# Extracting Legal Information from Patent Cases

Final Project Report for Applied Natural Language Processing (INFO 256)
May 17, 2019

**Chun-fan Liao**
University of California, Berkeley
School of Law
ciliao@berkeley.edu

**Kaifei Peng**
Wuhan University
Hongyi Honor College
pkf123@berkeley.edu

## 1 Introduction

Precedents, or prior court decisions that are related to the legal cases at issue, are crucial for both parties. To prevail in the legal disputes or litigation, it is necessary and universal for legal professionals to search for precedents that are in favor of their clients' standings. However, legal cases, such as patent cases, have become more complicated nowadays, and attorneys need to spend more time seeking the right precedents that are both strongly relevant or similar to the current case at hand and that are advantageous to their clients, which has concurrently cost clients considerably as time elapses. Therefore, finding relevant cases both faster and more accurately is paramount to legal industry.

There does exit legal information retrieval database in the market, such as Westlaw, LexisNexis, or Google Scholar, to aid legal professionals with keywords search and result filtering capabilities. However, only most relevant paragraphs of each case to the specified keywords are displayed in the search result page, where many essential information are absent, such as court ruling or key concepts for each case, which, taking patent cases as example, may be the most important patent technology terms or patent functionality description.

In this paper, we try to address this issue by utilizing information extraction (IE) methods on elected patent cases, to extract useful legal information out of court decision texts. There are basically two types of information of interests: the regular expression-related ones, such as party identification, court names and court rulings, and the NLP-related ones. such as key patent terms and key patent description sentence. After collecting court decision dataset and manually annotating each of those cases, we conduct data preprocessing on the texts, then we apply regular expression rules and perform supervised and unsupervised learning methods to extract required information. After that, we use *ranking* and *matching percentage* metrics to evaluate the performance.

## 2 Related Work

For IE on legal texts and court documents, a hybrid of methods are often combined to achieve the purpose. (Moens et al., 2007) detects arguments in legal texts by viewing it as a text classification task, which is solved by selecting rigorous text feature and applying multinomial naive Bayes classifier and maximum entropy model. (Biagioli et al., 2005) recognizes provisions mentioned in law documents by text classification using Multiclass Support Vector Machine (MSVM). Further information relevant to the specific detected provision type was extracted by frame-filling based on shallow syntactic parsing and semantic annotation with a finite-state compiler. (Lee, 1998) extracts specific criminal events with detailed information and adopts template filling methods.

In (Bruninghaus and Ashley, 2001), specific names and products were replaced with general roles and general domain-specific information terms, and actions of roles were captured. Then, transformed texts were used as the input to classification rules, and see if a specific *Factors* can be matched as the binary output.

In (Justeson and Katz, 1995), a grammatical structure is introduced to capture more frequent multi-word phrases within a document, which can be implemented using regular expression. In (Bommarito et al., 2018), *LexPredict* introduces *LexNLP*, an open source package on NLP for legal texts. It claims to be capable of not only managing NLP routines, such as document segmentation or IE, but also building models for word embedding or tagging. However, it cannot correctly segment some types of legal terms, such as patent claim citations or case citations.

(Dozier and Haschart, 2000) illustrates the structure of documents in the dataset and implements a personal name extraction system with the *WestLaw* dataset. The system can segment the *Synopsis* of the document into paragraphs with different functions and thus locate in which paragraphs the names to be extracted reside. After that, each paragraph is parsed and rules are applied to extract names.

## 3 Data

We have so far annotated 220 patent-related cases of *US Court of Appeals for the Federal Circuit* (Federal Circuit Court) from the WestLaw database that are particularly related to *prior art*, the previously publicized invention or knowledge. The reason why we specifically select cases from the Federal Circuit Court is because except the Supreme Court, it is the most important court regarding patent litigation with abundant cases, and decisions from Federal Circuit Court can be readily served as essential precedents that can bind lower district court rulings.

The dataset is split as 70% / 15% / 15%, i.e. 154 cases in the training set, 33 cases in the validation set, and the 33 remaining cases in the test set. We have selected cases which verdicts have a clear inclination toward the plaintiff(s) or the defendant(s), so that the case outcome can be more easily classified.

