical-pooling: In addition to capturing averaging and maxing features from word to document, intermediate sentence level is considered to add sentence boundary

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

$$\mathbf{f}_{s_i} = \text{max-pooling}_j(WordEmb(w_j^{s_i})) \tag{5.3}$$

where max-pooling are operated over each dimension of vectors $WordEmb(w_j^{s_i})$. With the same max-pooling operation as above, we compute document-level features as:

$$\mathbf{f}_d = \text{max-pooling}_w(WordEmb(w)) \tag{5.4}$$

The document vector is finally composed of word-level, sentence-level and document-level features.

$$\mathbf{g}(d) = \left[\mathbf{f}_d \; ; \; \frac{\sum_i \mathbf{f}_{s_i}}{|\{s_i\}|} \; ; \; \frac{\sum_w WordEmb(w)}{|\{w\}|}\right] \tag{5.5}$$

This composition includes the same information as average-pooling and max-pooling through word-level and document-level features, then further expands with sentence-level pooling which is dependent on sentence boundary.

5.3.1.2 *doc2vec*:

We utilize *doc2vec* model to estimate the document vectors $\mathbf{g}(d)$. The *doc2vec* vector of a document embeds information to predict each of its words given each word's surround contexts (as illustrated in Figure 5.3.2). Thus, *doc2vec* vectors embed content predictive features of documents.

$$\mathbf{g}(d) = doc2vec(d) \tag{5.6}$$

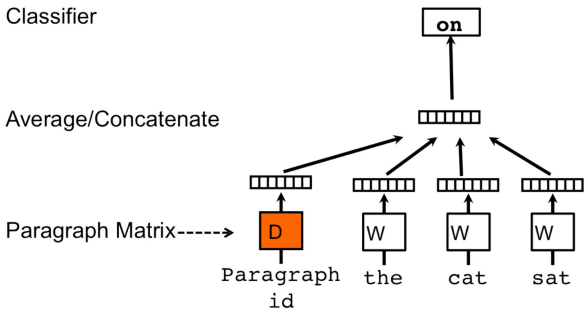


Figure 5.3.2: *doc2vec* model architecture

5.3.2 Encoded Summarization (EncSum):

We apply the composition described in Section 4.3 to encode a legal case document into a vector space from which we can compare the similarity between a case query and a case candidate.

In COLIEE 2018, when dealing with this task, Tran et al. [44] observed several obstacles. First, the candidate cases are ≈5.7K-token long in average (Table 5.3.1). 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 5.3.1: Statistics of candidate case documents in COLIEE 2018 training data.

Property	**Max**	**Avg.**
#words/doc	85,551	5,690
#paragraphs/doc	1,117	43
#lead-sentences-words/doc	35,942	1,187
#summary-words/doc	8,827	589

Table 5.3.2: 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 5.3.2). 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.

We apply various implementations of the method with various text encoding methods and learning targets. The encoding methods are: sequential n-gram context encoding, tree pq-gram context encoding, rhetoric embedding. The learning targets are gold summaries and lead sentences. The base model is implemented with n-gram and gold summaries without rhetoric embedding.

5.3.2.1 Sequential n-Gram Context Encoding

Sequential n-gram context encoding treats the input texts as a set of n contiguous words. This method assumes the temporal property of text which is read sequentially, but ignores any latent structures of the text.

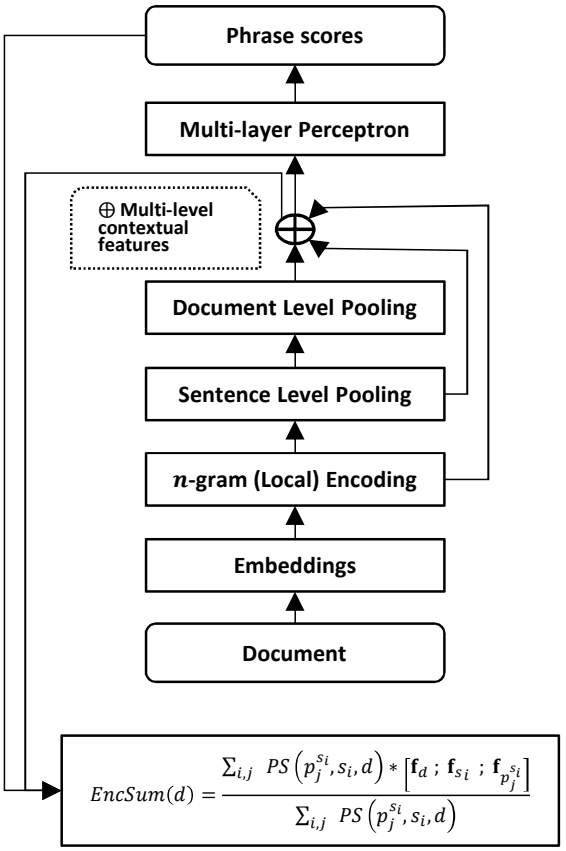


Figure 5.3.3: Encoded Summarization composition with n-gram local contexts

5.3.2.2 Dependency Tree *pq*-Gram Context Encoding

Tree *pq*-gram context encoding as mentioned in Section 2.3 demonstrates the method of capturing syntactic dependency of words in a sentence. *pq*-gram are more explicit than *n*-grams in term of expressing a local context since *pq*-grams are more interpretable than *n*-grams to human. Furthermore, a *pq*-gram context can contain words that are located discontiguously.

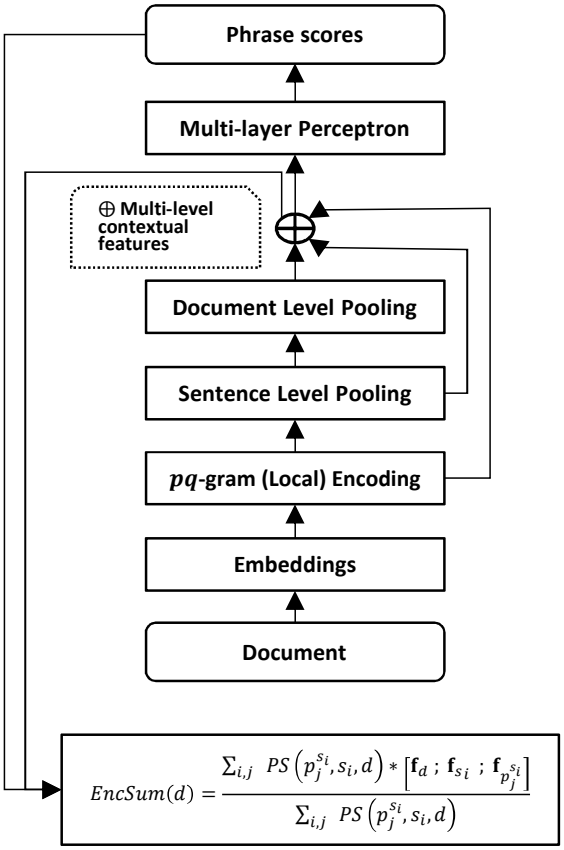


Figure 5.3.4: Encoded Summarization composition with *pq*-gram local contexts

5.3.2.3 Sentence Rhetorical Embedding

The importance of a phrase can be assessed with the rhetorical status of the enclosing sentence. Not only the phrase contains similar contexts found in the summary but also the phrase is used in the document sentence having the same rhetorical status as the summary sentence contains such contexts.

We add rhetoric information to a phrase as an embedding vector of prediction probability of the phrase's enclosing sentence using the rhetoric classification system in Chapter 3.

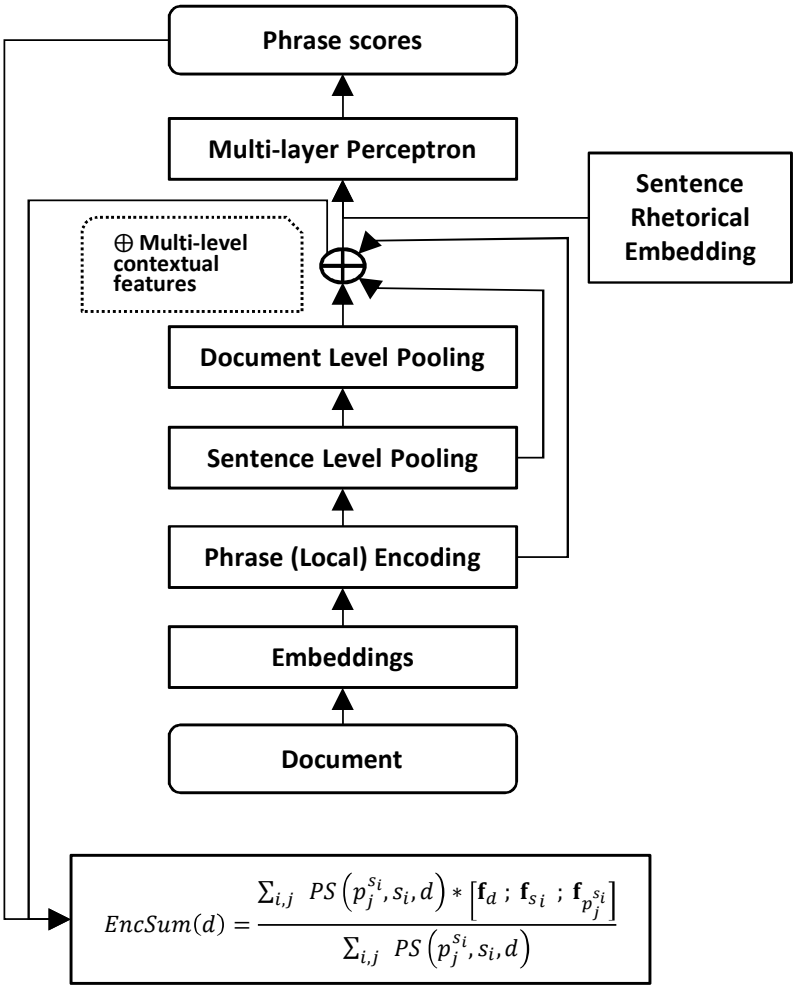


Figure 5.3.5: Encoded Summarization composition with rhetorical embedding

5.3.2.4 Phrase Scoring with Gold Summaries

This is straightforward as we use expert drafted summaries as the target for training phrase scoring models.

