Initial Steps Toward Automating Legal Document Editing

Lauren Blake ***(lblake@stanford.edu)***

Introduction

Drafting successful legal documents requires experienced lawyers, careful research, and nuanced reasoning. Nonetheless, drafting also requires many mundane and time intensive tasks that delay the process and increase its cost. One consequence of this is that law firms hire a large number of support employees to assist with these low-level tasks. According to US Bureau of Labor Statistics data released in May 2016, approximately 40% of workers in US legal services are in support roles (e.g. paralegals and legal assistants) and these workers are paid $12bn annually.

Starting by reducing time spent on editing, one of the most common low-level tasks, machine learning has the potential to significantly improve writing and reviewing legal documents. To reach this potential, many challenges need to be further understood and subsequently addressed, including how machine learning can comprehend the key points in a legal document and reflect lawyers' negotiations on those key points in its recommended edits.

Objectives

This paper initially evaluates how supervised learning algorithms could automate editing legal documents. The paper's scope is narrowed to analyzing a particular set of legal documents and edits.

The legal documents are non-disclosure agreements ("NDAs"), which are also called confidentiality agreements. NDAs are a type of private contract that determine how the participants can use sensitive information and are routinely exchanged between companies before considering business development or M&A transactions. For example, a company for sale will require potential buyers to sign an NDA before providing them with financial statements and other non-public information.

Edits are limited to the initial set of comments on a document, which consist of one participant's requests for specific text to be inserted into or deleted from an NDA after reviewing the other participant's original document. These initial requested changes reflect the negotiation between the two parties on key terms (e.g. duration of the agreement, non-solicits of employees, restrictions on buying stock, limitations on sharing information with others) and contain significantly more meaning than fixing typos or making cosmetic changes to the wording.

The long-term ideal model for automating editing would modify the original document provided by one party to reflect another party's preferred legal terms and language while maintaining the document's integrity (e.g. clear writing, proper grammar, and consistent defined terms). This would significantly reduce or even eliminate the editing done by lawyers today.

To determine what may be feasible, the models in this paper take on the relatively simpler classification task of predicting whether words in the edited NDA text come from the original NDA text provided by one party or were requested as insertions or deletions to the text by the other party. The models in this paper reverse engineer the edits that would be made by the long-term ideal model by identifying changes that were previously made to the edited document (without the benefit of seeing the original document). Conversely, the long-term ideal model would suggest changes that could be made to the original document in the future.

By using similar legal document text data and models, the classification task discussed in this paper gauges the complexities of building the long-term ideal model, highlighting the difficulties with unbalanced and sparse data, and indicates next steps forward.

Related Work

The intersection between machine learning, Natural Language Processing ("NLP"), and law has been investigated by the academic community and technology firms. Their focus has been primarily placed on information retrieval used in e-discovery software and predictive models for case outcomes. Although relatively less common, content analysis techniques are being developed, which are used to interpret public documents (e.g. court cases, laws, SEC filings) in academia and support the technology services offered by a few legal software start-ups that summarize and track private contracts. Beyond these topics, a literature review did not identify any relevant academic or industry research on specifically editing legal documents.

Looking beyond the legal context, Colgrove, Tibshirani, and Wong initiated research in machine learning for automating editing by using Naive Bayes models to predict what section of a Wikipedia article a user should edit based the user's revision history. These models were not particularly successful, providing more accurate predictions on the article instead of the article section edited by a user[1].

Despite limited directly relevant published work, there is an extensive body of research on NLP techniques to draw from. Given their shared goal of tagging tokens within text sequences, Parts of Speech ("POS") Tagging and Named Entity Recognition ("NER") have many commonalities with identifying edited text and have informed numerous decisions in this paper's analysis, including the text processing and the models used.

Data Set

The data set contains 105 NDA documents from a single investment firm, which include approximately 4,000 sentences, 2,400 unique tokens, and 202,000 tokens total. Each NDA was initially written by a company running a sale process and then edited by the investment firm.

Text processing was completed to create sentence and word tokens and remove word stems (as detailed below). Most importantly, the correct original, inserted, or deleted label for each tokens was identified based on comparing the original and edited versions of each NDA. Across the entire data set, 87% of tokens were original, 7% were inserted, and 6% were deleted.

Text Processing Steps

1. Identify the inserted or deleted portion of the text by running an advanced version of the "diff" terminal command on the original and edited version of each document. HTML tags mark the start and end of these edited portions of the text.
2. Tokenize the text into sentences with the Punkt tokenizer provided by the Natural Language Tool Kit ("NLTK") Python package. The Punkt tokenizer recognizes periods ending sentences versus those within sentences using unsupervised learning.
3. Tokenize the sentences into words using a customized regular expressions tokenizer in the NLTK package, which accommodates the HTML tags.
4. Normalize the tokens by removing suffixes with the Porter stemmer from the NLTK package.
5. Label each token as "original", "inserted", or "deleted" based on its position relative to the HTML tags.

