Introduction and Vision

The practice of law remains relatively untouched by computer technology, although it plays such a vital role in the society. One can imagine many ways in which computer algorithms and techniques can influence and change the practice of law. One area where computer technology may be very useful is in facilitating legal analysis in the common law system.

Many countries throughout the world follow the common law system developed in England. E.g. most parts of the United Kingdom, Australia, Canada, India, Hong Kong, Ireland, Pakistan, South Africa, the United States (excluding Louisiana). [1] Also called case law, common law is law created through judicial decisions, rather than legislative statutes. [2] The principle of stare decisis [2], a key feature of the common law system is largely followed in the United States. [3] Under this principle, judicial decisions serve as precedent for future decisions in similar cases. Thus, judges don't just interpret and apply the law, they contribute to it. [4]

Due to the nature of the common law system, lawyers, judges, jury members and other legal service providers need to constantly go through the routine of identifying and comparing relevant and similar cases. Each case has various aspects, interpretations and arguments. Controversy is a “built-in” characteristic of legal cases. Therefore, a system that allows computers to take case texts as input, then effectively classify cases, analyze similarity of various aspects between cases, and present different perspectives/interpretations from cases is extremely desirable. Not only will such a system save time and energy for legal workers, it can also potentially extract interpretations and perspectives that are not easily noticed or picked up by humans. Such a system is in much need because the task of extracting useful information and reasoning from such a massive amount of case descriptions will gradually expand beyond the capabilities of human beings.

- Goal

In this work, we took some initial steps towards building such a system. Our goal is to use computational methods to draw instant conclusions about a legal case. For this paper, we are going to use various methods to look at 50 cases. These cases are extremely well known; most of them are central to legal education in the U.S. We classify these input legal cases into 5 categories: 1) murder, 2) theft, 3) rape, 4) invasion of privacy and 5) product liability. In a lot of cases, these categories overlap. So we attempt to
not only assign a category to each legal case, but also be able to further analyze the similarity of cases and hence return to the users several related categories or cases so that they can further look into them. And we want to let the user know in which aspects are these cases similar.

- Data

Existing legal cases come in different formats of texts. It can be a short brief like the following:
Carroll's wife suffered a fractured skull while trying to leave his car during the course of an argument. This allegedly led to a mental disorder, which was diagnosed as a schizoid personality type. When he selected to attend an electronics school requiring him to be absent for nine days, the couple had a violent argument. Later that night he shot her in the back of the head.

Or the same case can be presented as a long description of thousands of words such as in the following link:


A lot of the cases which are not as important don’t have detailed descriptions as in the link. But each case will for sure have a short brief like the one above.

We use 50 legal cases which we chose from Casenote Legal Briefs: Criminal Law[5] and Casenote Legal Briefs: Torts[6]. Ten in each of the five categories: murder, theft, rape, invasion of privacy and product liability. We use cross-validation to test our system. We manually input all the briefs for these 50 chosen cases, which will be our training and test data for the second approach discussed in later sections. However, for some cases among these, we were not able to get a detailed description. Therefore, our training and test data for the N-gram approach mentioned in the next section is slightly less.

- Approaches

In this project, we used primarily two approaches. The first one takes long descriptions of cases as input, transform the cases into an n-gram corpus with the goal of categorizing each case into its proper category. We will highlight results that shed some light on analyzing cases using statistical methods. The second approach takes short case briefs as input, both for training and testing, and does feature extraction, leveraging bag of words and Wordnet. The potential benefit and value of the second approach is two-fold. First, as pointed out above, the majority of cases don’t have an existing clean and detailed description, or it’s not freely available. Second, the short input allows more sophisticated ways of classification and analysis to be done in the same amount of time and with the same amount of computing/storage resources. We will see that the results from this system compare favorably with the basic n-gram approach. In Part II, we will briefly touch on analyzing plot structure in legal cases using Genesis, a developing narrative processing tool.
Division of work

Jue Wang chose and manually input the legal case briefs. David Nackoul built the n-gram corpuses. David Nackoul researched the N-gram model and the plot unit method. Jue Wang developed the part of bag of words feature extraction. The rest of the paper will be organized as follows. In Part I, (a) the N-gram model will be presented and discussed (written by David Nackoul); (b) the method of bag-of-words feature extraction leveraging Wordnet will be presented and discussed; (c) we will talk about the comparison between these two approaches. Part (b) and (c) are written by Jue Wang. In Part II we will present some interesting preliminary results of the plot unit approach. Part II is written by David Nackoul. Part III will be our conclusion, discussion and future work in advancing computer assistance in the practice of law, and it is written by Jue Wang. Please direct the questions to the author regarding his/her part.

Part I. Statistical Approaches

(a) An n-gram approach to classifying legal cases.

One of our goals is to help automatically classify legal cases. Given that goal, we would be foolish not to consider n-gram analysis as a starting point. Over and over, n-grams have proven to be a valuable tool for analyzing complicated texts. Because they do not rely on any type of grammatical or semantic analysis, they can quickly and robustly crunch away at large, complicated corpora. Legal cases are a great example of large, complicated corpora.

In Lab 1, we used various n-gram techniques to classify email messages as spam or legitimate. For the purposes of our project, we elected to do something similar. In this section, we will walk you through the legal text categorizer we built and the corpora we created in order to test n-gram analysis on legal cases. But first, we will talk a little bit about what we expected to see out of an n-gram approach.