## 4 Method

### 4.1 Scopes: Information Extraction Template for Patent Cases

We introduce an IE template for each patent case of Federal Circuit Court that we collected. The objective of our work is to fill out this template to list useful information for attorneys, which may later be integrated to other legal case retrieval systems.

Take *Amazon.com, Inc. v. Barnesandnoble.com, Inc., 239 F.3d 1343 (Fed. Cir. 2001)* as an example, the filled template is expected to be as follows:

1. **District Court Ruling**: In favor of plaintiff
2. **Circuit Court Ruling**: In favor of appellant: Vacated and remanded
3. **Key Phrases**: preliminary injunction, prior art, single action, shopping cart, server system, client system, purchase order, web site
4. **Patent Terms**: 1–Click, single action ordering, client/server environment
5. **Patent Summary**: "Amazon's patent is directed to a method and system for "single action" ordering of items in a client/server environment such as the Internet."
6. **District Court**: United States District Court for the Western District of Washington
7. **Plaintiff**: Amazon.com, Inc.
8. **Defendant**: barnesandnoble.com, inc., and barnesandnoble.com llc
9. **Appellant**: barnesandnoble.com, inc., and barnesandnoble.com llc
10. **Appellee**: Amazon.com, Inc.

Those items are self-explanatory. What worth mentioning is the *Key Phrases*, *Patent Terms*, and *Patent Summary*. *Key Phrases* generally list critical legal issues or other essential technical terms involved for the case. *Patent Terms* are the specific terms used to denote the patent technology. *Patent Summary* is the most important sentence(s) in the judge opinion texts to describe the functionality of the patent at issue.

### 4.2 Extracting basic information with regular expression

Fortunately, all documents in our dataset begin with a structured opening information, described below, from top to bottom:

1. Case number of the document is listed on the top of the whole text;
2. The title of the document, which is in format *PARTY_A v. PARTY_B* where *PARTY_A* and *PARTY_B* both provide the name as well as the role (plaintiff or defendant) of both parties involved in this case;
3. The date of the court decision;
4. The first appearance of the court name in the following content is the full name of the district court, which with the united states district court", making it easy to extract;
5. The final verdict of the case is a short single paragraph limited to 1-5 words and several specific words such as "affirmed", "reversed" and "vacated", etc..

Since 1. to 5. listed above all appear in a fixed order and comply to obvious patterns, they can be extracted using simple text/string processing programs and regular expressions.

### 4.3 Preprocessing of texts before extracting patent terms and key sentences

#### 4.3.1 Removing raw annotations

Raw text files retrieved from Westlaw's website contains needless annotations:

1. Captions and figure number of figures in raw .pdf documents (e.g. "*1394", "Fig. 4");
2. Some sentences are annotated with superscripts (e.g. "... the 579 patent is invalid.1");
3. Some words are partially enclosed in square brackets (e.g. "[w]here", "[a]ccording"). These square-bracket-enclosed words are originally used as indicators of law-related citations.

It is necessary to remove annotations listed above to keep the data as clean as possible and ensure the quality of upcoming tokenization and sentence segmentation tasks. We again use regular expressions to find these annotations and remove them. Annotations that are difficult to detect are manually removed.

#### 4.3.2 Tokenization and sentence segmentation

We have utilized *SpaCy* for most NLP-related part of this project. Nevertheless, court decision documents in our dataset contain peculiar domain-specific tokens such as law numbers (e.g. "col 1. ll. 12-34"), case numbers (e.g. "239 F.3d 1324"), trademarks (e.g. "(r)") and abbreviations of organizations (e.g. "Lab.", "Inc."), etc. SpaCy's default tokenizer becomes insufficient in handling these special cases. Thus, it is vital to add special cases mentioned above to make a more robust tokenizer.