5.3.2.5 Phrase Scoring with Lead Sentences

We think of alternative solutions for obtaining phrase scoring models when gold summaries are not available. One is to use lead sentences as a kind of heuristic summaries.

Lead sentences are usually topic sentences stating the points to be discussed in the rest of the containing paragraphs, which makes lead sentences potential for performing content condensation. With this in mind, we apply various text encoding methods on the lead sentences to obtain a representation for the corresponding document. Furthermore, we build a document encoding model based on the phrase scoring framework guided by lead sentences. This method has the advantage of applicability to any documents, thus, can be used to obtain one kind of document embeddings with the property of lead sentences.

Legal case documents usually contain huge amount of contents. As in Table 5.3.1, a legal case document contains 5.6K words and 45 paragraphs in average, and could goes over 80K words and 1K paragraphs. This challenges the efficiency of not only human experts but also automatic retrieval systems. Editor summarization condensates contents by ≈90% which results in ≈10% key contents. The ≈10% key contents, however, require legal experts, thus, are not always available.

Lead sentences can be used to summarize topics discussed throughout the documents. For each paragraph, lead sentences are usually important sentences stating the points to be discussed by the later sentences. Lead sentences may contain more key information than summary. While the summary only contains decisive points, lead sentences include almost every points discussed throughout the case, which cover ≈20% of document contents (Table 5.3.1). However, some keywords drafted by editors are noted down in the summary and may not be available in the case details as we measured that 87% of summary content words are found in paragraphs. In Table 5.3.3, ROUGE-1 scores show that there is less than half of content words shared between lead sentences and summary where 30.5% words in the lead sentences appear in the summary and 49.6% words in the summary appear in the lead sentences. By using lead sentences instead of editor drafted summaries, we take into account some trade-offs:

- Missing out the 50.4% of summary contents with editor drafted keywords.
- Gaining 69.5% of lead sentence contents not in the summary.
- Ease of obtaining lead sentences versus not always available manually drafted summary.
- Still achieving the purpose of content condensation for building a legal case retrieval system.

Table 5.3.3: Overlapping between lead sentences and summaries of legal case documents from COLIEE2018 training data. Stopword removal and stemming are performed before computing ROUGE scores. "Lead in Sum": how much of lead sentences' contents appear in summary. The otherwise is for "Sum in Lead".

Measure	Lead in Sum	Sum in Lead
ROUGE-1	0.305	0.496
ROUGE-2	0.115	0.185
ROUGE-L	0.283	0.460
ROUGE-W	0.164	0.132
ROUGE-S	0.107	0.272
ROUGE-SU	0.108	0.275

5.3.3 Query-Candidate Relevance Vector

The relevance vector consists of the features indicating the relevance of a candidate given a query. We compose this vector from lexical features and deep learning features.

The lexical features are computed by lexical matching which by themselves present the relevance measurement.

For the deep learning features which are encoded information in continuous vector space, by comparing each dimension independently, we can estimate the compatibility of a query and a candidate over the dimension. Thus, we compute the relevance features from deep learning features as the element-wise product of query vector and candidate vector. First, we obtain query vector $\mathbf{g}(q)$ and candidate vector $\mathbf{g}(c)$ for each of the 5 above document vector compositions (Equations 4.15, 5.1, 5.2, 5.5, and 5.6). Then, we compute the relevance vector of query q and candidate c by the following element-wise product.

$$\mathbf{h}(q, c) = \mathbf{g}(q) \circ \mathbf{g}(c) \tag{5.7}$$

The combination of lexical features and deep learning features is presented in the query-candidate relevance vector as the concatenation of lexical matching features and the element-wise product of deep learning feature vectors of the query and the candidate.

$$\text{relevance-vector}(q, c) = [\text{lexical-features}(q, c); \mathbf{h}(q, c)] \tag{5.8}$$

5.4 The Retrieval Systems

From the lexical features and deep learning features, we can customize our own legal case retrieval system. We present our experiments with the following customization:

- Lexical: the retrieval system build from only lexical features (Section 5.2).
- *WordEmb*-Avg-pooling: average pooling of word embeddings (Equation 5.1).
- *WordEmb*-Max-pooling: max pooling of word embeddings (Equation 5.2).
- *WordEmb*-Hierarchical-pooling: hierarchical pooling of word embeddings (Equation 5.5).
- *doc2vec*: vanilla doc2vec model (Equation 5.6).

- EncSum: base Encoded Summarization whose phrase scoring model is trained with n-gram context encoding.
- EncSum-p=#,q=#: Encoded Summarization whose phrase scoring model is trained with pq-gram context encoding.
- EncSum-rhetoric-emb: Encoded Summarization whose phrase scoring model is trained with incorporation of sentence rhetorical information.
- EncSum-lead: Encoded Summarization whose phrase scoring model is trained with lead sentences instead of gold summaries.
- Lexical+###: combination of lexical with the above deep learning features. We will show in the experiments that the combination yield potential improvement.
- Voting: Combining potential models. The voting method could benefit form these models' divergence in modeling query-candidate relationship.

5.5 Experiments on Legal Case Retrieval Task

In the data used in our experiments, the legal cases are sampled from a database of predominantly Federal Court of Canada case laws, provided by Compass Law. The data are provided by COLIEE competition [45] held in two years 2018 and 2019. In each of both the datasets, the data contain 285 queries, each query is attached with 200 candidate cases. Each candidate case is presented as a raw text document file which describes the details of the case. While a summary is presented in the query case, the candidate cases may not have summary section.

We formulate the task as bipartite ranking problem and devise the learning to ranking method to solve it. We utilize pair-wise ranking strategy: pairing each noticed case with an irrelevant case from the candidate list. We adopt Linear-SVM as the learning algorithm for solving the optimization problem. The input of the learning-to-rank algorithm is the query-candidate relevance vectors obtained from Equation 5.8 in Section 5.3.3. After obtaining the scored candidates as a ranked list, we proceed to select top k highest scored candidates as the predicted noticed cases.

The phrase scoring model was trained on only 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 4.2.2) is set to $t = 20\%$ document-length. As shown in Tables 5.3.1 and 5.3.2, the average length of summaries is $\approx 10\%$ document-length for COLIEE 2018 dataset, and $\approx 9\%$ document-length for COLIEE 2019 dataset. Thus, with a threshold $t = 20\%$ document-length, we could expect to cover potential information with good recall rate ($\approx 70\%$) while keeping an acceptable summary length. The parameter setting of the base phrase scoring model is described in Table 2.3.4. For the rhetorical embedding, we use the model CNN(1000)-BiLSTM(1000)-FC from Chapter 3.

We evaluated our approach by performing leave-one-out validation where we tested on each and every query from the provided 285 queries and the rest as training data.

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

- MAP: Mean average precision.

- P, R, F1: Precision, Recall, f-measure whose values are averaged by query. This is straightforward as we average the results of all folds in the leave-one-out validation.

The results in validation phase, test phase and compared with other COLIEE participants (Tables 5.5.1, 5.5.2, 5.5.9, 5.5.10, 5.5.11, and 5.5.12) show that **Lexical+EncSum**, the combination of lexical features with encoded summarization, achieves the best performance.

5.5.1 Validation Results

5.5.1.1 Overall

The validation results of COLIEE 2018 (Table 5.5.1) and COLIEE 2019 (Table 5.5.2) show that lexical features and deep learning features complement each other really well. The highest performance with either lexical or deep learning features is lower than the lowest performance of the combination. The improvement by the combination hints the existence of important information captured by deep learning features but not captured by lexical features.

WordEmb-Hierarchical-pooling performs better than *WordEmb*-Max-pooling and *WordEmb*-Avg-pooling. The hierarchical pooling consists of *WordEmb*-Max-pooling, *WordEmb*-Avg-pooling features and further sentence-level pooling which regards the sentence information boundary.