What we expect: It's all in the word

For the purposes of Lab 1, trigrams worked great. Spam emails tend to differ both syntactically and in word choice from regular emails. Consider this gem from the Lab1 spam training corpus:

“SUBJECT: Positive ? D jmmfsm xzzrej qsl ybsvkdqz cmuz uugzvogkzn xjhwiuco lfcmkvis kjfrln ct fvgvhvamyyk cc uhtgppbca dfuuetxbru swkmodttw euheoklchn pvt qktvllie rgd kwsaxlkl pzafbvwq jgerq skwqnnwftq qlfrynzbtkjolpw ryasry xnxfoyswm srnf wtrqcc vdxzbr navlibkgi izrikmgeyw ueov feorrgj owjruyslwvdwjeescbeqmjtldzxifv”

While that email is complete gibberish, there are other emails that contain words that are used in everyday email exchanges. However, these spam emails tend to use the words in an entirely different context than one would use them in a conversation. While we have definitely told friends about discounts on Amazon or asked people where to buy a home, we rarely send emails hocking discount
loans on houses. So, for the case of spam emails, trigrams reign supreme because they provide information about word choice AND context.

However, in the case of legal documents, we expect context to be less crucial. That is, we expect single words should provide a workable categorization model by themselves. Trigrams should provide some additional context; however, we expect that their value will be somewhat mitigated by the rigid structure of legalese. Consider the following “template” sentences found in multiple cases:

“These are the undisputed facts of the case.”

“These are the undisputed facts of this case.”

“At approximately 9:00PM, Matchett and Samson left the bar...”

“At approximately 7:00 A.M. Miss Peterson dressed and left the apartment...”

Just like legal phrasing, the words used to describe each crime are very systematic. In product liability cases, the defendant is usually referred to as “the manufacturer.” In assault cases where there has been drinking, parties are usually referred to as “intoxicated.” Rape cases almost always center around the issue of “consent.” We expect that these categorically common and standard wordings will make legal cases very good candidates for unigram analysis.

**Creating a corpus**

In order to run n-gram analysis, we needed to create a corpus. In order to create a corpus, we needed to decide which part of legal cases should be mined for n-grams. Most legal cases contain a section called “facts.” The facts section presents the proven aspects of the case; It usually contains the events, the places, the time-line, the proven utterances, and the official statements. We decided that using this part would make an ideal corpus because it allows us to categorize cases based on situations, not legal analysis. This way, a lawyer would be able to search for similar cases even if the case has not gone to trial yet.

Of the 50 cases used to build the briefs for the bag of words analysis, 37 contained full facts sections. Those 37 were permuted into 3 sets of corpora. The test cases in the first corpus were randomly selected, the cases in the second corpus were randomly selected from the remaining, and the cases in the third corpus were randomly and equally selected from the test cases of corpora 1 and 2. Each corpus contained 21 training cases and 16 test cases, divided into 5 categories: murder, rape, theft, invasion of privacy, and product liability. Each corpora contains 5 murder training cases, 4 murder test cases, 2 training theft cases, 2 test theft cases, 5 rape training cases, 2 rape test cases, 5 privacy training cases, 4 privacy test cases, 4 product liability training cases, and 4 product liability test cases. Ideally, we should have 4 rape training cases and 3 rape test cases, but that mistake was not caught until after corpora had been created and testing had been done. The docket numbers of each case appear in the appendix. Bracketed numbers indicate which test sets the case appears in.
In order to make each corpus, each case was manually stripped of all names and other irregularities such as website links. Names were replaced by the $NAME tag. Then, each case was run through the nltk word tokenizer. The current corpora are residing at

web.mit.edu/dnackoul/www/6.863/project/law_cases/.

Next, we modified the classifier from Lab 1. The classifier now supports multiple training corpora as well as bi- and unigram models.

It currently resides at web.mit.edu/dnackoul/www/6.863/project/legaltextcat.py. The categorizer is run with the following arguments: smoother number_of_categories [training corpora] test_files.

So, an example run may look like:

python legaltextcat.py add0.1 5 law_cases/train3/murder law_cases/train3/theft law_cases/train3/rape law_cases/train3/privacy law_cases/train3/product law_cases/test3/*

Note that there are now new options for the trainer. You can now choose bi_addl (bigrams no smoothing) bi_backoff_addl (bigrams with smoothing) and uni_addl (unigrams). Witten bell has been disabled because it is more difficult to pull meaningful inferences about the nature of legal cases from it.

The results For each corpus, we ran the classifier with 5 different probability models: add lambda unigram, add lambda bigram, add lambda trigram, back off add lambda bigram, and back off add lambda trigram. In each case, the optimal value of lambda was 0.1. The next page presents the data.
Correct Classifications (Out of 16)

<table>
<thead>
<tr>
<th>Probability Model</th>
<th>Corpus 1</th>
<th>Corpus 2</th>
<th>Corpus 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigram</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Bigram</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Unigram</td>
<td>13</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>Backoff Bigram</td>
<td>13</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Backoff Trigram</td>
<td>13</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

Corpus Size (in Tokens)

<table>
<thead>
<tr>
<th>Classification</th>
<th>Corpus 1</th>
<th>Corpus 2</th>
<th>Corpus 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>4472</td>
<td>3284</td>
<td>3475</td>
</tr>
<tr>
<td>Theft</td>
<td>4535</td>
<td>523</td>
<td>2392</td>
</tr>
<tr>
<td>Rape</td>
<td>6086</td>
<td>4397</td>
<td>6267</td>
</tr>
<tr>
<td>Privacy</td>
<td>3432</td>
<td>1700</td>
<td>3016</td>
</tr>
<tr>
<td>Product</td>
<td>903</td>
<td>1675</td>
<td>1326</td>
</tr>
<tr>
<td>Total</td>
<td>19428</td>
<td>11579</td>
<td>16476</td>
</tr>
</tbody>
</table>

Most Incorrectly Used Classifier

<table>
<thead>
<tr>
<th>Probability Model</th>
<th>Corpus 1</th>
<th>Corpus 2</th>
<th>Corpus 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigram</td>
<td>Rape</td>
<td>Rape</td>
<td>Rape</td>
</tr>
<tr>
<td>Bigram</td>
<td>Rape</td>
<td>Rape</td>
<td>Rape</td>
</tr>
<tr>
<td>Unigram</td>
<td>Product</td>
<td>Theft</td>
<td>Murder</td>
</tr>
<tr>
<td>Backoff Trigram</td>
<td>Rape</td>
<td>Theft</td>
<td>Murder</td>
</tr>
<tr>
<td>Backoff Trigram</td>
<td>Rape</td>
<td>Theft</td>
<td>Murder</td>
</tr>
</tbody>
</table>

