Information Extraction and Entailment for Statute Law and Case Law

Juliano Rabelo[1], Mi-Young Kim[1,2], Housam Babiker[1,3], Randy Goebel[1,3], and Nawshad Farruque[1,3]

[1]Alberta Machine Intelligence Institute, University of Alberta, Edmonton, AB, Canada
[2]Department of Science, Augustana Faculty, University of Alberta, Camrose, AB, Canada
[3]Department of Computing Science, University of Alberta, Edmonton, AB, Canada
{rabelo,miyoung2,khalifab,rgoebel,nawshad}@ualberta.ca

Abstract. The challenge of information overload in the legal domain is increasing every day. In the Competition on Legal Information Extraction and Entailment (COLIEE), we tackle four challenges which are intended to encourage the development of systems and methods to alleviate that pressure: a case law retrieval (Task 1) and entailment (Task 2), and a statute law retrieval (Task 3) and entailment (Task 4). For Tasks 1 and 2, we develop and apply a pairwise paragraph similarity approach, which ranked 4th place in Task 1 and 1st place in Task 2 among all teams participating in COLIEE 2018. For the statute law textual entailment challenge, we use a textual segmentation of legal data, based on characteristics of statute law, then confirm textual entailment based on identified negation/antonym relations. We have evaluated our system using the data from COLIEE 2018. The statute law competition in COLIEE focuses on the legal information processing required to answer yes/no questions from legal bar exams, and it consists of two phases: legal ad-hoc information retrieval (Task 3), and textual entailment (Task 4). We focus on Task 4, and our experimental evaluation demonstrates the effectiveness of the legal segmentation and negation/antonym detection. Our results show that our methods were ranked first for both Tasks 2 and 4 of the COLIEE 2018 competition.

Keywords: legal textual retrieval · legal textual entailment · document similarity · binary classification · imbalanced datasets

1 Introduction

Every day, large volumes of legal data are produced by law firms, law courts, independent attorneys, legislators, and many other sources. In that scenario, management of legal information becomes manually intractable, and requires the development of tools which automatically or semi-automatically aid legal professionals to handle the information overload. In the COLIEE competition, we face four facets of that challenge: case law retrieval, case law entailment, statute law retrieval and statute law entailment. Here we provide the details of our approaches to those tasks, evaluate the results achieved and comment on future work to further improve our models.

For the case law retrieval (Task 1), our approach relies on measuring the similarity between paragraphs of legal cases by generating feature vectors based on those similarities, and then using a classifier to determine if those cases should be noticed or not. We compare that strategy with a baseline algorithm which uses the full text of the cases with a similarity calculation, to confirm the hypothesis that considering individual paragraphs improves the comparison performance.

A similar strategy is applied for the entailment task (Task 2). In that case, we augment the model with word embeddings and use the available decision and summary information, in addition to the case law contents, for the similarity calculation. For both Tasks 1 and 2, we discuss the problem of class imbalance and propose, then evaluate, strategies to cope with that problem.

In the statute law task, our approach to Yes/No legal question answering requires two steps. First, an information retrieval system must retrieve relevant documents for a legal question (Task 3). Second, a textual entailment system compares the semantic connections between question and relevant documents, then determines if an entailment relation holds (Task 4). Here, we focus on Task 4. We first segment the identified relevant statute (identified by Task 3) based on legal conditions/conclusions/exceptions, and then detect negation/antonym relations for textual entailment. The method will be detailed in Section 4.

2 Case Law Retrieval

This task consists in finding which cases, in the set of candidate cases, should be “noticed” with respect to a given query case. “Notice” is a legal technical term that denotes a legal case description that is considered to be relevant to a query case. More formally, given a query case q and a set of candidate cases $C = \{c_1, c_2, ..., c_n\}$, the task is to find the supporting cases $S = \{s_1, s_2, ..., s_n \mid s_i \in C \wedge \mathit{noticed}(s_i, q)\}$ where $\mathit{noticed}(s_i, q)$ denotes a relationship which is true when $s_i \in S$ is a noticed case in respect to q.

2.1 Dataset Analysis

The dataset provided has 285 query cases, each consisting of a summary file (i.e., a short, summarized version of the case) and a fact file containing a list of paragraphs. For each query case, a set of 200 candidate cases is given, from which must be identified a subset of noticed cases. The candidate cases are given in raw text, i.e., they are a full text extraction of the case laws descriptions. Table 1 summarizes the dataset properties.

2.2 Our Method

As noted in Section 2, we are required to identify which candidate cases should be noticed with respect to each given query case. Noticing a case is something a

Table 1. Summary for the Case Law Retrieval Task dataset.

Property	Value
Number of query cases	285
Candidate cases per query	200
Total number of candidate cases	57,000
Total number of true noticed cases	2,813 (4.93%)
Average number of noticed cases per query	9.87
Standard deviation of noticed cases counts	8.8143

lawyer does when s/he believes that case is somehow relevant to the case being analyzed. “Somehow relevant” is to some degree subjective: a court is obliged to consider cases in which the same issue has been decided in a court whose decisions are binding. Similarly, a court is obliged to consider its own precedents. But a lawyer may also find a case is relevant and should be noticed because it is on the same legal subject, because it addresses similar facts, because it addresses similar parties, because it requires a similar type or structure of analysis, or it is for some other reason usefully analogous. Our method to identify noticed cases is built on an important assumption: we consider a case should be noticed if it is similar to the query case. Similarity can also be somewhat vague, but seems to be a generic concept that captures most variations of the valid reasons for noticing a case. So our assumption rests on the choice of similarity measure: there are several well-known existing document similarity measures [1] that we can adopt for this task.