As shown in Table 5.5.1, when limiting the document to only the summary part than the whole content, most of the models using *WordEmb* or *doc2vec* perform better, except *doc2vec* without lexical features. This suggests the important of summarization in legal case retrieval task.

The suggestion strongly presents in the results of the models using encoded summarization. The models with encoded summarization features outperforms other latent feature generation candidates including *WordEmb*, *doc2vec* on either the summary part or the whole document. Furthermore, the improvement of the encoded summarization suggests that this feature type not only embeds the summary properties of the document but also carries selectively important information from the document content.

The above points also suggests that the summary of a case contains important information but may not contain all relevant information for case retrieval. This is intuitively seen as that the whole case may discuss various legal points besides the main points. Since the encoded summarization weights the case content based on the summary which contains the main points of the case, the other various legal points which are potentially related to the main points may be captured. Hence, the selectively carried information by the encoded summarization could be the related points to the main points of the case.

Table 5.5.1: Validation results on COLIEE 2018 dataset. We select top 10 highest scored candidates when measuring precision, recall and f-measure. "(summary)" indicates that the corresponding encoding method is applied only on the summary part of the document.

Model	**MAP**	**P**	**R**	**F1**
Lexical	0.530	0.420	0.520	0.398
WordEmb-Avg-pooling	0.452	0.386	0.440	0.356
WordEmb-Max-pooling	0.325	0.306	0.326	0.275
WordEmb-Hierarchical-pooling	0.528	0.434	0.481	0.400
doc2vec	0.552	0.438	0.533	0.415
WordEmb-Avg-pooling (summary)	0.515	0.444	0.499	0.410
WordEmb-Max-pooling (summary)	0.400	0.370	0.362	0.324
WordEmb-Hierarchical-pooling (summary)	0.619	0.503	0.570	0.469
doc2vec (summary)	0.422	0.367	0.407	0.334
EncSum(i)	0.659	0.510	0.584	0.478
EncSum(ii)	**0.690**	**0.529**	**0.608**	**0.494**
EncSum-$p = 1, q = 3$ (ii)	0.626	0.500	0.576	0.466
EncSum-rhetoric-emb (ii)	0.650	0.498	0.588	0.470
EncSum-lead (ii)	0.541	0.450	0.511	0.417
Lexical+ *WordEmb*-Avg-pooling	0.686	0.522	0.653	0.502
Lexical+ *WordEmb*-Max-pooling	0.687	0.515	0.642	0.494
Lexical+ *WordEmb*-Hierarchical-pooling	0.772	0.565	0.705	0.545
Lexical+*doc2vec*	0.684	0.518	0.644	0.496
Lexical+ *WordEmb*-Avg-pooling (summary)	0.688	0.528	0.646	0.505
Lexical+ *WordEmb*-Max-pooling (summary)	0.711	0.544	0.677	0.524
Lexical+ *WordEmb*-Hierarchical-pooling (summary)	0.783	0.579	0.725	0.560
Lexical+*doc2vec*(summary)	0.704	0.539	0.675	0.520
Lexical+EncSum(i)	0.849	0.601	0.761	0.583
Lexical+EncSum(ii)	**0.888**	**0.623**	**0.788**	**0.607**
Lexical+EncSum-$p = 1, q = 3$ (ii)	0.842	0.602	0.762	0.584
Lexical+EncSum-rhetoric-emb (ii)	0.854	0.606	0.768	0.588
Lexical+EncSum-lead (ii)	0.794	0.591	0.744	0.572

Table 5.5.2: Validation results on COLIEE 2019 dataset. We select top 5 highest scored candidates when measuring precision, recall and f-measure.

Model	**MAP**	**P**	**R**	**F1**
Lexical	0.715	0.495	0.641	0.485
WordEmb-Avg-pooling	0.218	0.177	0.210	0.161
WordEmb-Max-pooling	0.270	0.223	0.260	0.206
WordEmb-Hierarchical-pooling	0.417	0.331	0.405	0.311
doc2vec	0.567	0.404	0.540	0.398
EncSum(i)	0.542	0.430	0.516	0.402
EncSum(ii)	0.576	0.436	0.534	0.410
Lexical+ *WordEmb*-Avg-pooling	0.733	0.508	0.658	0.496
Lexical+ *WordEmb*-Max-pooling	0.750	0.526	0.679	0.513
Lexical+ *WordEmb*-Hierarchical-pooling	0.782	0.549	0.704	0.534
Lexical+*doc2vec*	0.725	0.493	0.638	0.482
Lexical+EncSum(i)	0.792	0.552	0.700	0.533
Lexical+EncSum(ii)	**0.833**	**0.579**	**0.724**	**0.557**

The encoded summarization (EncSum) approach alone achieves MAP of 0.576 and F1 of 0.410 on COLIEE 2019 dataset, lower performance than the best lexical combination (MAP: 0.715, F1: 0.485). The effect is different from the observation in COLIEE 2018 dataset where the performance of encoded summarization (MAP of 0.690 and F1 of 0.494) is higher than lexical matching approach (MAP:0.530 F1:0.398). 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. The improvement by the combination of show that, even though the encoded summarization may not perform well alone, it still provides useful information for identifying relevant cases.

5.5.1.2 Lexical Features' Impact

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),

and, c is a subset of candidate components including:

- paragraphs (p)
- lead sentences (l)
- generated summary (e) (described in Section 4.2.2).

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.

Table 5.5.3: Lexical feature impact analysis by validation results on COLIEE 2018 dataset. We select top 10 highest scored candidates when measuring precision, recall and f-measure.

Lexical Combination	**MAP**	**P**	**R**	**F1**
s-s	0.372	0.331	0.378	0.302
s-p	0.482	0.386	0.486	0.367
p-s	0.435	0.356	0.434	0.331
p-p	0.469	0.372	0.463	0.355
sp-s	0.458	0.371	0.450	0.346
sp-p	0.510	0.403	0.506	0.384
sp-sp	**0.530**	**0.420**	**0.520**	**0.398**

Table 5.5.4: Lexical feature impact analysis by validation results on COLIEE 2019 dataset. We select top 5 highest scored candidates when measuring precision, recall and f-measure.

Lexical Combination	**MAP**	**P**	**R**	**F1**
s-p	0.690	0.484	0.620	0.470
s-l	0.589	0.420	0.528	0.405
s-e	0.561	0.401	0.517	0.390
p-p	0.680	0.476	0.601	0.461
p-l	0.619	0.443	0.563	0.429
p-e	0.588	0.413	0.534	0.402
sp-p	0.712	0.490	0.635	0.480
sp-l	0.634	0.448	0.570	0.435
sp-e	0.602	0.429	0.553	0.416
sp-pl	0.713	0.493	0.639	0.483
sp-pe	0.709	0.485	0.633	0.476
sp-ple	**0.715**	**0.495**	**0.641**	**0.485**

The validation results (Tables 5.5.3, and 5.5.4) of lexical features with various combinations (from the 4 matching options for COLIEE 2018 and 6 matching options for COLIEE 2019) described in Section 5.2 show that the combination of lexical matching options does have positive effect to improve the performance on both COLIEE 2018 and COLIEE 2019 datasets. On one hand, it is meaningful to have expert summaries for lexical matching as in COLIEE 2018, and on the other hand, pseudo/generated summaries could also help boost retrieval performance in COLIEE 2019 where candidate summaries are not available.

5.5.1.3 Various Implementations of Phrase Scoring Models

Table 5.5.5: Validation results of EncSum variants on COLIEE 2018 dataset. We select top 10 highest scored candidates when measuring precision, recall and f-measure.

Model	MAP	P	R	F1
EncSum(ii)	**0.690**	**0.529**	**0.608**	**0.494**
EncSum-$p = 1, q = 3$ (ii)	0.626	0.500	0.576	0.466
EncSum-rhetoric-emb (ii)	0.650	0.498	0.588	0.470
EncSum-lead (ii)	0.541	0.450	0.511	0.417
Lexical+EncSum(ii)	**0.888**	**0.623**	**0.788**	**0.607**
Lexical+EncSum-$p = 1, q = 3$ (ii)	0.842	0.602	0.762	0.584
Lexical+EncSum-rhetoric-emb (ii)	0.854	0.606	0.768	0.588
Lexical+EncSum-lead (ii)	0.794	0.591	0.744	0.572

Table 5.5.6: Validation results of pq-gram based models on COLIEE 2018 dataset. We select top 10 highest scored candidates when measuring precision, recall and f-measure. "(summary)" indicates that the corresponding encoding method is applied only on the summary part of the document.