In terms of sentence segmentation, several customized rules are embedded into the spaCy pipeline to tackle corresponding tricky problems. Some important ones are listed below:

1. A sentence was broken into several parts (lines). This can be solved by detecting whether a sentence ends with a period. If not, the sentence is deemed not complete and should be merged with next *sentence piece*. It is not necessary to consider other punctuations such as "!" and "?", since court decision documents are relatively objective and seldom show sentimental inclination.
2. Exceedingly long sentences are split with ":".
3. If a sentence contains unpaired brackets or quotation marks, this sentence should be merged with next *sentence piece* until the brackets or quotation marks are correctly matched. This is crucial since court decision documents tend to include a large amount of explanatory information enclosed in brackets.
4. A line in raw text isolated by two blank lines beside it is regarded as a single sentence, even if it might not end with a period or is a grammatically incomplete sentence.

#### 4.3.3 POS tagging and dependency parsing

The raw text is preprocessed through spaCy nlp pipeline as well as POS tagging and dependency parsing. The output documents of spaCy pipeline are stored on disk for downstream tasks, i.e. patent terms and key sentences extraction.

#### 4.3.4 Annotating the patent terms and key sentences (annotating answers)

Since we use supervised learning (e.g. logistic regression), we need to manually establish the golden standards (answers) of patent terms and key sentence of each document. Prior to annotating answers, it is necessary to develop a set of annotating rules such that there is a latent consistency within answers to each documents.

For noun phrases to be considered as a patent term, the following criteria is used:

- Is the noun phrase in the elected key sentence?
- Is the noun phrase in the first sentence of the Synopsis (the first part of the court decision that summarizes the case)?
- Does the noun phrase occur at least 2 times?
- Is the noun phrase enclosed by quotation marks?
- Is the noun phrase followed by the symbol "(r)" (copyright)?
- Does the noun phrase comprise all capitalized characters?
- Are first characters of the noun phrase all capitalized?
- Is the noun phrase part of a specific syntax, e.g. enclosed by "an" and "comprising" or following "directed to/toward"?

What worth mentioning is for each document, we annotate two patent terms. If a term can be

decomposed into shorter but similarly meaningful noun phrase, additional patent term is derived. For example, if there is a patent term "two-way communication device", another patent term "two-way communication" is also derived, to increase the hit chance.

For sentence to be considered as a key sentence, the following criteria is used:

- Is the sentence similar to the first sentence in the Synopsis?
- Does the sentence start with the noun phrase "The 'NUMBER patent"?
- Does the sentence contain specific syntax like "directed to", "relates to", "drawn to", etc?
- Is the sentence served as a functionality description of the patent?

### 4.4 Extracting patent terms: Supervised key phrases extraction

In this work, extracting patent terms from a document is viewed as a supervised key extraction task. Candidate noun chunks (phrases) are picked out using certain pre-defined filtering strategies. Next, a model is run over all candidate noun chunks and a score is generated for each noun chunk . For this part both TF-IDF model and logistic regression model are utilized.

#### 4.4.1 Finding candidate noun chunks

Since we assume that the patent terms appear in former parts of a document, we only take first 40% of noun chunks and abandon the rest. Then, we filter out organization names, law numbers and other useless terms using an exclusion dictionary. Additionally, if two noun chunks are identical after lemmatization, they are regarded as the same type. At this point we have acquired a relatively clean set of candidates.

#### 4.4.2 Evaluating candidate noun chunks with TF-IDF scores

For each candidates noun chunks, we calculate the average TF-IDF score of its valid component words (words without a specific self-defined stopword set are considered "valid").

One variation here is that we modify the original TF-IDF formula as following:

$$\alpha\beta\text{–}tfidf = tf^{\alpha}(log(idf))^{\beta}$$

Compared to original tfidf formula, here we add two exponential power on both the *tf* term and *log(idf)* term. The basic ideas are:

1. We want to penalize the terms with exceedingly high term frequency. During the tests, we discovered that terms extracted with exceedingly high score are indeed topic-related terms but not the exact patent term we desire. For instance, in *Amazon.com, Inc. v. Barnesandnoble.com, Inc., 239 F.3d 1343 (Fed. Cir. 2001)*, "shopping cart" ranks as the top term, which is indeed topic-specific and related to the exact patent name "1-Click(r)". Intuitively, we need alleviate the phenomenon that a frequent term "drowns" the true patent terms. As a result, $\alpha$ is set to be less than or equal to 1 to reduce the impact of tf term.
2. We want to increase the strength of penalization on terms that appear in multiple documents. Intuitively, the patent term we are extracting is a domain-specific term that best describes the main focus of document, which is thus supposed to appear in other documents as few times as possible. As a result, $\beta$ is set to be greater than or equal to 1 to enhance the influence of *log(idf)* term.