Interpreting the results

There are a couple of items to take away from the data. First, it looks like there was not enough data to adequately support a trigram or bigram model without back off. Even accounting for the fact that
trigrams may be less useful for legal cases, trigrams should do better than 3 out of 16 correct classifications for any of the corpora. On average, random guessing should get 3.2 correct. Bigrams do slightly better, getting 5, 6 and 8 correct. In all cases, bigrams and trigrams are most likely to incorrectly classify a text as rape. This is because rape always has the biggest corpus size, and therefore contains the largest variety of bigrams and trigrams. Looking through the cases, we would expect rape cases to be, in general, longer and more detailed than other cases. Because the crime is such a delicate matter, the cases were usually much more thorough and careful in describing the facts.

The unigram case performed spectacularly, classifying 13, 8, and 15 correctly. It only underperformed the back off models in one corpus. The unigram model was most likely to incorrectly classify a text as either product liability or theft in corpora 1 and 2 respectively. These classifications had the fewest amount of tokens, so the individual probabilities of each word were inflated. Theft having the lowest amount of tokens in corpus 1 appears to be an aberration caused by the lack of factual descriptions for our theft cases. On the other hand, product liability having the lowest number of tokens is not unusual. Product liability cases tend to have a very bare bones description of the facts. Usually, they consist of what product was being used, who was using it, and how it malfunctioned. The performance of unigrams validates our hypothesis: much categorical value can be derived from legal cases by simply looking at the choice of words.

The performance of the back off models was identical. They performed the best, which was expected because they can back off to the unigram model as needed. In the cases that do not back off, the phrase is probably common enough to have a high probability. The lack of difference between back off bigrams and back off trigrams is not surprising either. Given the poor performance of trigrams on our corpora, there is likely little to be gained from them. Interestingly, the back off models inherit weaknesses from both unigrams and bi/trigrams. In the first corpus, they incorrectly classify two of the three cases they miss as rape. They even misclassify a murder case as rape that the unigram case correctly classifies. In this instance, they receive the most tri- and bigrams from the rape cases. In the second corpus, they most incorrectly classify cases as theft, inheriting the inflated probabilities from the unigram model.

The bulletproof vest: a weakness of n-grams

There is one misclassification that has not been discussed yet. This misclassification is the one case that was incorrectly classified as murder by the unigram and back off models in corpus 3. In fact, that case was never classified correctly. Upon further inspection, it is apparent what went wrong.

The case is Linegar vs. Armour of America. Linegar was a highway patrolman for the Missouri State Police. He was killed in a shootout. The bulletproof vest he was wearing left an exposed area which was penetrated during the shooting. The facts of the case mostly describe the act of murder; There's confrontation, gunfire, and wound descriptions. The choice of wording definitely suggests a murder case. However, the case is a product liability case. Linegar's widow sued Armour of America for making a faulty vest.
N-gram analysis is useful, but this case demonstrates a definite flaw. By simply looking at which words groupings were most likely, the system missed the bigger picture. No matter how much training data we had, it is very unlikely that this case would ever be correctly classified. In fact, we thought we had accidentally misclassified the case until we read the description closely. Figuring out it is product liability requires the ability to draw real meaning from a text, something that trigrams are not capable of.

The n-gram analysis corpora is in Appendix A. The items are case numbers. Brackets indicate in which corpora it is a test case.

(b) Bag-of-words Feature Extraction using Wordnet

1. Overview of bag-of-words model

Bag-of-words model assumes individual words in a text are independent. In this model, a piece of text is represented as an collection of words; each word is a feature; grammar and order doesn't matter. Each feature of this representation is the existence or number of appearances of the corresponding feature word. Bag-of-words is equivalent to a unigram model. E.g. "Carroll's wife suffered a fractured skull" will be represented as

\{'Carroll': 1, 'wife':1, 'suffered':1, 'a':1, 'fractured':1, 'skull':1\}

Due to the properties of our corpora, the two major challenges we are facing in adopting bag-of-words model here are 1) feature words selection; 2) deciding the feature value of a feature word for a given text. We cannot simply count the number of appearances of the feature words in the given text, especially because of the sparsity and noisiness of our legal case brief corpora. This leads us to the next section.

2. Corpora Properties: Sparisity, Noisiness and Wordnet

The corpora we are using consists of briefs of a variety of 50 well-known legal cases in the United Stated. They are listed in Appendix B. On average, each case brief consists 92 words. The distribution of length (number of words) is shown in the graph on the next page.

<table>
<thead>
<tr>
<th></th>
<th>Murder</th>
<th>Theft</th>
<th>Rape</th>
<th>Invasion of Privacy</th>
<th>Product Liability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Length</td>
<td>100.4</td>
<td>79.1</td>
<td>86.4</td>
<td>110.9</td>
<td>82.3</td>
</tr>
</tbody>
</table>
So our corpora in total contains about 4000 words, on the same order as the length of one case description used in the n-gram approach.

- **Sparsity**

Furthermore, these cases cover a wide range of legal issues. Even within one category, we see a large variance of details. For example, the following two cases are both in the Murder category.

Case 3. Clarence Herbert underwent surgery following a heart attack and lapsed in to a coma from which medical authorities gave virtually no chance of his recovering. He was not completely brain-dead. His family expressed a desire that he be taken off all life support equipment, including nutrition. This was done, and Herbert died. The Government then charged Barber, the attending physician, with murder.