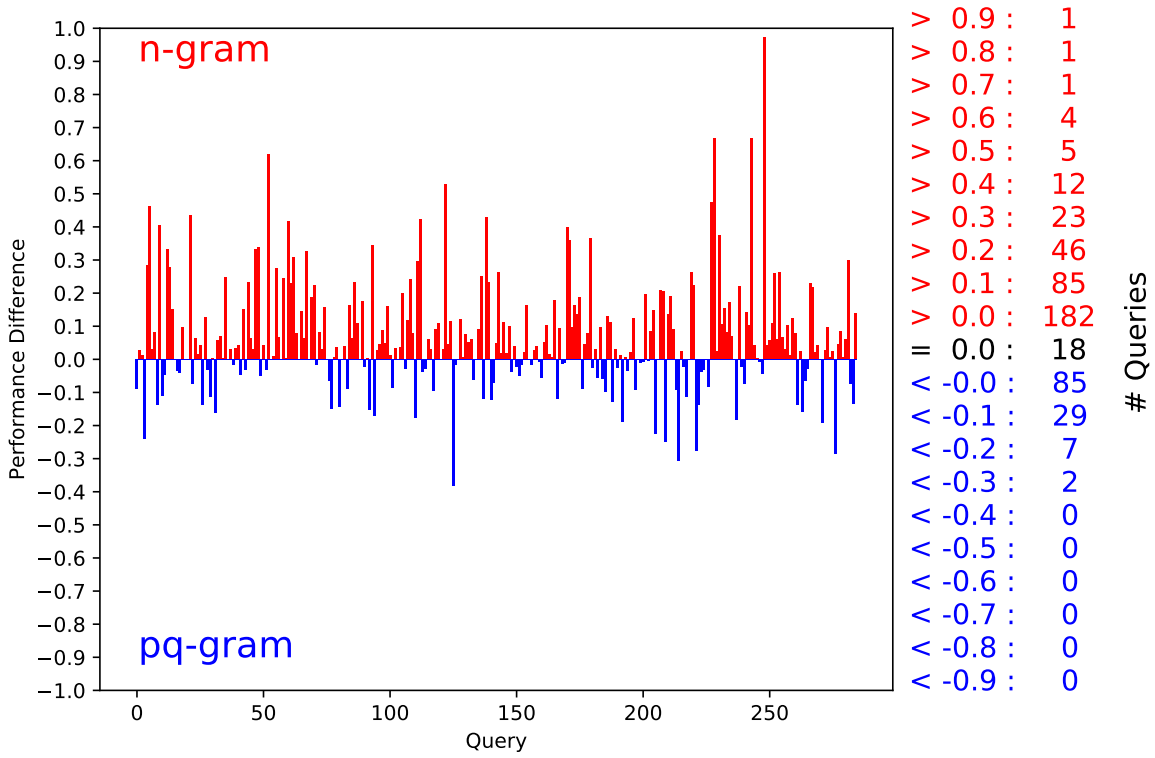
Model	MAP	P	R	F1
EncSum-$p = 1, q = 1$ (ii)	0.462	0.403	0.418	0.358
EncSum-$p = 1, q = 2$ (ii)	0.595	0.486	0.547	0.450
EncSum-$p = 1, q = 3$ (ii)	**0.626**	**0.500**	**0.576**	**0.466**
EncSum-$p = 2, q = 2$ (ii)	0.574	0.476	0.529	0.439
EncSum-$p = 3, q = 1$ (ii)	0.587	0.482	0.540	0.446
Lexical+EncSum-$p = 1, q = 1$ (ii)	0.646	0.495	0.627	0.477
Lexical+EncSum-$p = 1, q = 2$ (ii)	0.814	0.596	0.752	0.578
Lexical+EncSum-$p = 1, q = 3$ (ii)	**0.842**	**0.602**	**0.762**	**0.584**
Lexical+EncSum-$p = 2, q = 2$ (ii)	0.816	0.598	0.753	0.579
Lexical+EncSum-$p = 3, q = 1$ (ii)	0.817	0.596	0.752	0.577

pq-Gram Based Models. Performances vary among different pq values. The best performance belongs to $pq = (1, 3)$. The performances are comparable among models with pq in $(1, 2), (2, 2), (3, 1)$. The worst performance belongs to $pq = (1, 1)$.

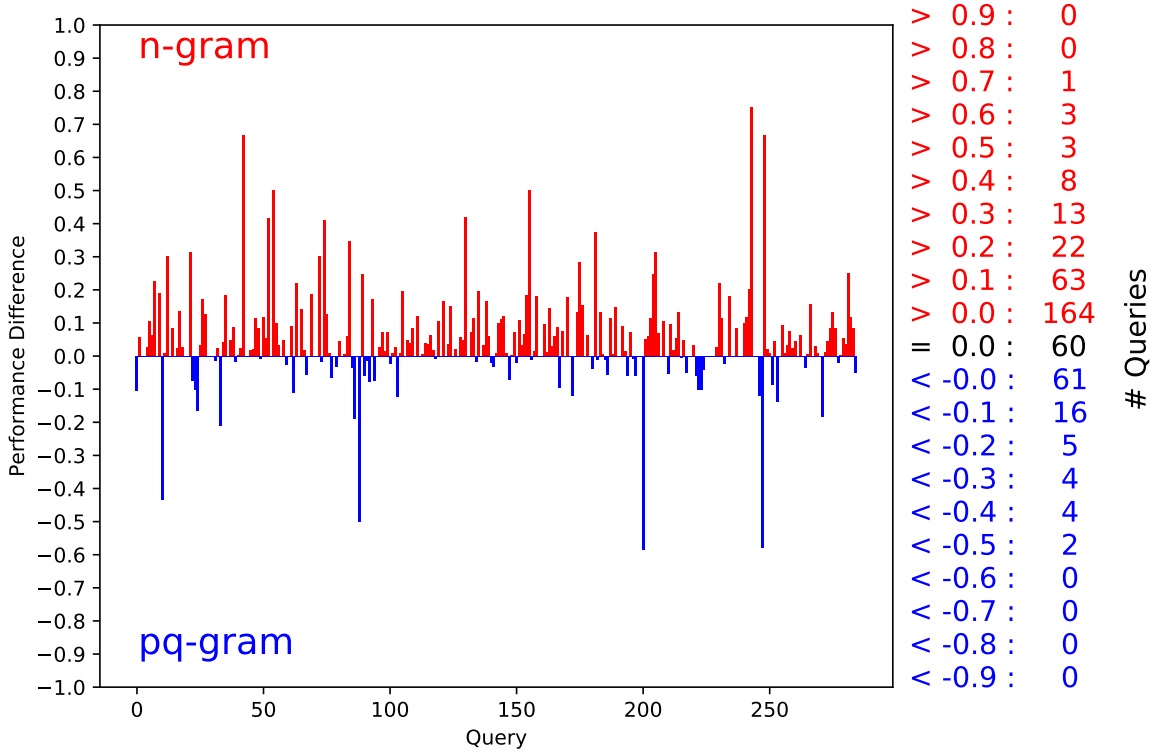
Lead Sentences Based Model. Our approach of utilizing lead sentences into building legal case retrieval system has achieved promising results. Despite of inferior performance to drafted summary based models, the use of lead sentences has the benefits of: 1) not requiring editor drafted summary, and 2) still embodying the gist of a document. The performance improvement when applying various text encoding on the editor drafted summaries showcases the benefits of using summary for retrieving relevant case documents. However, the summary of a case is not always available as editor drafted summary can require significant effort depending on the duration of the case which can be days or years whereas extracting lead sentences is trivial. On the other hand, lead

sentences are usually topic sentences which embody the gist of a document. Since lead sentences are not only used as one way to obtain a summary for a given document, but this method is also a strong baseline in related works on automatic document summarization, it suggests the possibility of applying automatic document summarization systems to obtain document summaries as the prior step to the retrieval systems.

Per Query Performance Difference. When comparing performance per query by the leave one out validation, though the overall results of using either *pq*-gram, rhetorical information or lead sentences are inferior to the base setting: *n*-gram, with gold summaries, without rhetorical information, performance differences are clearly observed (Figures 5.5.1,5.5.2, and 5.5.3). There are quite a number of cases where *pq*-gram, rhetorical information and lead sentences show superior performances. Among these variations, the model using rhetorical information shows the most significant difference. This suggests that case retrieval systems can be improved when the differences are modeled appropriately.