The Baseline Approach The baseline algorithm implemented for the case law retrieval task is a simple document similarity calculator: if the similarity measured between the query case and the candidate case is above a threshold, the documents are considered similar (i.e., they should be noticed). We perform some preprocessing on the documents before measuring similarity, i.e., removal of punctuation, digits and stop words, case normalization and stemming.

In addition to these preprocessing techniques, we also join hyphenated words into a single word, since the input documents contain hyphenated words (e.g., “attor-ney”, “deci-sion”). The algorithm joins two tokens separated by an hyphen if the result is a valid English word, which is determined by dictionary lookup. That prevents joining two words which should really be separated by a dash (e.g., “ex-president”, “vice-consul”). The dictionary for the word lookup was built by parsing an English-German dictionary [15], keeping only the English words. The outcome of that process was a list of $\sim 170,000$ English terms.

In our baseline approach, the similarity between two cases is calculated as the cosine of the vectors formed by a bag of words representation of each document [2]. The results achieved for different threshold values are shown on Table 2.

The best F-measure scores were 0.2944 and 0.2947 (for thresholds of 0.35 and 0.40, respectively) and are a basis for comparison with other methods.

Table 2. Baseline results.

Threshold / Measure	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Recall	0.8115	0.6352	0.4827	0.3668	0.2609	0.1809	0.1038
Precision	0.0897	0.1235	0.1743	0.2459	0.3387	0.4480	0.5407
F-measure	0.1616	0.2068	0.2561	0.2944	0.2947	0.2577	0.1741

Pair-wise Paragraph Comparison Calculating the similarities using candidate cases full text is intuitive, but in fact there are studies showing that the main concepts of legal cases are usually encoded in a few paragraphs [3]. To leverage that idea, we implemented a similarity-based approach that compares the query case and each candidate case paragraph by paragraph.

That comparison step would now generate not a similarity score, but a similarity matrix. We could then use an aggregation function (such as average) to enable us to set a threshold and make a decision on whether the cases are similar, but this approach would also sacrifice the more detailed information available, making it similar to the baseline approach presented above.

Another option which would leverage the paragraph-wise similarities would be to use that as an input to a supervised learning classifier. However, the similarity matrices generated have a variable number of rows and columns (since their sizes are defined by how many paragraphs the input cases have). So we created a histogram over the similarity matrices as a means of regularizing the input to the classifier. We define the bounds of the similarity bins and count how many cells in the matrix fall under each bin, then we normalize the histogram by the number of cells in the matrix. This process produces a fixed size vector which can then be used as an input to a supervised learning classifier.

To leverage other studies [3], we also measure the similarity considering only "legal terms"[1], i.e., we use a list of law-related terms and pre-process the paragraphs retaining only relevant terms. The similarities scores were collected and processed in the same way as the "regular" paragraph similarities, thus augmenting the feature vectors which are then provided to the learning classifier.

One problem which emerged was the significant class imbalance in the competition dataset (see Section 2.1), which contains only 4.93% noticed cases. To cope with that problem, we under-sampled the "false" class (i.e., the candidates which are not noticed cases). We did not want to shrink the training dataset to only as many positive samples as there are available, so we kept the negative dataset size five times bigger than the positive dataset. We also attempted to oversample the positive dataset, first by replicating the positive samples and then by synthesizing samples using SMOTE [4]. None of those approaches proved to be effective for this task. We also tried to use a meta cost sensitive classifier [5], but it only made sense when we used a fully balanced dataset for training, so we dropped that idea in favor of the use of a bigger training dataset.

[1] the legal terms list was created by using a few law dictionaries

For the classification itself, we tried a few different options: Random Forest [6], Gradient Boosting [7] and GCForest [8]. Given the gain in performance was significant (see Table 3), we chose to run the rest of the experiments using only the Random Forest classifier.

Table 3. Results for the Pairwise Paragraph Comparison Approach.

Algorithm Measure	Precision	Recall	F-measure
Random Forest	0.4200	0.3194	0.3629
Gradient Boosting	0.1422	0.3541	0.2029
GC Forest	0.3684	0.2430	0.2928

Pair-wise Paragraph Comparison Augmented with Word Embeddings A problem with the similarity based approach is that there might be paragraphs which express similar concepts using very different phrasings. But since words with similar concepts should have similar vector representations in word embeddings [9], its application could help overcome that problem. The framework used in our experiments was Word2vec [9] and the similarity was calculated using the Word Mover's Distance [10]: basically the smallest distance all embeddings from one document (in our case, paragraph) need to "travel" to reach the other document's embeddings. We used a word embedding model pre-trained on a general purpose set of documents [12]. We believed we could achieve better results by training our own embeddings using case law data, thus improving the representation for case law specific terms. So we downloaded and parsed Case Law Reports [11], but in our experiments that was not enough to out perform the general purpose model.

