# Building a Paralegal Bot

Chris Han – Attorney

Anthony Chi – Data Scientist

**Last Updated: August 26th, 2018**

## **Inspiration and Motivation**

## It is without a doubt that the legal profession is quite a time/effort intensive profession. Other than the attorneys deriving legal solutions for their clients, there are many behind-the-scene tasks that compose a significant amount of time. For instance:

## Respond to servicing inquiries

## Drafting legal documents

## Research and reference, including legal precedence, judges and opponents, legal language and interpretations, etc

## Scheduling, contacting, and responding to clients

## Even though these tasks do not often see the light of day, they are crucial. Attorneys practicing at small firms are most likely impacted as they don’t have dedicated personnel responsible for these mundane and time consuming tasks. Therefore, we want to leverage current technologies to help lawyers become more effective and efficient.

## In this project, we are going to work on three main areas:

## 1. E-mail scraper: this feature will scan new items in the inbox, and determine the most suitable course of action for that email.

## 2. Chat bot: this feature will be potential client facing, responding to common questions

## 3. Contract generator: base on intent and purpose, this feature will come up with draft of legal documents for the attorney to review and revise.

## These features will require iterations of use and feedback to become better. As more data is collected, these features will make ‘guesses’ more confidently.

## **E-mail Scraper**

## It is quite a task to scan new items in the inbox and determine the most suitable course of action. We will begin with a small and controlled environment, such as court distribution of docket texts. After we feel like we have a good handle on this data, we’ll then move on to the more general texts inside of attorneys’ inboxes.

### Project Overview

1. Docket texts are public court records that detail legal case events in a chronological order. New texts/records are available when the case status has been updated by the court. These new texts are sent to the responsible attorneys, often via email. Lawyers may need to take action base on these new records. For a large law practice, there may be dedicated personnel to take action. However, for smaller operations, lawyers spend a large portion of their day on these docket texts. Regardless of size of practice, practices ideally would want to automate the docket text handling processes.
2. The As-Is process for many law practices, especially smaller practices: lawyers spend time reading through docket text emails, and determine if it is necessary to take action, and what action to take.
   * 1. The To-Be process: an automated algorithm scans the newly received docket texts and determine whether or not to take action. If action is necessary, the algorithm would also take the appropriate action such as draft emails, schedule appointments with the clients, etc.

### Problem Statement

1. Ideally, this should be a supervised classification exercise. For every docket text, we would have corresponding targets indicating how attorneys should act. However, we do not have access to a dataset with docket texts and targets, and neither do we have resources to go through the docket texts and label them. We will have more update on this in a different section.
2. Instead of performing a supervised classification exercise, we tried an unsupervised classification exercise. More specifically, topic modeling. The goal of topic modeling was to ask algorithms to group similar docket texts. The groupings would then give us hints on what actions to take. We could then use available resources to determine if these groupings made sense and how we should tweak the algorithms.
3. There are many difficulties with this exercise:
4. Difficult to acquire a large database: even though the docket texts are publicly available, they are only available with a fee. The process of acquiring data means someone with access to the government court database would have to log in, search for cases that may or may not be relevant, and download the docket texts by per case basis.
5. This is an unsupervised modeling exercise: there is no target for this dataset, meaning the dataset does not come with answers on whether a lawyer should take action. Therefore, unless someone is willing to go through each texts to determine the targets, this is an unsupervised classification (topic modeling) exercise.
6. SME is required: to perform NLP or topic modeling for this exercise, we need to avoid GIGO (garbage in, garbage out). This meant that at each step, we required domain knowledge experts to interpret our input and output results:
   1. During NLP, we need to clean enough to remove all the unwanted texts, but not too much so the meanings or relevant keywords are lost.
   2. After NLP, is there an easy way to filter the features to provide the correct targets?
   3. During Topic Modeling, we need to determine the optimal number of topics/groupings, as well as interpret each topics correctly.
   4. Testing: we require domain knowledge experts to tell us whether our model is assigning topics correctly for the test dataset.
   5. After Topic Modeling, we need domain knowledge experts to tell us if any action needs to be taken, and what type of actions are relevant.
7. If topic modeling does not work, how should we proceed?
   1. We can collect more data, and try topic modeling. But this can be expensive.
   2. Make this exercise a supervised classification exercise by asking available resources to label each feature, and have domain knowledge expert inspect the labels and features.