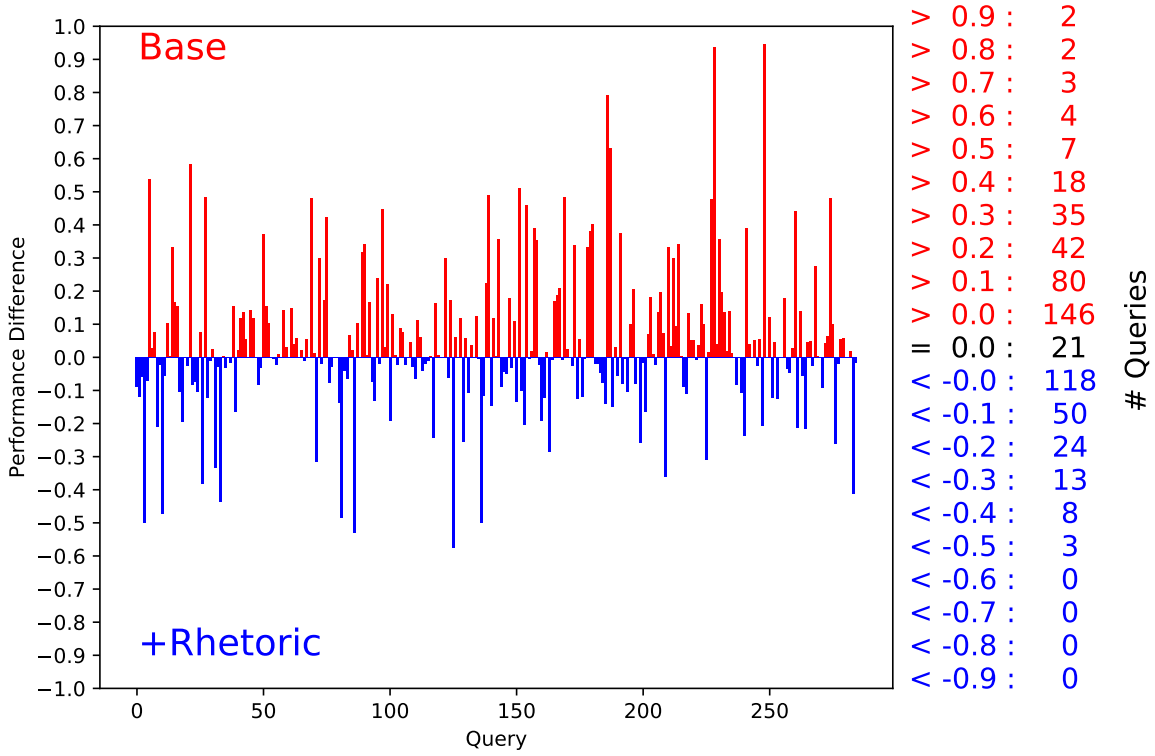


(a) Without lexical features

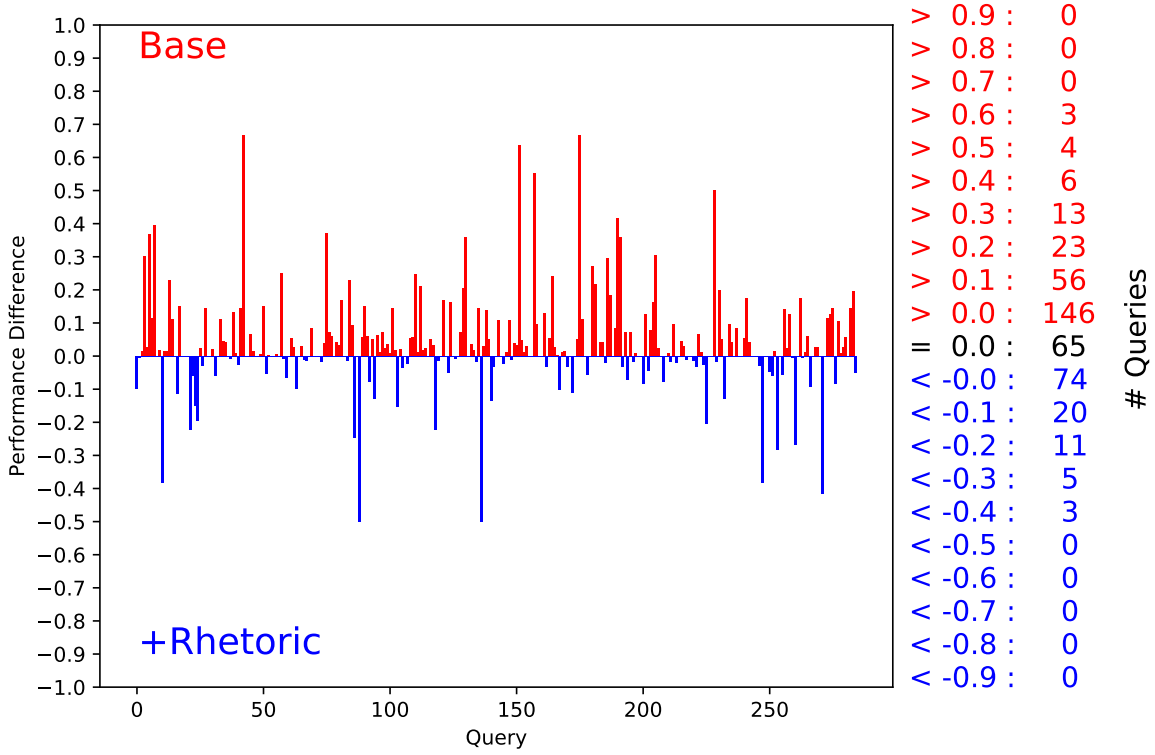


(b) With lexical features

Figure 5.5.1: Performance (MAP) difference of the model using n-gram versus the model using pq-gram. $n = 5, p = 1, q = 3$.