Post processing By analyzing the available dataset (Table 1), we see there are on average 10 noticed cases for each query case; so we assumed a range for minimum and maximum number of noticed cases for each query case. That could help, especially in decreasing false positives if there are cases for which the algorithm identifies too many noticed cases. But for this task, the post processing did not help as much as it did for Task 2. Our assumption is that the much higher standard deviation seen in the Task 1 dataset impaired the results.

The pipeline we used in our official COLIEE submissions is shown on Fig. 1. The light gray boxes represent modules which are used in specific experimental setups. The configurations for training and testing share the first two modules.

2.3 Results

Below we show the results achieved by all variations of the method on the training dataset, and the variations selected for submission on the COLIEE test dataset.

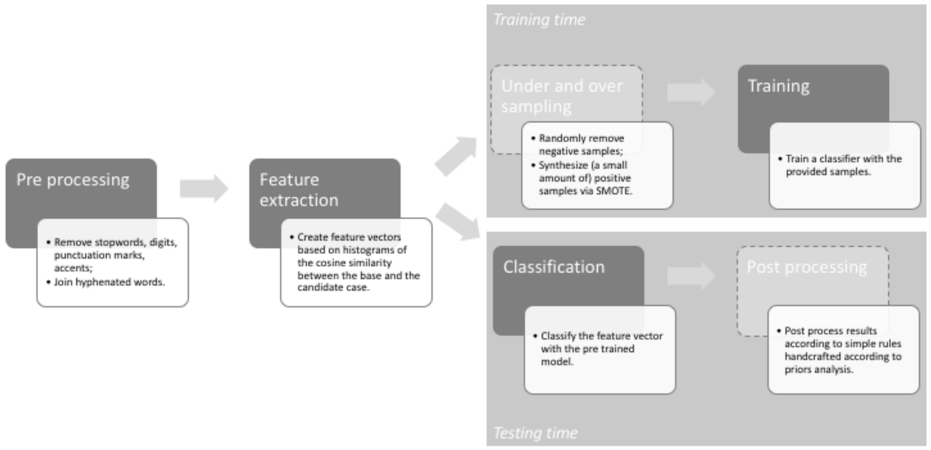
Training time
Under and over sampling
Randomly remove negative samples;
Synthesize (a small amount of) positive samples via SMOTE.
Training
Train a classifier with the provided samples.
Pre processing
Remove stopwords, digits, punctuation marks, accents;
Join hyphenated words.
Feature extraction
Create feature vectors based on histograms of the cosine similarity between the base and the candidate case.
Classification
Classify the feature vector with the pre trained model.
Post processing
Post process results according to simple rules handcrafted according to priors analysis.
Testing time

Fig. 1. Pipeline used for the official COLIEE task 1 submission.

On the Training Dataset To evaluate the results on the training dataset, we set apart 29 cases from the original 285 cases (see Table 1). Those cases were not used on the development of our method (training classifiers, manual inspection, etc.). The results achieved on that validation dataset are shown on Table 4.

Table 4. Results for the Case Law Retrieval Task (validation dataset).

Algorithm	Precision	Recall	F-Measure
Baseline Approach	0.2609	0.3387	0.2947
Pair-wise Paragraph Comparison (PPC)	0.4200	0.3194	0.3629
PPC with word embeddings	0.4198	0.3090	0.3560
PPC with SMOTE	0.3660	0.3888	0.3771
PPC with Post Processing	0.3087	0.3784	0.3400

From those experiments, we selected to submit to the official COLIEE evaluation the results of PPC, PPC with SMOTE and PPC with Post Processing. PPC with word embeddings did not provide F-measure improvements, in comparison with PPC, and was computationally quite expensive.

On the COLIEE Test Dataset The results achieved by the three selected methods are presented on Table 5.

The results attained on the test dataset were consistent with those obtained on the validation dataset. We performed a random test (a significance test recommended for analyzing an F-measure performance [23]) over our results and could conclude the differences on performance shown by the three configurations above are not statistically relevant. Compared against a naive classifier

Table 5. Results for the Case Law Retrieval Task (real COLIEE test dataset).

Method	Precision	Recall	F-Measure
Pair-wise Paragraph Comparison (PPC)	0.3724	0.3227	0.3458
PPC with SMOTE	0.3538	0.3926	0.3722
PPC with Post Processing	0.3484	0.4038	0.3740

which outputs all entries as noticed cases (which would achieve 1.0, 0.0533 and 0.1012 for recall, precision and F-measure respectively on the test dataset), the improvement shown by our classifiers are statistically relevant for a significance level of 0.01.

2.4 Future Work

We plan to continue to explore further alternatives for this task. For example, the analysis showed the similarity based features do provide relevant information for the classifiers, but were not great discriminants for the problem. To overcome that problem, we might extract other features. One alternative is to extract LIWC features [13]. A more immediate option is the use of the summary data (we only used the case law main contents in our experiments).