Case 9. Ceballos lived in his apartment over his garage. One month some tools were stolen from his garage. Later, he noticed the lock had been bent and that there were pry marks on the door. Ceballos then set up a trap gun which would fire when someone opened the garage. Two boys, Robert and Stephen, both armed, pried off the lock. When Stephen opened the door, he was shot in the face by the trap gun.

- **Noisiness**

Moreover, there is large overlap across different categories. One can imagine, relatively often will rape lead to murder. The consequence of using a defective product tends to be injury or even death, which can also result from murder. In a lot of the cases where the plaintiff feel his/her privacy is being invaded,
the fact is certain media revealed his/her personal tragedy which might involve death or crimes, e.g. rape, theft. We can see here that we are dealing with extremely noisy data.

- Wordnet

With these characteristics of the corpora in mind, we clearly find the original bag-of-words approach not sufficient to fulfill our goal. So we leverage a great tool in NLTK called Wordnet to measure similarities between words. [7] Instead of simply counting the exact number of appearances of the feature word in the text, our system calculated how much the feature word contributed to the text, taking similar words in the text into consideration too.

Specifically, we use two kinds of similarity functions in Wordnet: Path Distance Similarity and Lin Similarity. path_similarity() returns a score denoting how similar two word senses are, based on the shortest path that connects the senses in the is-a (hyponym/hypernym) taxonomy. lin_similarity() returns a score denoting how similar two word senses are, based on the Information Content (IC) of the Least Common Subsumer (most specific ancestor node) and that of the two input Synsets.

Lin Similarity: 2 * IC(lcs) / (IC(s1) + IC(s2)).
Path Similarity: 1.0/(distance+1)

We mainly depend on the Lin Similarity and use Path Distance Similarity to complement it, since we don't find the performance of either of these alone satisfying. We will further discuss the problems we encountered in using Wordnet in the following sections.

3. Feature Words Selection and Extraction

The naive approach would be to consider all the words in the corpora as features. The first problem with this is that we will end up with feature vectors of length equal to the size of the dictionary. Second, this model will be extremely sensitive to new words/test examples due to overfitting. So the selection of feature words is inevitable and crucial. Here we are going to present our approach.

- Verbs v. Nouns

Calculating the Path Distance Similarity is a time-consuming task, therefore we want to reduce the number of times we have to perform that to as few as possible. One thing we noticed is that verbs carry much more useful information than nouns on average. This is natural because a crime is centered around an action. Although the subjective and tools used to commit the crime are important too, they are not as useful features in our task. So we restrain our feature words to verbs, in the sense that Wordnet can find a synset for the word with pos = VERB.

We cannot deny that certain nouns are very important in deciding the properties of a case, e.g. rape, sex. A little trick we played here is to take advantage of the fact that we do not run the text through a tagger first. So for words like rape and sex, even though their part of speech in the sentence might be a noun,
we still consider their contributions to the features of the text. Our system won't filter them out, because they can act like verbs too.

A few other nouns, which can never act as a verb, turn out to be also crucial in deciding the properties of a case, e.g. larceny, privacy, product, liability. Considering the significance of these words, we add them to the list of manually selected feature words, along with some extremely important verbs. Here is the list of the manually selected feature words in our system.

'kill', 'murder', 'larceny', 'theft', 'rape', 'sex', 'intercourse', 'privacy', 'negligence', 'warning', 'liability', 'product', 'liability'

This selection is purely based on common knowledge. Some are actually proved by the system to be useless for this corpora, due to the rareness of the word, e.g. intercourse. Some have been automatically selected by the system already, e.g. kill, murder, rape. After the system has decided on the feature words it wants to use purely based on statistical analysis, we will go through this list of pre-selected words. If a certain pre-selected word is not in the set of feature words already, we will force the system to add it to the set of feature words. But for words with a verb meaning, in the vast majority of the tests, the system already chose the word as a feature word.

- Preprocessing

In building the system, we ran into the problem of multiple meanings of a word. An example would be the word 'off'. In Wordnet, its first synset is 'murder.v.01'. He got in the way so I had him offed. But this usage will almost never appear in a legal case brief or facts description. We ran into so many example like this. Also, certain common verbs, such as 'be', 'do', 'have', 'let', 'go', 'get' and etc., are not informative in our task and will be large noises if we keep them. So we created a list of excluded words. In the stage of preprocessing, we filter out all these words, along with names of people which have verb meanings. The list is attached in Appendix C.

- How much a (candidate) feature word contribute to a piece of text

After preprocessing, for each pair of verbs A and B in the training dictionary, we calculate two metrics of similarity. path_similarity[A][B] and lin_brown_similarity[A][B]. Verbs can have multiple synsets in Wordnet. The word 'shoot' has 20 verb meanings.

```python
>>> wn.synsets('shoot', pos=wn.VERB)
[Synset('shoot.v.01'), Synset('shoot.v.02'), Synset('blast.v.07'), Synset('film.v.01'), Synset('shoot.v.05'), Synset('dart.v.02'), Synset('tear.v.03'), Synset('shoot.v.08'), Synset('photograph.v.01'), Synset('shoot.v.10'), Synset('shoot.v.11'), Synset('inject.v.03'), Synset('shoot.v.13'), Synset('shoot.v.14'), Synset('fritter.v.01'), Synset('shoot.v.16'), Synset('shoot.v.17'), Synset('shoot.v.18'), Synset('shoot.v.19'), Synset('inject.v.01')]
```
The NLTK built-in function path_similarity() and lin_similarity() take synsets as input, instead of words. So in order to measure the similarity between two verbs, we need to loop through various synsets for both A and B, and choose the pair of synsets which yield the highest similarity score. We can reduce the running time for this by limiting the loop to the first N synsets (if the verb has more than N synsets). This is valid because usually the meanings that are listed later are rarely in use. In our system, we chose N = 3.