The results of permutating $\alpha$ and $\beta$ are shown in Part 5.

#### 4.4.3 Evaluating candidate noun chunks with logistic regression

We incorporate the benefits of supervised machine learning approaches into this task by applying a logistic regression model. For each candidate noun chunks, we extract the following features:

- The offset from the beginning of documents
- The tfidf score of its root word divided by the average tfidf score of its valid component words
- The POS tags of its root word and its head word in dependency tree
- Whether the term contains the trademark "(r)"
- Other word shape information (e.g. capitalized, containing digits, etc.)

After extracting these features, a logistic extraction model is trained using these features as data and annotation introduced in 4.3.4 as labels. With this model, we can evaluate a confidence score for each noun chunks to be the true patent term.

| Filter size | Filter number |
|---|---|
| 2 (for bi-grams) | 10 |
| 3 (for tri-grams) | 5 |
| 4 (for 4-grams) | 2 |

Figure 1: CNN structures used

### 4.5 Extracting key sentences

Using spaCy pipeline and a customized sentencizer, we can obtain the sentence set. Similar to extracting patent terms, for each sentence we evaluate a confidence score for it to be the key sentence.

#### 4.5.1 Logistic regression

A logistic regression model using featurized sentences as input is implemented for this part, the features of a sentence include:

- Dicretized sentence length (dicretized range: 0-5, 5-10, 10-20, 20-50 and 50+);
- The;
- Average TF-IDF score of sentence's valid component words;
- Maximum value of TF-IDF scores for all noun chunks in the sentence;
- Whether the sentence contains trademark "(r)";
- Whether the sentence contains patent id numbers; and
- Whether the sentence contains specific verbs (e.g. "claim", "describe") which serve as indicators of target sentence.

The training and predict processing is similar to Part 4.4.3 and the result is presented in Part 5.

#### 4.5.2 CNN method

We have also tried Convolutional Neural Network (CNN) for text to extract key sentences. The structure of CNN used is illustrated in Fig. 1.

Firstly, each sentence is run through a embedding layer (using pretrained *gloVe* embeddings) and is transformed into a sequence of word embeddings. Next, a convolution layer is added and filters of various sizes are designed for detecting bi-grams (e.g. "... patent claims ..."), tri-grams (e.g. "... the assignee of ...") and 4-grams. After convolution, a maxpooling layer is used to reduce the hidden representation vector into a fixed size. Finally a dense layer and is added to calculate the final score for a sentence.

As discussed, 154 documents are used to train, 33 for validation and 33 for testing. We perform a early stopping strategy to reduce the effect of overfitting.

From the result presented in Part 5, we conclude that CNN is able to extract indicating patterns for a sentence to be a key sentence. However, in general CNN experiments we try suffer from overfitting problem and since the training set in this project is relatively small, the result is relatively unstable and varies a lot across different train/valid/test sets.

## 5 Discussion

### 5.1 Metrics

In this work, two metrics are utilized:

1. **Average ranking**, which means the rank of the gold standard within all extracted patent terms or key sentences. The smaller the rank, the better the result. For example, if the computed rank of a patent term is 2, the score of that specific term is ranked the second place.
2. **Percentage of terms and sentences lied within top 5**, which means among all the training/test data, how many of them has the score of golden standard lied within top 5 of the extraction results. The greater the percentage, the better the result.

We use those two metrics instead of traditional ones such as precision or recall, not only because we are not conducting binary classification, and it is rather difficult to retrieve the gold standard's score as rank 1 in every document.

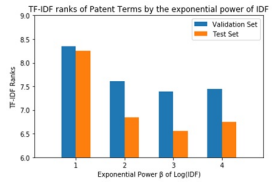
### 5.2 Permutating $\alpha$ and $\beta$

We have tried different numerical $\alpha$ and $\beta$ to try to yield better ranks and top-5 percentages. We have tested $\alpha$ value as 0.3, 0.4, 0.5, 0.6, and 0.7, and $\beta$ value as 1, 2, 3, and 4. The result are shown in Fig. 2 and Fig. 3.