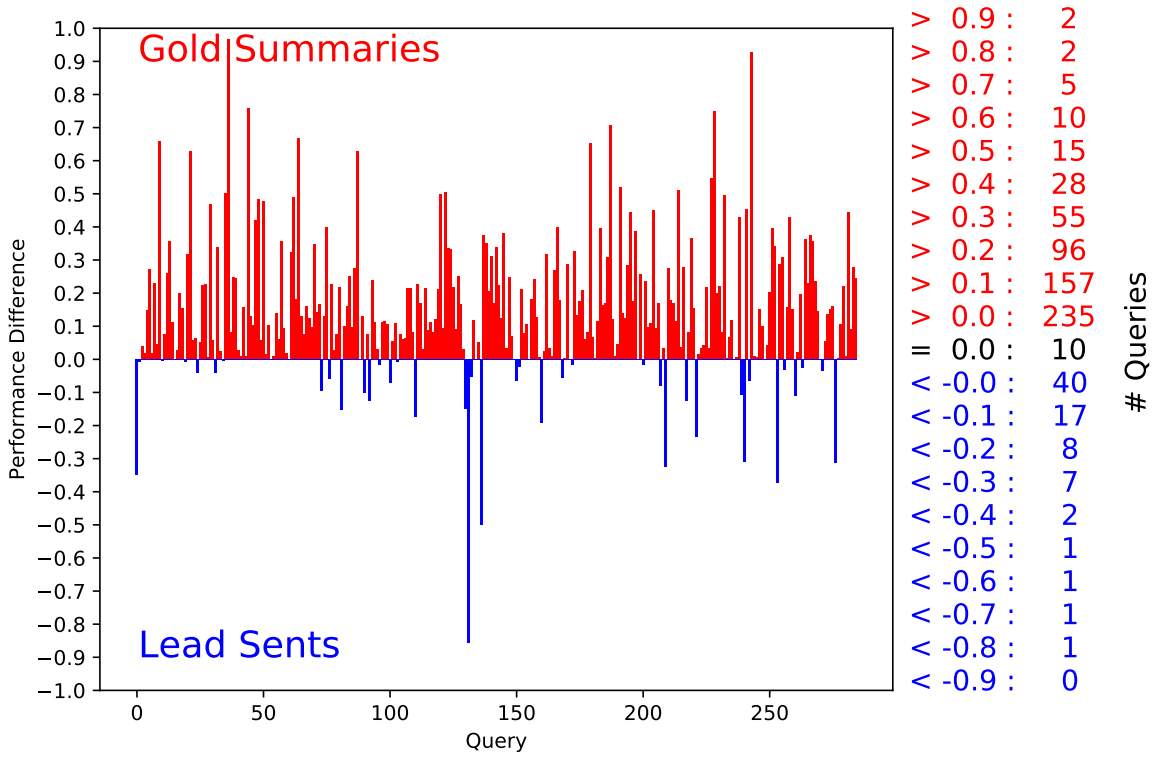


(a) Without lexical features

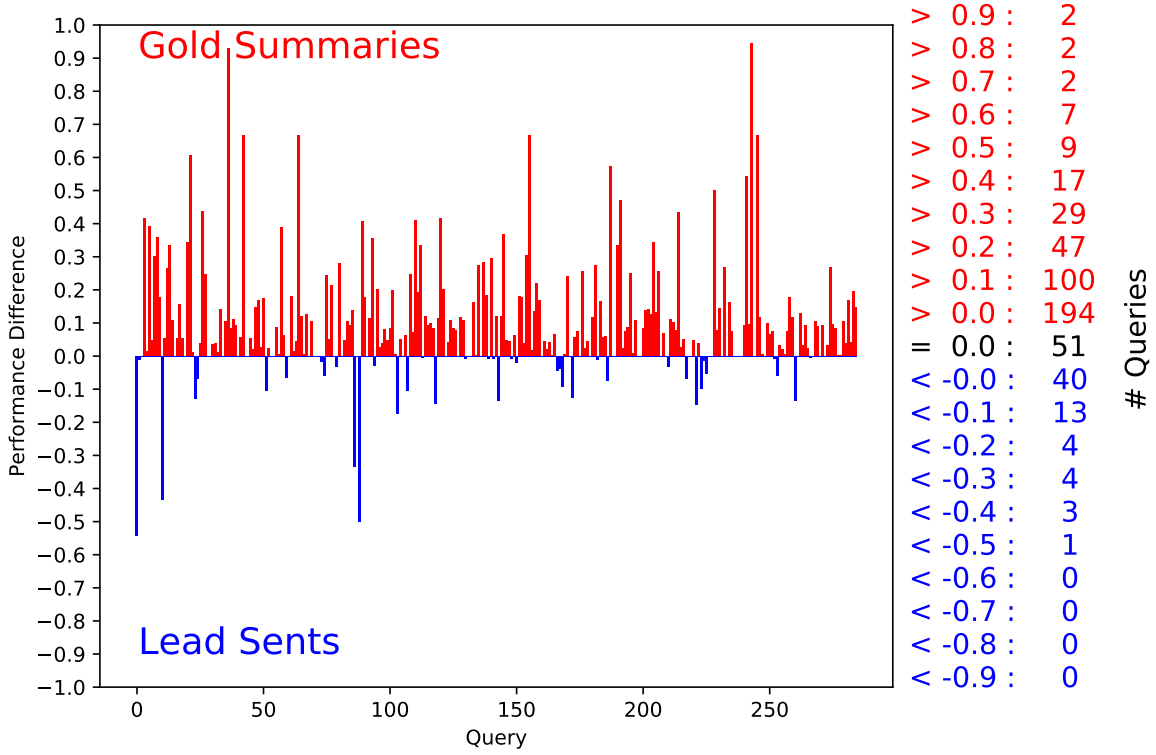


(b) With lexical features

Figure 5.5.2: Performance (MAP) difference of the base model versus the model with rhetorical embedding



(a) Without lexical features



(b) With lexical features

Figure 5.5.3: Performance (MAP) difference of the model trained with gold summaries versus the model trained with lead sentences

5.5.1.4 Voting

One way to resolve the difference modeled by each implementation of EncSum is voting. We devise a simple voting mechanism where scores output by each model are normalized with a min-max normalization based formula (Equation 5.9).

$$normalize(scores) = \frac{scores - max(scores)}{max(scores) - min_t(scores)} \tag{5.9}$$

where $min_t(scores)$ returns the score of the candidate at rank t. When t is the size of the candidate list, the normalized scores are in range [-1,0]. By subtracting the scores with a value of $max(scores)$, we equalize the rank 1 candidate of each model.

We achieve best performance voting from three models: Base, *pq*-Gram and w/Rhetoric-Emb (Table 5.5.7).

Table 5.5.7: Validation results of voting on EncSum models on COLIEE 2018 dataset. We select top 10 highest scored candidates when measuring precision, recall and f-measure. We select min_t where $t \in \{10, 20, ..., 200\}$ for score normalization (Equation 5.9). All models include lexical features, and are selected from models listed in Table 5.5.5. "||": voter separator.

Model	**MAP**	**P**	**R**	**F1**
Base	**0.888**	**0.623**	**0.788**	**0.607**
pq-Gram	0.842	0.602	0.762	0.584
w/Rhetoric-Emb	0.854	0.606	0.768	0.588
$min_{t=10}$				
Base||w/Rhetoric-Emb	0.893	0.627	0.794	0.612
Base||*pq*-Gram	0.894	0.628	0.797	0.612
w/Rhetoric-Emb||*pq*-Gram	0.889	0.624	0.792	0.609
Base||w/Rhetoric-Emb||*pq*-Gram	**0.906**	**0.635**	**0.807**	**0.621**
$min_{t=200}$				
Base||w/Rhetoric-Emb	0.898	0.631	0.800	0.615
Base||*pq*-Gram	0.893	0.628	0.796	0.612
w/Rhetoric-Emb||*pq*-Gram	0.890	0.625	0.792	0.609
Base||w/Rhetoric-Emb||*pq*-Gram	**0.906**	**0.635**	**0.806**	**0.620**
Average over min_t				
Base||w/Rhetoric-Emb	0.896 ±0.0026	0.629 ±0.0019	0.797 ±0.0030	0.613 ±0.0018
Base||*pq*-Gram	0.894 ±0.0008	0.628 ±0.0005	0.797 ±0.0015	0.613 ±0.0005
w/Rhetoric-Emb||*pq*-Gram	0.890 ±0.0013	0.625 ±0.0007	0.793 ±0.0008	0.609 ±0.0007
Base||w/Rhetoric-Emb||*pq*-Gram	**0.905** ±0.0015	**0.635** ±0.0005	**0.807** ±0.0006	**0.620** ±0.0005

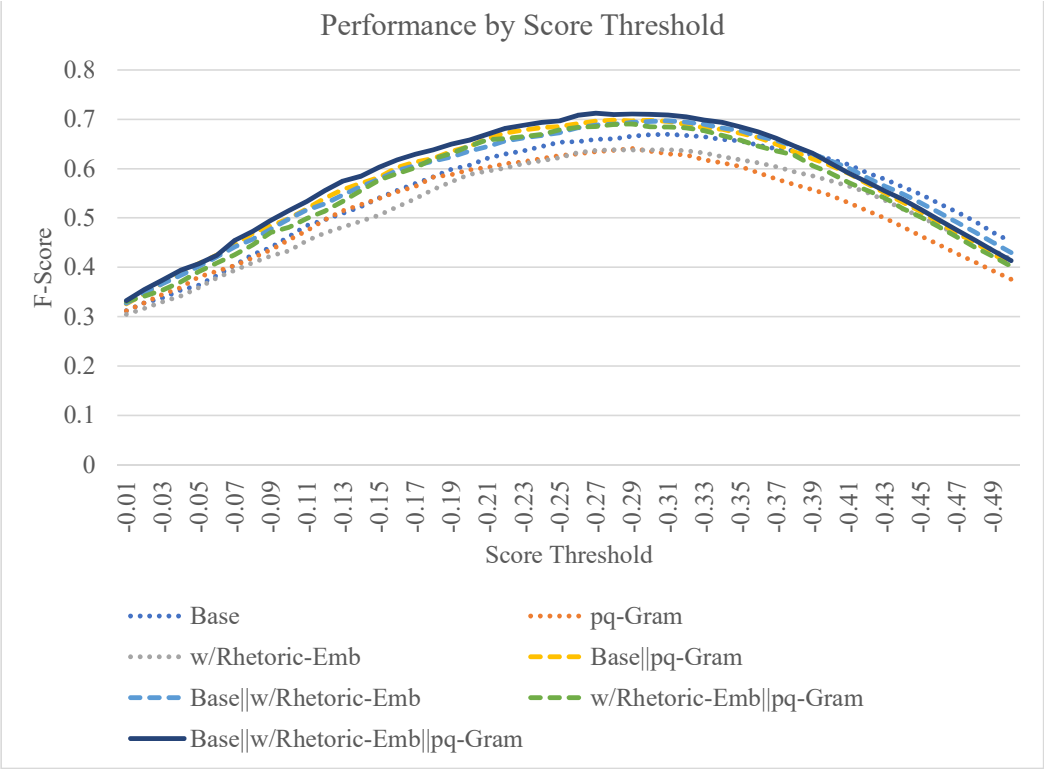


Figure 5.5.4: Performance by score threshold.

5.5.1.5 Score-Threshold-Based Selection

From all the validation results, we see that while our method of selection of predicted supporting cases, which is based on top k of ranked candidates, is simple, but it has a limitation: the value of k does not make use of scores returned by ranking models.

We present another selection strategy, namely, score-threshold-based selection. On the normalized scores by Equation 5.9 with $min_{t=200}$, we select only the candidates whose scores are greater than a threshold s. The results of different values of s is shown in Figure 5.5.4. We achieve best F1 of 0.712 when $s = -0.27$ with the voting of three models: Base, *pq*-Gram and w/Rhetoric-Emb, compared to the best of using top-k selection by a large margin of ΔF1= +0.092.

Table 5.5.8: Performance in f-measure by score threshold. Score normalization (Eq.5.9) is computed with $min_{t=200}$.

	Score Threshold					
Model	-0.25	-0.26	-0.27	-0.28	-0.29	-0.30
Base	0.654	0.655	0.659	0.660	0.666	0.669
pq-Gram	0.626	0.630	0.634	0.637	0.641	0.636
w/Rhetoric-Emb	0.622	0.632	0.637	0.638	0.638	0.638
Base||*pq*-Gram	0.685	0.691	0.696	0.698	0.697	0.698
Base||w/Rhetoric-Emb	0.673	0.683	0.689	0.690	0.693	0.696
w/Rhetoric-Emb||*pq*-Gram	0.678	0.684	0.685	0.689	0.691	0.685
Base||w/Rhetoric-Emb||*pq*-Gram	0.697	0.708	**0.712**	0.710	0.711	0.710

5.5.2 Test Results

Table 5.5.9: Results on test data of COLIEE 2018. We select top 10 highest scored candidates when measuring precision, recall and F-measure. "(summary)" indicates that the corresponding encoding method is applied only on the summary part of the document.