Once we have the similarities path_similarity[A][B] and lin_brown_similarity[A][B], we can calculate score[A] denoting how much word A contributes to the entire corpora. The algorithm is as follows:

```
for each allowed word in the corpora:
   if lin_brown_similarity[A][B] > th2 and path_similarity[A][B] > th3:
       score[A] = score[A] + lin_brown_similarity[A][B]^2
```

There are four things to note here.

1) Why do we choose to use the Lin Similarity instead of Path Distance Similarity to represent the contribution to the score of words?

As mentioned before, the definition of the two similarities are as follows:

- Lin Similarity: $2 \times \frac{IC(lcs)}{IC(s1) + IC(s2)}$.
- Path Similarity: $1.0/(\text{distance}+1)$

Both of the similarities' value ranges from 0 to 1. However, distance in Path Similarity is non-negative integers. Therefore Path Similarity value can only be 1, 1/2, 1/3, 1/4, ... With verbs having many meanings, path_similarity[A][B] is not a very distinguishable measure. On the contrary, Lin Similarity has a much finer scale. E.g.

```
lin_brown_similarity['murder']['kill'] = 0.90100178174212653
path_similarity['murder']['kill'] = 0.5
```

2) Why do we use the square of lin_brown_similarity[A][B]?

We want to see decrease in contribution as the words become less correlated. We tried different orders of power. Quadratic decrease seems to work the best.

3) Implementation.

In our implementation, we do not loop through all the words in the case corpora again. We created a dictionary of allowed verbs at the beginning. Each entry of the dictionary looks like this ['kill', 3]. 3 is the number of times that the word 'kill' appears in the corpora. So in the algorithm, we actually just need to loop through all the words in the dictionary. Instead of adding the square of lin similarity, we add that multiplied by the number of times B appeared in the corpora. This speeds up the algorithm to some extent.
4) Thresholding.

Because Wordnet is a very extensive tool, it finds a lot of words correlated although their correlation is meaningless or adversarial for our training. To eliminate this kind of noise, we impose threshold on similarity. We only consider two words are correlated if their Lin Similarity is larger than 0.7 and their Path Distance Similarity is larger than 0.3, meaning the path between the two words has less than 4 hops.

- Distinguishable features

The goal of the algorithm described in the previous section is to identify the words that are important to the entire corpora covering all 5 categories. In this section, we will present our algorithm of finding the most distinguishable features across different categories. In other words, we want to find words that are crucial to a particular class of cases but are less likely to be seen a lot in other classes.

We calculate a Distinguishing Score $D[A]$ for each candidate feature words. First, we calculate the contribution score of A to each class in the training corpora. Let's say they are $S[A,'murder']$, $S[A,'theft']$, $S[A,'rape']$, $S[A,'privacy']$ and $S[A,'product']$. $D$ is defined as follows:


When the second maximum of the scores is 0, we define $D[A]$ as 100, or any sufficiently large number. $D[A]$ shows how unique word A is to the class it represents or "belongs" to. We chose the ratio of the maximum and the second maximum, because our final classification decision heavily relies on the differences between these scores.

- Feature words selection

With score[A] and D[A] defined and calculated, our feature words selection works as follows:

We sort words based on their contribution score to the entire corpora (score[A]), from high to low. Then we loop through this sorted list. For each candidate we look at their Distinguishing Score D[A]. If D[A] < 1.8, we do not consider it as feature words. Apart from this, we also make sure that each class gets relatively same number of feature words. To be more specific, since we have 13 pre-selected feature words already, we let the system choose no more than 4 feature words for each class of crimes, based on score[A] and D[A]. Some of the words the system chooses overlap with the pre-selected ones. So we in general end up with around 30 feature words in total. Our feature vectors will be about 30 dimensional.

4. Weight assignments on feature words

Different feature words are of different levels of importance to the classification task. Ideally, we would want to project the training data into the feature space and calculate the variance of each cluster in each dimension. However, this process is very computationally heavy and is hard to be implemented in
an incremental manner. In addition, with only 10 examples in each category, this may very well lead to overfitting. So instead we take the simple approach of assigning slightly higher weights to the pre-selected words (1.3 times of the weight of the statistically selected feature words).

5. Classification and metric of similarities.

We calculate the center (in feature space) of the training examples in each cluster. And to analyze a test example, we calculate the distance between the test example's feature vector and the center of each of the 5 clusters. Then we look at these distances \([\text{DIST[murder]}, \text{DIST[theft]}, \text{DIST[rape]}, \text{DIST[privacy]}, \text{DIST[product]}]\). We classify the case to the cluster that yields the shortest distance. Sometimes, we will find two distances are very similar. This is usually because the case involves multiple aspects which touch on different crimes. We will analyze sample cases like this in the next section.

6. Results and case analysis

We use cross-validation to test our system. We divide the 50 cases into 3 sets. Each set has relatively similar number of cases in each class. In each experiment, we use two sets as training data and the other set as test data. The number of words in each set in each crime class are shown in the following table.

<table>
<thead>
<tr>
<th></th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>580</td>
<td>182</td>
<td>242</td>
</tr>
<tr>
<td>Theft</td>
<td>258</td>
<td>250</td>
<td>283</td>
</tr>
<tr>
<td>Rape</td>
<td>384</td>
<td>222</td>
<td>258</td>
</tr>
<tr>
<td>Invasion of Privacy</td>
<td>334</td>
<td>266</td>
<td>509</td>
</tr>
<tr>
<td>Product Liability</td>
<td>226</td>
<td>241</td>
<td>356</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1782 (15 cases)</strong></td>
<td><strong>1161 (15 cases)</strong></td>
<td><strong>1648 (20 cases)</strong></td>
</tr>
</tbody>
</table>
Here we presented the following results: 1) accuracy on training data and test data for each experiment; 2) the ratio of the number of pre-selected feature words to the total number of feature words; 3) accuracy when using only preselected feature words. After this we will also present some similarity analysis.