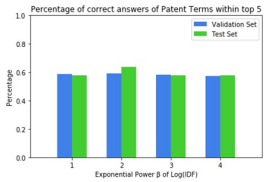
As shown in the figures, best result is retrieved when $\alpha$ equals 1 and $\beta$ equals 3.

### 5.3 Result and Analysis

For patent terms, by using $\alpha$-$\beta$ TF-IDF method, the average rank is 6.563, and the top-5 percentage is 57.6% (when $\alpha$ equals 1 and $\beta$ equals 3);

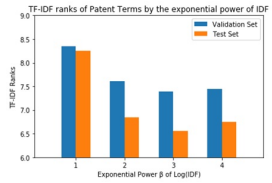



(a) TF-IDF ranks of different $\alpha$

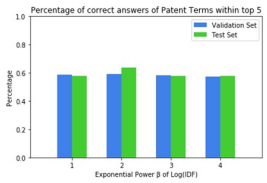



(b) Top-5 percentage of different $\alpha$

Figure 2: Permutating $\alpha$




(a) TF-IDF ranks of different $\beta$




(b) Top-5 percentage of different $\beta$

Figure 3: Permutating $\beta$

by using the logistic regression, he average rank is 7.303, and the top-5 percentage is 69.7%.

For key sentence, by using logistic regression, the average rank is 6.484, and the top-5 percentage is 66.7%; by using CNN, he average rank is 6.032, and the top-5 percentage is 54.5%.

The result is not very appealing. The major reason is not only the subject matter is a difficult task, but also the annotator's subjectivity matters. In this work, only one of the author acts as the annotator, which might severely effect the performance due to annotator's subjective preference. If there can be more than one annotators with a set of consistent labeling rule, the subjectivity problem can be mitigated.

Furthermore, complex neural network algorithms does not necessarily perform better. As for the key sentence extraction, TF-IDF method performs better than CNN on both metrics. This may partly due to small training set we have. If there is more training data, the supervised methods may yield better result.

## 6 Conclusion and Future Work

In this work, we utilized information extraction techniques to extract important information of patent cases from the Federal Circuit Court. We used both rule-based regular expressions to extract more evident information and supervised machine learning methods as well as modified TF-IDF formula to retrieve more latent but equally essential information. Due to the challenging and subjective characteristics of the problem as well as limited training data, there is room to improve.

Potential future work includes using *bidirectional LSTM* and *attention* on sentence extraction. Also, essential court opinion sentences are also crucial to legal professionals, which may be extracted using similar steps introduced in this work. Last, we only extract one sentence for each case as the representative key patent function description, but the length of the sentence varies significantly. A better method is to conduct *case summarization*, to extract similar length of contents, say 50 words, for each case. It may consist of more than one extracted key sentence.

To stimulate future works, we have released our source codes and data for public use: `https://github.com/ponedo/legal_IE`.

## References

Carlo Biagioli, Enrico Francesconi, Andrea Passerini, Simonetta Montemagni, and Claudia Soria. 2005. Automatic semantics extraction in law documents. In *Proceedings of the 10th international conference on Artificial intelligence and law*, pages 133–140. ACM.

Michael J. Bommarito, Daniel M. Katz, and Eric M. Detterman. 2018. Lexnlp: Natural language processing and information extraction for legal and regulatory texts. *arXiv preprint arXiv:1806.03688*.

Stefanie Bruninghaus and Kevin D. Ashley. 2001. Improving the representation of legal case texts with information extraction methods. In *Proceedings of the 8th international conference on Artificial intelligence and law*, pages 42–51. ACM.

Christopher Dozier and Robert Haschart. 2000. Automatic extraction and linking of person names in legal text. In *RIAO*, pages 1305–1321.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identication in text. *Natural Language Engineering*, 1(1):9–27.

Richard Lee. 1998. Automatic information extraction from documents: A tool for intelligence and law enforcement analysts. In *Proceedings of 1998 AAAI Fall Symposium on Artificial Intelligence and Link Analysis*, volume 23. AAAI Press Menlo Park, CA.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. *Natural Language Engineering*, pages 225–230.