We also intend to perform other experiments with word embeddings. Although the results achieved in our experiments were not so promising, we believe the idea may be refined. The first thing would be to use specific-domain data to train the model. We do have some data available, which has been downloaded from [11], and we could augment it with the 57,000 candidate case laws provided as this task's training dataset. Another slightly different option is to re-train a general purpose word embedding model with our data. We believe those alternatives would improve the vector representations of the model for the problem at hand. We could also try coarser-grained embeddings (sentence or document level embeddings [20], as opposed to the current word-based implementation). However, we would need significantly more data to train those embeddings, which makes it a poor fit for the COLIEE scenario, where relatively little data is available. A more viable alternative would be the application of different frameworks, such as GloVe [16] or Fasttext [17], which may capture better vector representations in our scenario, or trying a different learning model for the embeddings. We used the default CBOW model, but an option worth trying would be to use skip-gram (which usually works better than CBOW for smaller datasets [18]).

One of the major problems which impacted our results was the severe class imbalance. For the training to be performed reasonably, we needed to balance the training dataset with undersampling/oversampling. Still, the classifier is applied to a real test dataset which presents the original imbalance. Overcoming that situation requires modeling the problem in a different way. One option would be modeling it as outlier detection. We could train a one-class classifier (e.g., One Class SVM [19]) with the false samples (the more abundant class). Samples classified as not belonging to that class would be considered true noticed cases.

Another alternative would be applying clustering to the training dataset, possibly allowing multiple clusters for each class to capture diversity in the training data. New samples would be classified with the class of the nearest clusters.

A more sophisticated alternative would be the extraction of the ratio decidendi [21] for each case. If we can successfully implement that and identify an effective similarity measure between rationes decidendi, it would be simple to establish some sort of threshold to determine which cases should be noticed. However, the single previous description on automatically extracting the ratio decidendi [21] has been tested on only one case law; the technique may not be adequate in an open world scenario. Extracting a ratio decidendi means fully understanding a legal case, and implementing a general approach to that problem is not something already solved in the literature. But in general, the development of methods for fully understanding case law is precisely the ultimate goal of COLIEE Tasks 1 and 2.

3 Case Law Entailment

Given a base case and its respective decision, and a second case which is relevant in respect to the base case, this task consists in determining *which paragraphs* of the second case entail the decision of the base case. More formally, given a base case b and its decision d, and another case r represented by its paragraphs $P = \{p_1, p_2, ..., p_n\}$ such that $noticed(b, r)$ as defined in section 2 is true, we need to find the set $E = \{p_1, p2, ..., p_m \mid p_i \in P\}$ where $entails(p_i, d)$ denotes a relationship which is true when $p_i \in P$ entails the decision d.

In the following subsections, we present an overview of the provided dataset, discuss details of our method and analyze the results it achieved.

3.1 Dataset Analysis

The dataset provided for this task has 181 base cases, each consisting of a summary file, a fact file (a list of the case paragraphs), and a decision file. Each base case includes a respective related case represented by a list of paragraphs, from which the paragraphs that entail the base case decision must be identified. Table 6 summarizes the dataset properties.

Table 6. Summary for the Case Law Entailment Task dataset.

Property	Value
Number of base cases	181
Total number of paragraphs in the related cases	8,794
Total number of true entailment paragraphs	239 (2.71%)
Average number of entailment paragraphs per base case	1.32
Standard deviation of entailment paragraphs counts	0.7106

3.2 Our Entailment Method

Not surprisingly, this task is even more challenging than Task 1. A perfect solution would ideally require in-depth domain and common sense knowledge, plus a reasoning capability in order to mimic the whole process undertaken by a human annotator, who analyses the data to determine which paragraphs from one case entail a decision for a different case. Since we do not currently have such tools, we needed to try simpler approaches which do not attempt to explicitly reason and come to a conclusion. Our initial method tries to extract relevant features which (hopefully) correlate with the entailment relationship, and then train a classifier which is capable of learning that relationship. As in Task 1, we also face the same problems of data scarcity and class imbalance. But those problems are more extreme for Task 2. Table 6 shows we have less data for this task and class imbalance is more severe. We experimented three oversampling techniques to cope with the imbalance problem; in addition to simple replication and SMOTE (see section 2.2), we also implemented a new instance synthesis algorithm: for every instance p in the positive dataset, compute a new instance of p as $\mu_p + p$.

Preprocessing The preprocessing used here was a little more elaborate than for Task 1: in addition to using (case normalization, stop words, punctuation, accents and digit removal, and hyphenated words joining), we also implemented a language detection algorithm. This was necessary because the dataset contains many paragraphs written in French[2]. Since our approach is based on the similarity between documents, and the input cases are in English only, we decided to not consider the French candidates, although there are 6 expected answers which are identified by our algorithm as French written paragraphs for Task 2. This was considered an acceptable limitation of our method, since those cases represent only 2.5% of the total positive samples. Moreover, the task is already very difficult, so we decided not to tackle cross language entailment, for now. After the removal of the French paragraphs, we could use the English instances to train our model[3].