### Accuracy Results

<table>
<thead>
<tr>
<th></th>
<th>Test 1 (Set 1)</th>
<th>Test2 (Set 2)</th>
<th>Test3 (Set 3)</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Accuracy</td>
<td>94.3% (2 errors)</td>
<td>91.4% (3 errors)</td>
<td>93.3% (2 errors)</td>
<td>93%</td>
</tr>
<tr>
<td>Test Accuracy</td>
<td>93.3% (1 error)</td>
<td>80% (3 errors)</td>
<td>95% (1 error)</td>
<td>90%</td>
</tr>
<tr>
<td>Overall Accuracy</td>
<td>94%</td>
<td>88%</td>
<td>94%</td>
<td>92%</td>
</tr>
<tr>
<td>#statistically selected features</td>
<td>20</td>
<td>18</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>#pre-selected features</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Using only preselected features</td>
<td>72%</td>
<td>70%</td>
<td>82%</td>
<td>75%</td>
</tr>
</tbody>
</table>

### Feature Words [word, category]

- **Test 1**: [('fired', 'murder'), ('gun', 'murder'), ('provide', 'theft'), ('stabbed', 'murder'), ('paid', 'theft'), ('warnings', 'product'), ('fabricated', 'product'), ('use', 'product'), ('raped', 'rape'), ('printed', 'privacy'), ('designed', 'product'), ('wrote', 'privacy'), ('engaged', 'rape'), ('crowd', 'privacy'), ('encounter', 'rape'), ('model', 'privacy'), ('stolen', 'theft'), ('branch', 'theft'), ('garage', 'murder'), ('fearing', 'rape'), ('killed', 'murder'), ('murder', 'murder'), ('larceny', 'theft'), ('theft', 'theft'), ('sex', 'rape'), ('intercourse', 'rape'), ('privacy', 'privacy'), ('negligence', 'product'), ('liability', 'product'), ('product', 'product')]

- **Test 2**: [('shot', 'murder'), ('accounts', 'theft'), ('guns', 'murder'), ('warnings', 'product'), ('order', 'product'), ('rape', 'rape'), ('used', 'product'), ('contract', 'product'), ('armed', 'murder'), ('tapes', 'privacy'), ('article', 'privacy'), ('cash', 'theft'), ('crowd', 'privacy'), ('pocket', 'murder'), ('program', 'privacy'), ('dropped', 'rape'), ('company', 'theft'), ('killing', 'murder'), ('murder', 'murder'), ('larceny', 'theft'), ('theft', 'theft'), ('sex', 'rape'), ('intercourse', 'rape'), ('privacy', 'privacy'), ('negligence', 'product'), ('liability', 'product'), ('product', 'product')]


Purple ones are the pre-selected feature words, which are not chosen by the system based on statistical analysis.
From the results above, we can see that, pre-selected words can achieve an accuracy of 75% on average, which is exactly the same to the unigram performance in section (a). And with training, the system picks up words like 'gun', 'stab', 'fire' for murder cases; it also picks up words like 'article', 'tapes', 'book' for privacy cases; 'room' for rape; and 'manufacture', 'design' for product liability cases. Some of these words are interesting in the sense that we didn't expect them to be crucial at the beginning of the project. From another perspective, our system can add to our common sense by statistical analysis.

**Similarity Analysis**

Here we calculated the similarity score for all pairs of cases we have. The feature words here are trained on all the cases. We will show the distribution of the similarity score within each class and across different classes. Furthermore we will choose several interesting cases to analyze and demonstrate the ability of our system.

![Similarity measurement for cases in different classes](image1)

![Similarity measurement for cases within the same class](image2)

The mean of the similarity values for pair-wise cases within the same class is 41, and the mean of the similarity values for pair-wise cases from different classes is 56. While we see a gap here, it also proves that a lot of the cases from different classes have similar elements in them. We will look at one example here, which should be an invasion-of-privacy case, however it is being categorized as a rape case in our system.

Case 33. Cohn's daughter was raped and died. A trial of the accused was held. A television reporter learned the name of the decedent from public records made available at trial. The name was reported on a news program broadcast by a television station owned by Cox Broadcasting. The father of the decedent, Cohn, brought an action of invasion of privacy.
The scores for this case calculated by our system are

[murder: 30.1, theft 15.4, rape 7.6, privacy 10.5, product 21.2].

The average distance of this case to other cases in the privacy category is 26.0867, and the distance of this case to cases in the rape category is 23.8915. Under both metrics, the scores for rape and invasion-of-privacy are very close and much lower than other categories. This makes sense because this case actually involves rape, although defendant is not charged with rape here. If we want a system to correctly categorize this case as invasion of privacy, we cannot simply depend on statistical methods. We need to take into account expressions like "be charged of", "bring an action of". We also have to capture the relationship between words. E.g. here we need to consider "brought an action of invasion of privacy" together.

7. Files

The 50 legal briefs used in this part will be uploaded to Jue Wang’s public directory and also attached in the email. Two scripts are needed to run this algorithm: train_script.py and test_script.py. The training takes relatively long time since we need to calculate the similarities between all pairs of words. The current training is done on all cases, and the test is also on all cases. The two scripts can easily be modified to do cross-validation, like we did in our experiment above. The output of the current scripts is in the file output_bag_of_words.txt. More detailed output can be enabled by setting parameters in test_script.py.

(c) Comparison of N-gram and Bag-of-words Feature Extraction using Wordnet

As discussed before, the original bag of words model is indeed equivalent to unigram. Legal cases are intricate in so many ways so the corpora we are dealing with here is extremely noisy and the designated clusters overlap very often. At the same time, also because of the intricacy of the cases, each case has its uniqueness, which means we should always expect the corpora to be sparse. We are trying to analyze similarities regardless of the uniqueness.