Model	**P**	**R**	**F1**
Lexical	0.458	0.429	0.443
WordEmb-Avg-pooling	0.417	0.391	0.404
WordEmb-Max-pooling	0.331	0.310	0.320
WordEmb-Hierarchical-pooling	0.493	0.463	0.477
doc2vec	0.466	0.437	0.451
WordEmb-Avg-pooling (summary)	0.490	0.459	0.474
WordEmb-Max-pooling (summary)	0.432	0.405	0.418
WordEmb-Hierarchical-pooling (summary)	0.585	0.548	0.566
doc2vec(summary)	0.444	0.417	0.430
EncSum(i)	0.598	0.561	0.579
EncSum(ii)	0.608	0.571	0.589
EncSum-$p = 1, q = 3$ (ii)	0.578	0.542	0.559
EncSum-rhetoric-emb (ii)	0.600	0.563	0.581
EncSum-lead (ii)	0.493	0.463	0.477
Lexical+ *WordEmb*-Avg-pooling	0.569	0.534	0.551
Lexical+ *WordEmb*-Max-pooling	0.566	0.531	0.548
Lexical+ *WordEmb*-Hierarchical-pooling	0.607	0.569	0.587
Lexical+*doc2vec*	0.571	0.536	0.553
Lexical+ *WordEmb*-Avg-pooling (summary)	0.578	0.542	0.559
Lexical+ *WordEmb*-Max-pooling (summary)	0.598	0.561	0.579
Lexical+ *WordEmb*-Hierarchical-pooling (summary)	0.637	0.598	0.617
Lexical+*doc2vec* (summary)	0.622	0.583	0.602
Lexical+EncSum(i)	0.676	0.634	0.655
Lexical+EncSum(ii) (Base)	0.690	0.647	0.668
Lexical+EncSum-$p = 1, q = 3$ (ii) (pq-Gram)	0.692	0.649	0.669
Lexical+EncSum-rhetoric-emb (ii) (w/Rhetoric-Emb)	0.681	0.639	0.660
Lexical+EncSum-lead (ii)	0.610	0.572	0.591
Base$||$$pq$-Gram	0.717	0.780	0.687
Base$\|\|$w/Rhetoric-Emb	0.715	0.783	0.686
w/Rhetoric-Emb$||$$pq$-Gram	0.712	0.781	0.683
Base$\|\|$w/Rhetoric-Emb$||$$pq$-Gram	0.720	**0.784**	0.690
Base (ST)	0.787	0.697	0.689
pq-Gram (ST)	0.778	0.639	0.642
w/Rhetoric-Emb (ST)	0.728	0.728	0.671
Base$||$$pq$-Gram (ST)	0.807	0.705	0.692
Base$\|\|$w/Rhetoric-Emb (ST)	0.763	0.761	0.704
w/Rhetoric-Emb$||$$pq$-Gram (ST)	0.826	0.709	0.707
Base$\|\|$w/Rhetoric-Emb$||$$pq$-Gram (ST)	**0.841**	0.712	**0.715**

Table 5.5.10: Results on test data of COLIEE 2019. We select top 5 highest scored candidates when measuring precision, recall and f-measure.

Model	**P**	**R**	**F1**
Lexical	0.485	0.448	0.466
WordEmb-Avg-pooling	0.157	0.145	0.151
WordEmb-Max-pooling	0.239	0.221	0.230
WordEmb-Hierarchical-pooling	0.334	0.309	0.321
doc2vec	0.403	0.373	0.387
EncSum(i)	0.413	0.382	0.397
EncSum(ii)	0.426	0.394	0.409
Lexical+ *WordEmb*-Avg-pooling	0.489	0.452	0.469
Lexical+ *WordEmb*-Max-pooling	0.541	0.500	0.520
Lexical+ *WordEmb*-Hierarchical-pooling	0.590	0.545	0.567
Lexical+*doc2vec*	0.475	0.439	0.457
Lexical+EncSum(i)	0.544	0.503	0.523
Lexical+EncSum(ii)	**0.600**	**0.555**	**0.576**

Table 5.5.11: Participants' results on test data of COLIEE 2018. We participated in the competition under the name "JNLP". "JNLP-k=10" is our best system utilizing the combination of lexical and encoded summarization[44] using the base parameters.

Model	**P**	**R**	**F1**
HUKB1	0.497	0.308	0.381
HUKB2	0.405	0.304	0.347
JNLP-r=2.5	0.546	0.655	0.596
JNLP-k=10	0.676	0.634	**0.655**
Smartlaw	0.287	0.431	0.345
UA	0.372	0.323	0.346
UA-postproc	0.348	0.404	0.374
UA-smote	0.354	0.393	0.372
UBIRLED-1	0.133	0.623	0.219
UBIRLED-2	0.196	0.720	0.308
UBIRLED-3	0.561	0.102	0.172
UL	0.564	0.302	0.393

Table 5.5.12: Participants' results on test data of COLIEE 2019. We participated in the competition under the name "JNLP". "JNLP.task_1.p" is our best system utilizing the combination of lexical and encoded summarization using the pre-trained phrase scoring model.

Team	**Run name**	**P**	**R**	**F1**
CACJ	submit_task1_CACJ01	0.212	0.585	0.311
CLArg	CLarg	0.927	0.306	0.460
HUKB	task1.HUKB	0.702	0.400	0.510
IITP	task1.IITPBM25	0.626	0.385	0.477
IITP	task1.IITPd2v	0.465	0.346	0.397
IITP	task1.IITPdocBM	0.637	0.388	0.482
ILPS	BERT_Score_0.946	0.681	0.433	0.530
ILPS	BERT_Score_0.96	0.819	0.342	0.483
ILPS	BM25_Rank_6	0.467	0.518	0.491
JNLP	*JNLP.task_1.p*	0.593	0.549	*0.570*
JNLP	**JNLP.task_1.pl**	0.600	0.555	**0.576**
JNLP	**JNLP.task_1.ple**	0.600	0.555	**0.576**
UA	UA_0.52	0.351	0.336	0.344
UA	UA_0.54	0.364	0.324	0.343
UA	UA_0.57	0.356	0.333	0.344

5.6 Summary

We have presented our approach for modeling document summary into continuous vector space. We showed that our approach has positive signs in building an effective legal case retrieval system. The results show the importance of exploiting the summary for solving legal case retrieval task. Furthermore, the improvement by the encoded summarization suggests that this feature type not only embeds the summary properties of the given case but also carries selectively important information from the case content which could be potentially related legal points to the main points of the case. Furthermore, the combination of lexical features and deep learning features generated with neural networks yields positive results for solving the legal case retrieval task. The experimental results show that lexical features and deep learning features complement each other pretty well. The highest performance with either lexical or deep learning features is lower than the lowest performance of the combination. The improvement of the combination hints the existence of deep learning features not captured by lexical approach. We have also 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.

Chapter 6

Conclusion

We have presented our approaches for exploiting structural information of documents to develop document encoding methods for measuring document similarity and predicting document relevance, which are implemented to build legal case retrieval systems. In summary, we have developed our methods for:

- Learning vector representation of document components via context expansion with document hierarchy and cross-references and apply the method to legal information retrieval task. The application is still simple as it only compares vectors one by one. To continue, we will focus on more structural properties of documents which are not yet fully exploited in this work, for instance, the comparison of vector representations in hierarchical fashion.

- Encoding sentences with convolutional operations on pq-gram representations of dependency trees, and our application on sentence-pair modeling. The representations are flexible with adjusting pq values while keeping the semantic dependency from dependency trees. Besides, the representations provide more local information in the form of intermediary encodings than n-gram and subtree (containing all children) representations by the average number of children per node, which, however, may require feature comparison/alignment to reduce search space. Our approach achieve competitive performance with related methods using tree composition in sentence encoding.

- Rhetorical status recognition, the first step of our goal to obtain the discourse analysis, using deep learning for feature extraction and conditional random field for solving the recognition task as sequential labeling problem. We have achieved encouraging results since, we have not yet utilized any linguistic features, but only statistical features by deep neural network from a considerably insufficient dataset. The applicability of deep neural network to this task sets the step for incorporating rhetorical information into more sophisticated deep learning models of tasks such as summarization, information retrieval.

- Modeling document summary into continuous vector space. We showed that our approach has positive signs in building an effective legal case retrieval system. The results show the importance of exploiting the summary for solving legal case retrieval task. Furthermore, the improvement by the encoded summarization suggests that this feature type not only embeds the summary properties of the given case but also carries selectively important information from the case content which

could be potentially related legal points to the main points of the case. Furthermore, the combination of lexical features and deep learning features generated with neural networks yields positive results for solving the legal case retrieval task. The experimental results show that lexical features and deep learning features complement each other pretty well. The highest performance with either lexical or deep learning features is lower than the lowest performance of the combination. The improvement of the combination hints the existence of deep learning features not captured by lexical approach. We have also showed that the phrase scoring model trained from one dataset can provide useful features for representing documents in other dataset, which shows the generalization of our method. Besides, the result difference in how document relevance is modeled by each implementation of encoded summarization shows the potential of achieving a better legal case retrieval system.