Looking for Features Our first attempt relied solely on searching for words which could indicate a paragraph was written in a way the judge would be expressing his/her decision. This would not consider any relationship with the base case, but instead would only look for "decision-like" paragraphs. This was not expected to provide the final method for Task 2, but only a means of verifying the correlation between some words and the task goal. In fact, by only looking

[2] The source of COLIEE case law is from Canadian case law, which includes both English and French; our algorithm could identify 145 instances of French paragraphs in the dataset

[3] The official COLIEE test dataset did not present any cases of French paragraphs. Still, removing those paragraphs was important because we used the training dataset, which does contain French paragraphs, for training our classifiers.

up a few words such as first person pronouns ("I," "my" etc.), we could identify around 25% of the entailment paragraphs (a good recall for such a simple approach). However, the precision was around 2%, suggesting that this method alone was far from ideal (our F-measure would be 0.0370). However, it did show that there might be words correlated with the entailment paragraphs.

Word Clouds A slightly more sophisticated approach in that direction was based on the generation of word clouds for the true and false samples. We divided all the candidate paragraphs according to their label (true entailment or not) and generated an aggregate word cloud (or bag of words) for each class. We then took the first n words from each cloud, removed duplicates, and generated a dictionary. Each paragraph in the related cases were then encoded as one-hot vectors according to that dictionary; the resulting vectors were fed to a classifier.

Since the resulting vectors were very sparse, we augmented this approach with word embeddings: for a word w in the paragraph which was not in the dictionary, we looked up the most similar word in the embedding (i.e., the closest concept) and used the similarity score as the feature for w in our feature vector. Consequently, the feature vectors were not one-hot encoded, being now in the $[0 - 1]$ range. Results on the evaluation dataset are provided in Section 3.3.

The Similarity-Based Approach Since the above approaches did not provide acceptable results, our next attempt was based on the similarity method presented on section 2. We measured the similarity between each paragraph in the noticed case and the base case, generating a feature vector comprised of the measures of similarity between the paragraph and (1) the decision of the base case, (2) the base case summary and (3) the base case paragraphs (actually a histogram of the similarities between each noticed case paragraph and all paragraphs from the base case). In the end, we had one feature vector representing each "candidate" paragraph (i.e., a paragraph in the noticed case, which in this task is a candidate for entailing the decision on the base case). That approach provided much better results. Our experiments included Random Forest (RF) [6] and Gradient Boosting (GB) [7] classifiers. See Section 3.3 below for details.

LIWC Features In an attempt to further improve those results, we augmented the feature vectors with LIWC features [13]. LIWC (Linguistic Inquiry and Word Count) is a text analysis tool that provides useful insights on a text blob based on 94 features. These 94 features are attributed to various dimensions of a text, for example: Linguistic (e.g., pronouns, adjectives, adverbs, etc) and Affective (e.g., positive/negative emotions, cognitive processes, etc). Those features can be seen as dictionaries of relevant words from different dimensions created by domain experts. When a text fragment is fed to LIWC, it produces an output that shows the count of 94 feature related words normalized by the total word count of that text fragment. LIWC is widely used in Psycholinguistics [14] to identify different mood/affective disorders and understand different cognitive processes from human language. Those features did provide relevant improvements on the model performance, as shown on Subsection 3.3.

Post Processing Our final attempt to improve the results consisted in a post processing phase. Since the model receives only a feature vector representing a candidate paragraph, it has no built-in schema for handling how the cases are structured nor input on how many responses for each case would be reasonable. So it could classify 30 paragraphs as positive entailment instances for one case and identify none for another case. But by analyzing the priors in the provided dataset (see 6), we can see that, on average, there are ~ 2 entailment paragraphs for each case. Therefore we tried to post-process the results to retain at most 5 entailment paragraphs for cases which had many candidates identified as true samples, and at least 1 paragraph for cases which had no candidates identified as true samples. To select retained paragraphs, we used the classifier confidence score. That measure provided a relevant increase in the model performance, especially for the precision score, as it is shown on table 7 (Subsection 3.3).

The overall pipeline used for the official COLIEE submission is depicted in Fig. 2. The features were extracted according to the "similarity-based approach" described before. But, differently from Task 1, we did not use SMOTE for over sampling, and post processing was used in all three submissions for Task 2.

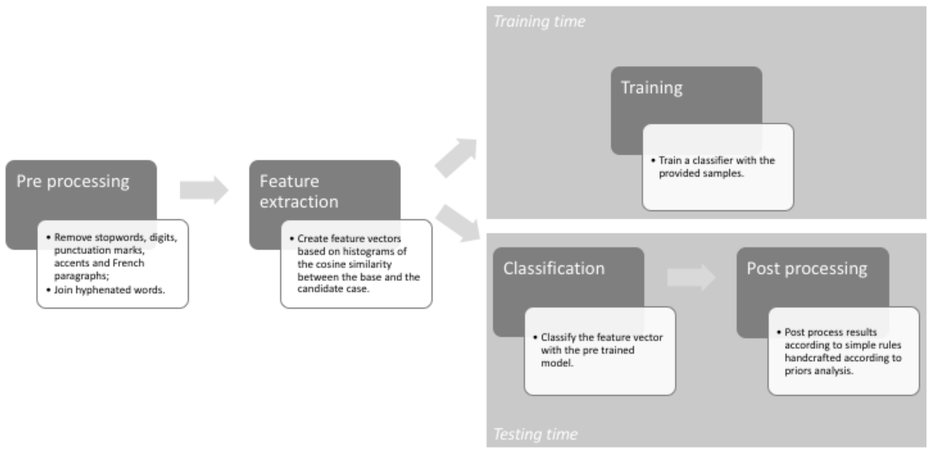
Training time
Training
• Train a classifier with the provided samples.
Pre processing
• Remove stopwords, digits, punctuation marks, accents and French paragraphs;
• Join hyphenated words.
Feature extraction
• Create feature vectors based on histograms of the cosine similarity between the base and the candidate case.
Classification
• Classify the feature vector with the pre trained model.
Post processing
• Post process results according to simple rules handcrafted according to priors analysis.
Testing time