The n-gram approach discussed in section (b) deals with this sparsity by using much more detailed descriptions (more training data). The bag-of-words approach leveraging Wordnet tackles the sparsity problem by measuring similarities between words, and also use common sense to preselect some basic features. We can see that the number of words in the second corpora (legal case briefs) is less than 8% of the first corpora (longer case descriptions). Yet, the overall accuracy of the second approach is 90% while the overall accuracy of the first approach is around 75%, which is the same when the second approach only uses pre-selected feature words. We think the reason for this is many fold. We choose to talk about the following two reasons here.

First, the writing style of the longer legal descriptions tends to be very consistent. So an n-gram model is likely to pick up expressions not unique to the case itself, but to the entire area of legal texts. As our results proved, words are the key in classification and analyzing similarity in this scenario. And an
traditional unigram model works very much like a keyword search engine. What the second approach
tries to leverage is Wordnet’s power of identifying similar words. However, since Wordnet has an
extremely extensive collection of meanings of words, we had to find our own ways of handling noise
introduced by adopting Wordnet. Ideally, we would want a similar tool as Wordnet, which is created
specifically for the practice of law.

Part II. Genesis: Legal Case Analysis Through Plot Units

This section was toned down a bit after feedback and realization that it is outside the domain of 6.863.
However, we believe it is still interesting enough to present. This section is a proof of concept for using
the Genesis group's research to analyze legal cases.

One way to look at legal cases is through the lens of plot units. A plot unit is a recurring plot structure
that appears in multiple stories. For example, revenge is a common plot unit. It is something that is
seen in Shakespeare, foreign relations, and even law. Collectively, plot units allow us to reason about
stories at a higher level. When asked the question “Why did Jane attack Jill,” an acceptable answer
might be “Jane was taking revenge on Jill.” At this point, we do not need to know the whole story, we
already have a basis: Jill must have done something to harm Jane and Jane is taking payback. We not
only know what precipitated the attack, but we know the motivation as well.

The decisions that judges have to make can be formulated in terms plot units. Asking whether or not a
person is guilty is equivalent to asking the following question: Did this person play a specific role in a
specific plot unit. Was he the arsonist in an arson plot unit? Was he the thief in a larceny unit? That
kind of reasoning is what this section will entail. We will present a proof of concept for using automated
plot unit analysis to reason about legal cases. But first, a tour-de-force of the system.

Genesis in a page

The Genesis group's system works as follows. Text input is read into the system. That text is parsed
using the START parser. Afterwards, WordNet is consulted to transform the text into thread memory.
Essentially, thread memory is a recursive structure composed of Things, Derivatives, Relations, and
Sequences. Things are just objects, like a rock. Derivatives are words that take in one argument, like
appear (Jane appears). Relations, like harm, take in two arguments (Jane harms Jill). Sequences may
take in an unspecified amount of arguments (Such as a path that is composed of multiple stops). Each
argument may be any one of the aforementioned types. Thread memory also stores hierarchical
information about the type of the word. For example, a person is an entity but not the other way
around.

Once thread memory has been built out of a story, the StoryProcessor loops over the story, attempting
to apply commonsense rules. Commonsense rules take two forms, predictions and explanations.
Predictions are used to determine the possible effects of an event. Explanations are used to determine
the possible causes of an event. The firing of these commonsense rules allows Genesis to form a semantic graph of the story where events are linked by causality.

Plot Units

Once the StoryProcessor is finished, Genesis can begin searching for plot units. Plot units are hand coded into the system. Here's an example of a plot unit that finds instances of a Pyrrhic victory.

xx is a person.

yy is anything.

xx's wanting yy leads to yy.

yy leads to xx's becoming happy.

xx's wanting yy leads to xx's becoming unhappy.

Plot units consist of three things: the actors and their types, the events that happen, and the causal links between the events. This description forms a graph structure with the events as nodes and the causal links as directed edges. In order to match plot units in a story, Genesis looks for a subgraph match where every link in the graph is allowed to be a path in the graph of the story.
Story graph of Macbeth with a Pyrrhic victory highlighted

**Applying Genesis to legal briefs**

As a proof of concept, we wanted to apply plot unit analysis to several legal briefs. In order to do this, we needed to come up with a set of interesting plot units and relevant common sense. Because the system is in its infancy, we also needed to simplify the English in order to allow Genesis to make meaningful interpretations. We ended up choosing three murder cases, two larceny by deception cases, and two insanity cases. We will present the three most interesting cases, one from each category, in order to highlight some of the potential of plot unit analysis.

**Case 1: It's a trap!**

The first story revolves around a person named Ceballos. Ceballos created a spring loaded handgun trap in his garage because people, including Steven, had stolen items from him. Steven came back to steal more things and was shot and killed by the trap. Is that murder? Well, Genesis says that it depends on how you look at it. Genesis presents two perspective. In perspective one, there is a rule that says that a person who creates a trap is responsible for killing someone who dies from it. Perspective two does not have that rule. If you take out that rule, you cannot find a manslaughter plot unit (that is someone's action leading to another person becoming dead). This perspective analysis would be very useful for law. Perspective analysis allows us to model how entities think. In the case of legal analysis, that entity may be a judge or even a state. Imagine being able to simultaneously view a case through the laws of two different states. For example, Texas has a rule where anyone who kills a home invader is not at fault. Massachusetts does not have that rule. Applying perspective would bring us closer to our vision of facilitating understanding of the law.
Case 2: The greedy lawyer

This case is about a lawyer who jilted his client out of money. Francis was in trouble with the INS, so he hired Graham to be his lawyer. Graham told Francis he needed to pay a fine, which was true, but he took too much money from Graham. Genesis’ analysis is as follows.

Genesis finds a larceny by deception. Francis gained Graham’s trust and then stole from him.