### Metrics

1. As an unsupervised exercise, there was no right or wrong. Our domain knowledge expert would determine the correctness of our groupings. For a supervised exercise (given a labeled dataset), accuracy will be the main metric that we measure/compare our modeling results. More specifically, accuracy measures both true positives and true negatives. In our specific application, a measure that is more relevant is recall. Recall is defined by true positives divided by true positives & false negatives. To maximize recall is to minimize false negatives. We should also look at the confusion matrix to understand the false positive and false negative distributions. https://en.wikipedia.org/wiki/Precision\_and\_recall

## II. Methodology

### NLP (Natural Language Processing) – Cleaning

1. Organization and Name identification
2. Oftentimes names and organizations are irrelevant in topic modeling, as names and organizations are extremely specific to certain contexts. Therefore, I’m using the Stanford NER tagger to identify and remove names and organizations out of the docket texts.
3. Even though Stanford NER is the best NER we could find, it still did not perform a thorough job in removing all names and organizations. This means we’ll iterate through the NLP and modeling process to find more irrelevant object in our docket texts for removal.
4. Normalization
5. Normalization is a process to lower case all words, find all numbers, dates, and punctuation and replace them with relevant information. This is a step to make sure we simplify our dataset.
6. Lemmatization and POS Identification
7. Lemmatization is a process to return words back to its original form. For example, plural back to singular, and past tense into present. This will help us simplify our texts and retain meaning.
8. Parts of speech identifiers also help us identify structure of sentences and words that should be removed. It is another method to identify numbers, dates, pronouns and punctuation that were missed in previous steps.
9. Stop Words Identification
10. Stop words are words that we often use and do not have particular meaning. These are also removed from the docket texts.
11. Keywords Identification
12. From a legal practices’ perspective, if certain keywords are identified in the docket texts, attorneys are required to take certain steps. The domain knowledge experts came up with a list of keywords that our algorithm should hard code. If some text contains one of these keywords, the text should be excluded out of topic modeling considerations.
13. Phrase Modeling
14. A phrase is combinations of words that have meaning that is different from each individual constituents. It is important to identify phrases going into topic modeling, as the model will examine phrase frequencies instead of word frequencies.

**Topic Modeling**

We are using the gensim library to perform LDA (Latent Dirichlet Allocation). LDA assumes that documents are probability distribution over latent topics. Topics are probability distribution over words. LDA looks at a number of documents and assumes that the words in each document are related. It then tries to figure out the 'recipe' for how each document could have been created. We just need to tell the model how many topics to construct and it uses that 'recipe' to generate topic and word distributions over a corpus. Based on that output, we can identify similar documents within the corpus.

### In order to understand the LDA process, we have to know how LDA assumes topics are generated:

1. determine the number of words in the document
2. choose a topic mixture for the document over a fixed set of topics (ie. topic A 20%, topic B 50%, etc)
3. generate words in the document by:
   * pick a topic based on the document's multinomial distribution
   * pick a word based on the topic's multinomial distribution

### Working backwards

Suppose you have a corpus of documents, and you want LDA to learn the topic representation of K[¶](http://localhost:8888/notebooks/Docket Text Topic Modeling - Building Model (Spacy and LDA) v4.ipynb" \l "2-Topics:)

topics in each document and the word distribution of each topic. LDA would backtrack from the document level to identify topics that are likely to have generated the corpus.