Fig. 2. Pipeline used for the official COLIEE task 2 submission.

3.3 Results

Below we provide the results achieved by all variations of the method on the training dataset, and the three variations submitted on the COLIEE test dataset.

On the Training Dataset To evaluate the results on the training dataset, we set apart 25 cases from the original 181 cases (see Table 6). Those cases where not used on the development of our method (training classifiers, manual inspection, etc). Table 7 shows the results on that validation dataset.

Table 7. Results for the Case Law Entailment Task (validation dataset).

Algorithm	Precision	Recall	F-Measure
Word cloud	0.080	0.133	0.100
Similarity based	0.108	0.333	0.163
Similarity based + Word cloud	0.074	0.467	0.128
Similarity based + LIWC	0.238	0.167	0.196
Similarity based + Post processing	0.2424	0.2857	0.2622
Sim. based with Grad. Boosting + Data augmentation	0.7	0.25	0.3684

As a comparison baseline, we might consider a naive classifier which outputs all paragraphs as entailing the corresponding decisions, which would have the following scores in our validation dataset: 0.0291 (precision), 1.0 (recall) and 0.0428 (F-measure). The other naive classifier option (which would output false for all candidate paragraphs) would have a recall score of 0.0 and, as a consequence, an F-measure of 0.0, and thus is not a viable benchmark.

From Table 7, the best results our model attained were achieved using the similarity approach with a Gradient Boosting classifier and data augmentation (see Subsection 3.2). However, we finished implementing it after the COLIEE dry run deadline, so we did not submit a formal result based on that approach. Not considering it, the best results were attained by combining the similarity based approach (with a Random Forest classifier) and post processing. That improved precision considerably and, despite a drop in recall, the F-measure had a steady improvement of 60.85% (from 0.163 to 0.2622).

The word cloud approach showed poor results when used by itself, and even impaired the results of the similarity-based approach when combined with it. Augmenting the similarity based approach with LIWC features provided a relevant increase of 20.24% on the F-measure.

On the COLIEE Test Dataset Since we achieved the best results with the similarity based + post processing approach (see Table 7), we submitted three results based on it (acronym SB+PP), varying solely the number of estimators for the Random Forest. The results on the test dataset are shown on Table 8.

The best F-measure achieved was 0.258, which is still quite low and even somewhat disappointing. Nonetheless, that was the best performance among all teams which submitted results for Task 2 in COLIEE 2018. We also performed here a random test [23] over our results and could conclude the performance attained by the three different configurations are statistically the same. Comparing that performance against the naive classifier (mentioned above) whose

Table 8. Results for the Case Law Entailment Task (real COLIEE test dataset).

Algorithm	Number of estimators	Precision	Recall	F-Measure
SB+PP	250 (default)	0.238095	0.283019	0.258621
SB+PP	100	0.190476	0.226415	0.206897
SB+PP	500	0.238095	0.283019	0.258621

F-measure would be 0.0509 on the test dataset, we can conclude the difference in performance is statistically different for a significance level of 0.01. The F-measure for the similarity based approach with a Gradient Boosting classifier and data augmentation was 0.12698, much lower than what we had on the validation dataset. That probably was due to a model over-fitting.

3.4 Future Work

The results presented above indicate this task has a high degree of difficulty, and show the methods we used to tackle the problem must be refined. In fact, it may be more promising to devise completely new approaches. Applying word embeddings is still worthy for this task, but we need to improve the vector representations. We plan to use the options cited on Section 2.4: train a word embedding model with our own data, retrain the used model, test skip-gram and experimenting with other frameworks. It is worth considering a refinement of a Gradient Boosting classifier, which achieved the best results on our validation dataset. With more time, we can investigate and configure the model to better generalize. However, we think the real gain in performance for Task 2 will come from a better understanding of the problem. We need to study what exactly makes an entailment relationship hold between case laws and then devise more effective strategies to model that relationship.

4 Statute Law Entailment

4.1 Method

For the statute law entailment task (Task 4), we first retrieve relevant statute law(s) for each query using English data and TF-IDF method in Task 3. A query and its relevant statute law(s) are the input to Task 4.

The original bar law examinations for Task 4 in the COLIEE data are provided in Japanese and English, and our initial implementation used a Korean translation from Japanese, provided by the Excite translation tool[4]. We chose Korean because we have a team member whose native language is Korean, and the characteristics of Korean and Japanese language are similar. In addition, the translation quality between two languages ensures relatively stable performance. Because our study team includes a Korean researcher, we can easily analyze the errors and intermediate rules in Korean.