Here Genesis uses the reasoning that because Francis employed Graham to be his lawyer, Francis trusted Graham. It follows that Graham stole from him both because he collected too much money, the physical act, and also because it was easy for him because Francis trusted him. Taken together, these events form a larceny by deception. Something happened that allowed Graham to gain Francis’ trust, and then he stole from him. Case 2 illustrates that a law case reasoning system can also deal with motive. Graham stole money because Francis trusted him and it was easy. In Macbeth, Macbeth murdered Duncan to become the king. Plot unit analysis may assist not only in answering “what happened?” but also in answering “why did this happen?”
Case 3: War of the Roses

The final case centers around a man who killed his wife because she was cheating on him. The courts ruled that he was temporarily insane. Genesis comes to the same conclusion. Genesis contains a rule that a person may become insane if their wife cheats on them. As anyone who has ever experienced infidelity will tell you, this is a very reasonable common sense rule. A second rule says that a person who is not sane may kill a loved one. Checking almost any newscast will validate this rule also. From those two rules, Genesis is able to interpret Crenshaw's actions as an act of insanity.

This case shows that a system can also make inferences about people's mental states. These states drive plot units just like they drive law. A mistake is a mistake because it put someone in a negative mental state. An act of insanity is such because a person was not in control of their actions. Rules about mental states will never be perfect; we never know how another person feels. However, that should not stop a system from making reasonable inferences. Such inferences are an important part of interpreting the law.

Part III. Conclusion

In this paper, we mainly explored two language processing approaches in facilitating legal case classification and analysis. The first one is an n-gram model trained on 37 detailed legal case descriptions. Our conclusion there is that unigram performs outstandingly well. The reason for that is individual words in legal cases are very important statistically. The second approach is a bag-of-words feature extraction model leveraging Wordnet. This approach yields approximately 90% accuracy in classifying the 50 legal briefs. When only using the manually pre-selected feature word, the accuracy is 75%. Although our system beats pure common-sense based approach here, we still find common sense extremely important. Approximately 1/3 of our feature words are chosen beforehand based on simple
common sense. More detailed discussions about the comparison of the two approaches can be found in Part I Section (c).

We took some initial steps towards facilitating legal case analysis using natural language processing techniques. There are a few things we noticed that might further improve the performance of the bag-of-words approach presented in Part I Section (b). First, running the training and test data through a tagger as preprocessing will help us eliminate words that are not verbs in the sentence. Furthermore, an intelligent parser, which can decide the synset of the verbs in sentences, is highly desirable. Second, a tool like Wordnet which is specifically designed for legal context is likely to improve the performance of this system.
Appendix A. N-gram analysis corpora

Murder:
386 Mass. 492 [2] [3]
456 Mass. 198 [2] [3]
169 Cal.App.3d 718 [2]
412 Pa. 525 [2]
461 S.E.2d 163
321 Md. 532 [1]
187 Cal.App.3d 410 [1]
12 Cal.3d 470 [1] [3]
1 Cal.3d 444 [1] [3]

Theft:
213 Md. 298 [2] [3]
132 N.E.2d 519 [2]
187 F.2d 87 [1]
877 P.2d 557 [1] [3]

Rape:
289 Md. 230 [2]
85 Misc.2d 1088 [2]
386 Mass. 682 [3]
721 A.2d 1111 [3]
129 N.J. 422
28. 312 S.E.2d 470 [1]
30. 488 U.S. 227 [1]

Privacy:
44 F.3d 1345 [2]
113 F.2d 806 [2]
420 U.S. 469 [2] [3]
385 U.S. 374 [3]
25 N.Y.2d 560 [1] [3]
94 N.Y.2d 436 [1]
251 F.2d 447 [1]
169 Misc.2d 500 [1]

Product:
24 Cal.2d 453 [2]
620 So. 2d 1244 [2]
100 N.Y.2d 38 [2] [3]
20 Cal.3d 413 [2] [3]
909 F.2d 1150 [1] [3]  
394 Mass. 131 [1] [3]  
181 F.3d 608 [1]  
529 U.S. 861 [1]

**Appendix B. Bag-of-words Feature Extraction using Wordnet analysis corpora**

**Murder:**
1. 386 Mass. 492  
2. 456 Mass. 198  
3. 169 Cal.App.3d 718  
4. 412 Pa. 525  
5. 461 S.E.2d 163  
6. 321 Md. 532  
7. 354 Pa. 180  
8. 187 Cal.App.3d 410  
9. 12 Cal.3d 470  
10. 1 Cal.3d 444

**Theft:**
11. 145 Wis. 373  
12. 213 Md. 298  
13. 132 N.E.2d 519  
14. 187 F.2d 87  
15. 305 N.Y. 864  
16. 50 D.L.R. 4th 1  
17. 2 W.L.R. 201  
18. 105 Cal. 66  
19. 877 P.2d 557  
20. 166 Pa. Superior Ct. 16

**Rape:**
21. 289 Md. 230  
22. 85 Misc.2d 1088  
23. 386 Mass. 682  
24. 721 A.2d 1111  
25. 129 N.J. 422  
26. MC v. Bulgaria  
27. 28 Md. App. 212  
28. 312 S.E.2d 470  
29. 265 NYS 284  
30. 488 U.S. 227

**Invasion of Privacy:**
31. 44 F.3d 1345  
32. 113 F.2d 806
33. 420 U.S. 469  
34. 921 F.Supp. 385  
35. 385 U.S. 374  
36. 25 N.Y.2d 560  
37. 94 N.Y.2d 436  
38. 18 N.Y.2d 324  
39. 251 F.2d 447  
40. 169 Misc.2d 500

**Product Liability:**
41. 111 NE 1050  
42. 24 Cal.2d 453  
43. Winterbottom v. Wright  
44. 620 So. 2d 1244  
45. 100 N.Y.2d 38  
46. 20 Cal.3d 413  
47. 909 F.2d 1150  
48. 394 Mass. 131  
49. 181 F.3d 608  
50. 529 U.S. 861

**Appendix C. Excluded Words in Preprocessing Stage**