### LDA's Magic

1. randomly assign each word in each document to one of the K topics
2. for each document
   * assume that all topic assignments except for the current one are correct
   * claculate two proportions:
     1. proportion of words in document d that are currently assigned to topic t = p(topic t | document d)
     2. proportion of assignments to topic t over all documents that come from this word w = p(word w | topic t)
   * multiply those two proportions and assign w a new topic based on that probability. p(topic t | document d) \* p(word w | topic t)
3. eventually we'll reach a steady state where assignments make sense

**Topic Modeling Visualization**

The goal here was to examine the available data, and achieve the following objectives:

Compare Data Sources: many data sources are becoming unavailable, turned into paid services, or only available on proprietary platforms. Many APIs were also inactivated, and webpages were inactivated or unable to be scrapped. I was only able to access three data sources: Quandl API, NASDAQ, and finance.yahoo.com.

After I compared the three data sources, there were apparent differences in Adjusted Closing Price and Adjusted Volume. There were also occasional missing data from Quandl API; the NASDAQ maximum historical data only spanned 10 years. With these difficulties, I relied soley on finance.yahoo.com’s data.

<insert the difference in data chart>

Using the available features, including open, close, high, low, volume and adj. close, I generated additional features:

* High-Low Range, as a percentage of previous day’s close: this described how volatile the day’s movements were
* Open-Close Range, as percentage of same day’s open: this described the general movement of the day’s activities
* Moving average (adjusted close & volume) for a specific time period.

The new and old features needed to answer the following questions or explore ideas:

* What am I trying to predict: specific price, change in price, simple direction, or general trend/direction? This will determine if this is a classification (bi or multi) or a regression exercise.
* Were there any relationship between the independent and dependent features?
* What did the feature distributions look like? Did I need to make any transformation?
* Were there any missing or outliers that I needed to look into?
* Did I needed to use scalers, or other feature engineering techniques such as PCA?
* Was there a model that can be applied to all equities? Or did I need to train for each equity?

### Exploratory Visualization

**Outliers**

<insert distribution plots for features>

**Missing Data**

<insert chart for missing feature>

**New Features created**

**Target**

<insert target distribution and transformation chart>

**Features Transformed**

**PCA**

### Benchmark

1. I used two benchmark models, namely XGBoost Classifier and Lasso Regression. The ultimate goal was to use LSTM to beat these benchmark models. Supposedly, LSTM should be able to incorporate long and short term memories to generate additional insights, and further improve the predictive power of resulting models.
2. During the LSTM optimization exercise, I attempted to explore how the hyperparameters such as window, epoch, batch size, and LSTM constructs affected the model performances. This surely helped me tune the model for better performances.
3. Also note this was a regression, as well as, a classification exercise.
   * 1. As the exercise became computationally expensive, I moved computational and storage needs onto AWS.

**III. Methodology**

**Data Preprocessing**

scalers

**Target**

**Outliers**

**Missing Data**

**Delete Feature**

**Feature Types Transformed**

**One-hot Features**

**New Features created**

**Features Transformed**

**PCA**

### Implementation

### Refinement

### **Lasso Regression** **(benchmark)**

### **XGBoost (benchmark)**

### **LSTM**

## IV. Results

### Model Evaluation and Validation

### Justification

## V. Conclusion

### Free-Form Visualization

### Reflection

# Findings and questions

# Questions:

# Predicting next day price may not be the best thing to do. We may want to predict price in 5 days, or even much more ahead? What’s a reasonable window to predict price before randomness takes over?

# Should I predict volume?

# Should I include other features to predict price?

# Does dropout improve performance?

# Does multiple layers improve performance?

# Should we actually try to predict price? Or should we predict a range? For example, -1~1% is neutral, 1~5% is good, 5%+ is excellent, etc.

# Findings:

# Best results is with window 15 (about 3 weeks), epoch 2000, and batch-size 50-250.

# There may be upward bias, meaning all that we have seen is upward trend (most of the time).

# When selecting hyperparameters, training predict time, testing predict time, training error eval time, and testing error eval time are negligible.

# The larger the window, the longer the training time. The higher the epoch, and the smaller the batch size, the longer the training time.

# Best training error achieved is with window 15, batch size 10, epoch 500: 2.5616e-06

# Best testing error achieved is with window 10, batch size 500, epoch 2000: 7.58083e-05

### Citation and Sources

**Relevant Files and Folders**