[4] https://www.excite.co.jp/world/korean/

For the statute law entailment, we used the characteristics that a legal statute consists of "general condition," "conclusion," "exceptional case," and "conclusion for the exceptional case." The query can belong to either "general condition" or "exceptional case." We filter out the cases in the statute that are not related to the input query, by figuring out where the input query belongs. We use condition/conclusion/exception detection rules defined in Kim et al. [22] as follows:

$conclusion := segment_{last}(sentence, keyword)$
$condition := \sum_{I \neq last} segment_i(sentence, keyword)$
$exception\_conclusion := \sum_{I \neq last} segment_i(sentence, exception\_keyword)$

The keywords of the condition are as follows: *in case(s), if, unless, with respect to, when, and ,(comma).* After this segmentation, the last segment is considered to be a conclusion, and the rest of the sentence is considered as a condition. (We used the symbol $\sum$ to denote the concatenation of the segments). We also distinguish segments which denote exceptional cases. Currently, we take the exception_keyword indication as *this shall not apply, if (unless).*

We determine if a query belongs to the general case or exceptional case by computing the proportion of shared words between query and condition, and query and exception_condition. We collect words from the detected condition (exception_condition) and conclusion (exception_conclusion), and also detect negation value for each condition (exception_condition) and conclusion (exception_conclusion): The negation level (neg_level(segment)) is computed as following: if [negation + antonym] occurs an odd number of times in the segment, its negation level is 1. If [negation + antonym] occurs an even (or zero) number of times, its negation level is 0. We consider the entailment holds if neg_level(condition of the relevant article) + neg_level(conclusion of the article) is the same to neg_level(condition of the query) + neg_level(conclusion of the query). Otherwise, we consider the entailment does not hold. To detect negation and antonym, we manually constructed negation/antonym dictionary from the training data.

4.2 Experimental Results

The dry run data size of COLIEE 2018 is 644 queries for Task 4 from the Japanese bar exam from 2006 to 2016, and the formal run data size is 69 queries from the bar exam of 2017. Our performance of textual entailment is 63.77%, which was ranked No.1 in the COLIEE 2018 competition as shown in Table 9.

We think that our approach showed the best performance in the competition because of the following two reasons: (1) negation detection and (2) structure identification of the statute law. Most of the other systems' approaches focused on the meaning capture of sentences through word embeddings or other semantic analysis, but they missed negation detection which can reverse the captured meaning. Their systems also used all words in the given sentences in order to capture meaning, but the use of all words can produce incorrect meaning, because statute law describes both of the general case and exceptional case. It should be analyzed which case a given query belongs to. Then, the other case which a given query does not belong to should be removed in the semantic analysis process. We think these two approaches that we used are the primary reason of our performance, and we plan to investigate more sophisticated methods for these approaches in future work.

Table 9. Experimental results on formal run data of COLIEE 2018

Participating Team	Accuracy(%)
UA(our team)	0.6377
KIS_Frame	0.5652
KIS_mo3	0.5507
KIS_dict	0.5362
KIS_SVM	0.5217
KIS_Frame2	0.5072
UE	0.4783

5 Conclusion

The similarity based approach used for Tasks 1 and 2 provided reasonable results on the COLIEE testing datasets. In Task 1, we ranked 4th place, whereas in Task 2 we ranked 1st place among all teams in COLIEE 2018. However, the absolute scores achieved show our method still needs to be improved before it could be used in real world applications. For the statute law competition, we proposed a method to answer yes/no questions from legal bar exams related to civil law statutes. We performed legal segmentation of the statute law, and used negation/antonym information for textual entailment. The performance of our system was ranked 1st place in Task 4 of the COLIEE 2018 competition.

Acknowledgments

This research was supported by Alberta Machine Intelligence Institute (AMII), and would not be possible without the significant support of Colin Lachance from vLex and Compass, and the teams from Ross Intelligence and Intellicon.

References

1. Lee, M. D., Pincombe, B., Welsh, M.: An Empirical Evaluation of Models of Text Document Similarity. In Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 27, pp 1254–1259 (2005).
2. Baeza-Yates, R. A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Boston, MA, USA (1999).