In this work, we have studied several known and human defined structures of documents. For future direction, we would like to study the other kinds of structures: auto-discovery of structures from data. For example, unsupervised dependency tree parsing, one instance of grammar induction, which have still been being developed[46, 47, 48, 49]. Another example is multi-head attention mechanism which learns word relations automatically from corpus without human intervention[50]. The attention mechanism is implemented in BERT[51], which have achieved the state of the art performance in a number of natural language processing tasks.

Bibliography

[1] William P Headden III, Mark Johnson, and David McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109. Association for Computational Linguistics, 2009.

[2] Phil Blunsom and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213. Association for Computational Linguistics, 2010.

[3] Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP 2014*, 2014.

[4] Aliaksei Severyn and Alessandro Moschitti. 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*, pages 373–382. ACM, 2015.

[5] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 130, 2016.

[6] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668, 2017.

[7] Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1466–1477, 2016.

[8] Tsendsuren Munkhdalai and Hong Yu. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 11–21, 2017.

[9] William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

[10] Ben Hachey and Claire Grover. A rhetorical status classifier for legal text summarisation. *Text Summarization Branches Out*, 2004.

[11] Filippo Galgani, Paul Compton, and Achim Hoffmann. Towards automatic generation of catchphrases for legal case reports. In *Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part II*, CICLing'12, pages 414–425, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-28600-1. doi: 10.1007/978-3-642-28601-8_35. URL http://dx.doi.org/10.1007/978-3-642-28601-8_35.

[12] L.T. Olsson and Australian Institute of Judicial Administration. *Guide to Uniform Production of Judgments*. Australian Institute of Judicial Administration, 1999. ISBN 9781875527250. URL https://books.google.co.jp/books?id=mKnAAQAACAAJ.

[13] Filippo Galgani, Paul Compton, and Achim Hoffmann. Citation based summarisation of legal texts. In *Pacific Rim International Conference on Artificial Intelligence*, pages 40–52. Springer, 2012.

[14] Arpan Mandal, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh. Automatic catchphrase identification from legal court case documents. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2187–2190, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4918-5. doi: 10.1145/3132847.3133102. URL http://doi.acm.org/10.1145/3132847.3133102.

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

[16] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[17] Yang Liu and Meng Zhang. Neural network methods for natural language processing. *Computational Linguistics*, 44(1):193–195, 2018. doi: 10.1162/COLI_r_00312. URL https://doi.org/10.1162/COLI_r_00312.

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

[19] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL (2)*, pages 302–308, 2014.

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[21] Ken Satoh, Mi-Young Kim, Yoshinobu Kano, Randy Goebel, and Tiago Oliveira, editors. *COLIEE 2017. 4th Competition on Legal Information Extraction and Entailment*, volume 47 of *EPiC Series in Computing*, 2017. EasyChair.

[22] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[23] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[24] Nikolaus Augsten, Michael Böhlen, and Johann Gamper. The pq-gram distance between ordered labeled trees. *ACM Trans. Database Syst.*, 35(1):4:1–4:36, February 2008. ISSN 0362-5915. doi: 10.1145/1670243.1670247. URL `http://doi.acm.org/10.1145/1670243.1670247`.

[25] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223, 2014.

[26] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, 2015.

[27] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8, 2014.

[28] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/N18-1101`.

[29] Natalie M Schrimpf. Using rhetorical topics for automatic summarization. *Proceedings of the Society for Computation in Linguistics (SCiL) 2018*, pages 125–135, 2018.

[30] Ashwini Rahangdale and AJ Agrawal. Information extraction using discourse analysis from newswires. *International Journal of Information Technology Convergence and Services*, 4(3):21, 2014.

[31] Subhabrata Mukherjee and Pushpak Bhattacharyya. Sentiment analysis in twitter with lightweight discourse analysis. *Proceedings of COLING 2012*, pages 1847–1864, 2012.

[32] Peter Jansen, Mihai Surdeanu, and Peter Clark. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 977–986, 2014.

[33] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Conference on Machine Learning*, pages 282–289, 2001.

[34] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[35] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D14-1181`.

[38] Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. Character-based bidirectional lstm-crf with words and characters for japanese named entity recognition. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 97–102, 2017.

[39] Rie Johnson and Tong Zhang. 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*, pages 103–112, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N15-1011`.

[40] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P14-1062`.

[41] Vu D. Tran, Minh L. Nguyen, and Ken Satoh. Automatic catchphrase extraction from legal case documents via scoring using deep neural networks. In *Workshop on MIning and REasoning with Legal texts*, 2018.

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

[43] Mi-Young Kim, Ying Xu, and Randy Goebel. Summarization of legal texts with high cohesion and automatic compression rate. In Yoichi Motomura, Alastair Butler, and Daisuke Bekki, editors, *New Frontiers in Artificial Intelligence*, pages 190–204, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39931-2.

[44] Vu Duc Tran, Son Truong Nguyen, and Minh Le Nguyen. Jnlp group: Legal information retrieval with summary and logical structure analysis. In *Twelfth International Workshop on Juris-informatics (JURISIN), COLIEE*, 2018.

[45] Yoshinobu Kano, Mi-Young Kim, Masaharu Yoshioka, Yao Lu, Juliano Rabelo, Naoki Kiyota, Randy Goebel, and Ken Satoh. Coliee-2018: Evaluation of the competition on legal information extraction and entailment. In *Twelfth International Workshop on Juris-informatics (JURISIN), COLIEE*, 2018.

[46] Valentin I Spitkovsky, Hiyan Alshawi, Angel X Chang, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1281–1290. Association for Computational Linguistics, 2011.

[47] Yong Jiang, Wenjuan Han, and Kewei Tu. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, 2016.

[48] David Marecek. Twelve years of unsupervised dependency parsing. In *ITAT*, pages 56–62, 2016.

[49] Jiong Cai, Yong Jiang, and Kewei Tu. Crf autoencoder for unsupervised dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1638–1643, 2017.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[51] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Publications

[1] Vu Tran, Minh Le Nguyen, and Ken Satoh. Building legal case retrieval systems with lexical matching and summarization using a pre-trained phrase scoring model. *17th International Conference on Artificial Intelligence and Law*, 2019.

[2] Vu D. Tran, Minh Le Nguyen, and Ken Satoh. Encoding local contexts of sentences with convolutions on pq-gram representations of dependency trees. *10th International Conference on Knowledge and Systems Engineering (KSE)*, pages 329–334, 2018.

[3] Vu D. Tran, Son Truong Nguyen, and Minh Le Nguyen. Jnlp group: Legal information retrieval with summary and logical structure analysis. *the Twelfth International Workshop on Juris-informatics*, 2018.

[4] Vu D. Tran, Minh Le Nguyen, and Ken Satoh. Combining lexical and latent features for legal case retrieval task. *the Third International Workshop on SCIentific DOCument Analysis*, 2018.

[5] Vu D. Tran, Minh Le Nguyen, and Ken Satoh. Automatic catchphrase extraction from legal case documents via scoring using deep neural networks. *MIREL 2018 workshop on MIning and REasoning with Legal texts*, 2018. URL `http://arxiv.org/abs/1809.05219`.

[6] Vu D. Tran, Minh Le Nguyen, and Ken Satoh. Similarity with document components vector representations by context expansion from document structures. *the Second International Workshop on SCIentific DOCument Analysis*, 2017.

[7] Danilo S. Carvalho, Vu Duc Tran, Khanh Van Tran, and Minh Le Nguyen. Improving legal information retrieval by distributional composition with term order probabilities. *the Fourth Competition on Legal Information Extraction/Entailment, EPiC Series in Computing*, 47:43–56, 2017.

[8] Danilo S. Carvalho, Vu Duc Tran, Khanh Van Tran, Viet Dac Lai, , and Minh L. Nguyen. Lexical to discourse-level corpus modeling for legal question answering. *the 10th International Workshop on Juris-informatics, JURISIN*, 2016.

[9] Minh-Tien Nguyen, Duc-Vu Tran, and Le-Minh Nguyen. Social context summarization using user-generated content and third-party sources. *Knowledge-Based Systems*, 144:51–64, 2018.

[10] Minh-Tien Nguyen, Duc-Vu Tran, Le-Minh Nguyen, and Xuan-Hieu Phan. Exploiting user posts for web document summarization. *ACM Trans. Knowl. Discov. Data*, 12(4), 2018.

Awards

- Research Grants for JAIST Students, 2016.
- **COLIEE 2018 Winning Group**: the best performance on the Legal Case Retrieval Task of the Competition on Legal Information Extraction/Entailment.
- **COLIEE 2019 Winning Group**: the best performance on the Legal Case Retrieval Task of the Competition on Legal Information Extraction/Entailment.