3. Kumar, S.: Similarity Analysis of Legal Judgments and applying Paragraph-link to Find Similar Legal Judgments. Master thesis. Hyderabad, India (2014).
4. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Over-sampling Technique. Journal of AI Research, v. 16, 321–357 (2002).
5. Sun, Y., Kamel, M. S., Wong, A., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, vol. 40 (12), 3358–3378 (2007).
6. Breiman, L.: Random Forests, Machine Learning, 45(1), 5–32 (2001).
7. Friedman, J. H.: Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, vol. 29, pp. 1189–1232 (2000).
8. Zhou, Z., Feng, J.: Deep Forest: Towards An Alternative to Deep Neural Networks. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 3553–3559 (2017).
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations of Words and Phrases and Their Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, vol. 2, pp 3111–3119, Lake Tahoe, NV, USA (2013).
10. Kusner, M., Sun, Y., Kolkin, N.I., Weinberger, K.: From word embeddings to document distances. In Proceedings of the 32nd International Conference on Machine Learning (ICML), pp 957–966 (2015).
11. Supreme Court of Canada, scc-csc.lexum.com. Accessed on Sep 4th, 2018.
12. Google Pre-trained Word Embeddings Model, https://code.google.com/archive/p/word2vec/. Last accessed on Sep 4th, 2018.
13. Pennebaker, J. W., Booth, R. J., Francis, M. E.: Linguistic Inquiry and Word Count: LIWC [Computer software]. Austin, TX: LIWC.net. (2007)
14. Tausczik, Y. R., Pennebaker, J. W.: The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. Journal of Language and Social Psychology 29(1), pp. 24–54 (2010).
15. Online-Wörterbuch Englisch-Deutsch, www.dict.cc. Accessed on Sep 4th, 2018.
16. Pennington, J., Socher, R., Manning, C.: GloVe: Global Vectors for Word Representation. Empirical Methods in Natural Lang. Processing. pp. 1532–1543 (2014).
17. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, vol. 5, pp. 135146 (2017).
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In Proc. Int. Conf. on Learning Representations (2013).
19. Manevitz, L. M., Yousef, M.: One-class Svms for Document Classification. The Journal of Machine Learning Research, vol. 2, pp. 139–154 (2002).
20. Le, Q., Mikolov, T.: Distributed Representations of Sentences and Documents. In Proc. of the 31st International Conf. on Machine Learning, Beijing, China (2014).
21. Valvoda, J., Ray, O.: From Case Law to Ratio Decidendi. New Frontiers in Artificial Intelligence. pp. 20–34. Springer International Publishing (2017).
22. Kim, M.-Y., Xu, Y., Lu, Y., and Goebel, R. Question Answering of Bar Exams by Paraphrasing and Legal Text Analysis. In Lecture Notes in Artificial Intelligence (JSAI International Symp. on Artificial Intelligence), Vol.10247:299-313 (2017).
23. Yeh, A. More Accurate Tests for the Statistical Significance of Result Differences. In Proceedings of the 18th Conference on Computational Linguistics - Vol.2:947–953, ACL-COLING'00. Germany (2000).

Responses to Reviewers' Comments

Here we summarize how the reviewers comments were addressed in our paper, and explain why we could not address some of the comments.

Reviewer 1

These were the comments and suggestions by Reviewer 1:

- *"As the data is imbalance, answering all the same (No) would be a good baseline, what will be the score for this baseline?"*
  This has been addressed on the last version of our paper for task 2 (page 12). Since already had a benchmark for task 1 using a simple bag of words + cosine similarity approach, we thought this would not be necessary for task 1. However, we have now added the comparison to the naive all cases are noticed classifier, also for task 1. Notice that we cannot consider the other naive all cases are not noticed classifier, as suggested by the reviewer, because in that case, recall and precision would be zero.

- "In the 'Method' section, it is easier to read if the contents are split into method itself and the process why that method is created."
  We chose to keep them together in one section to save space.

- *"Explanation of LIWC, not just citation, would help readers."*
  We already had a brief explanation of LIWC in the last version, but we expanded it in the new version being submitted. Unfortunately we cannot add much more detail; LIWC was used in only one experiment of one of the three models described in the paper and we need to save space for the other techniques.

- *"Regarding the statute law (Task3/4), they should have used the same method as the previous COLIEE 2017, but not clearly stated. The details for Task 3/4 are not described in this paper so authors need to add details here, e.g. what sort of resources/tools were used. It is even not described which language version (English/Japanese) of the organizers' data was used."*
  The description of the tool, language that we used, and more detailed method description for Task 3/4 has been added in Section 4.1.

Reviewer 2

These were the comments and suggestions by Reviewer 2:

- *"In task1, they do not make a good result but they do not show why their result is not so good. They should analyze their failure more deeply. In task2, they are in the first place in the competition but they do not show that their*

result is statistically meaningful. There might be possibility that the other methods are beaten just because of coincidence."
It is true that we do not have a very deep analysis of our results, and that is because we do not have much space available. We had to describe and analyse methods for 3 different tasks, with training and testing datasets for each. So we had to sacrifice the results analysis in favor of a clearer and in depth description of the models. In the current version, we also added a diagram of our pipeline (as requested by two reviewers previously) to improve the models explanation. The separation of training and testing datasets is done by the competition organizers, so the competitors do not optimize their systems to a specific training dataset; thus improving the chances differences in performance by different teams in the testing dataset do not happen by chance. That is probably why none of the papers in COLIEE have included statistical significance tests in any of the competition editions so far. Nevertheless, this would be an interesting addition to our paper and we have included a statistical significance analysis based on the recommendations of http://www.aclweb.org/anthology/C00-2137. We could not compare our performance against other competitors because the appropriate test for the F-measure requires we have not only the scores but also the actual output of the models being tested, so we compared our model results against the naive baselines (and against our own models different configurations reported in the paper).

– *"In task4, it seems that they use the same method as before and I am not sure that the reproduction of their method is worth. In stead they should show why their method is very successful."*
  The discussion of the reason why our method showed best performance has been added in Section 4.2.