Text Processing Example

Original Text: '...will have any legal effect. Within ten days, after being so requested...'

Text with HTML Tags Representing Edits: '...will have any legal effect. <del> Within ten days,<\del><ins> Promptly <\ins> , after being so requested...'

Sentence and Word Tokens after Stemming: [...('will'), ('have'), ('ani'), ('legal'), ('effect')][('within'), ('ten') ('days'), ('promptli'), ('after'), ('being'), ('so'), ('requested')...]

Labeled Tokens: [...('will', 'O'), ('have', 'O'), ('ani', 'O'), ('legal', 'O'), ('effect', 'O')][('within', 'D'), ('ten', 'D') ('days', 'D'), ('promptli', 'I'), ('after', 'O'), ('being', 'O'), ('so', 'O'), ('requested', 'O'),...]

Figure 1. Text Processing Overview ('O', 'I', and 'D' represent original, inserted, and deleted labels.)

Adding NER to the text processing was seriously considered and tested because NER should be able to increase the similarity between documents by replacing document-specific names with generic ones (e.g. replacing Google with Company). However, NER was ultimately not included because it did not meaningfully increase edited token classification accuracy on the training data. In part, this was likely caused by existing NER models mislabeling capitalized legal phrases (e.g. Evaluation Materials or Contemplated Transaction) as organizations or people and could be remedied by a NER model trained on a legal document corpus.

In order to maintain at least some context from prior tokens and labels, each sentence was treated as one observation. 90% of the sentences were randomly chosen as training examples and the remaining sentences were testing examples. Within a sentence, each token can potentially have a different label. Consequently, classification results were calculated based on identifying individual token labels correctly.

It is very difficult to gather NDAs because these documents are private contracts and may only be obtained from one of the signing companies. As a result, this paper's data set has a small sample size. This leads to sparsity and a limited number of inserted and deleted token examples. Furthermore, due to the imbalance between original and edited tokens in the data, model performance on the original tokens is more heavily weighted than performance on edited tokens in training unless additional adjustments to the data set are made.

Methodology

Hidden Markov Models, Structured Averaged Perceptrons, and Conditional Random Fields were evaluated based on their ability to successfully classify inserted and deleted tokens within each edited NDA sentence. These models were chosen because they are often relied on in NLP for similar tasks (e.g. POS Tagging and NER) that involve tagging text sequences.

Hidden Markov Model

The Hidden Markov Model ("HMM") is a generative model that estimates the probability of an unobserved sequence of inserted, deleted, or original label states when there is an observed sequence of word outputs. From the Markov Property, the model assumes the current inserted, deleted, or original label state and word output only depend on the prior state. Therefore, no additional features beyond the likelihood of the current word output conditional on the prior state and the likelihood of transitioning to the current state from the prior state are taken into account when the HMM makes predictions.

The supervised HMM from the NLTK package determined the maximum likelihood output and transition probabilities based on the frequencies observed for the word tokens and states in the training examples. The model subsequently tagged tokens in the testing examples based on the maximum likelihood state sequences generated with these probabilities by the Viterbi algorithm[2].

Structured Averaged Perceptron

The Structured Averaged Perceptron ("SP") uses online learning to continuously improve its original, inserted, or deleted label predictions. This discriminative model updates the weights placed on different features after an incorrect prediction and leaves the weights unchanged after a correct prediction. As commonly used in POS tagging, the features include a combination of suffixes, prefixes, entire words, and labels drawn from the two prior tokens, current token, and the two following tokens as well as a constant bias term.

The SP from the NLTK package determined the weights with 20 iterations through all of the training examples. After training was completed, average weights were calculated and tokens from the testing examples were tagged with the highest weighted label, in which prior token features were based on the SP's prior predictions[2].

Conditional Random Fields

The Conditional Random Field ("CRF") model can be considered a more expressive version of the multinomial logistic regression model that makes predictions for sequences of original, inserted, or deleted label states. Like other discriminative models, the CRF maximizes the probability of each label conditional on a rich feature set derived from the text. After testing the abundant features available, a restricted feature set was chosen which included the same set of features from the SP model along with extended sequences of words and labels.

The linear chain CRF from Stanford NLP group (primarily used for their NER classifier) finds the maximum likelihood conditional probabilities using stochastic gradient descent. The model tags tokens based on Monte Carlo inference approximations of the highest probability state sequence, using Gibbs sampling on possible state sequences[3].

Adjustments for Unbalanced Data

Most of this paper considers models trained on all training example sentence. In order to counteract the bias towards original labels from the unbalanced data set, two additional training processes were considered. The first process trained on edited sentences, discarding sentences with only original tokens from the training data.

The second process breaks the training sentences into bigrams and randomly undersamples the bigrams where the second token has an original label, resulting in an even distribution between original, inserted, and deleted labels for the bigrams' second tokens. The first token serves as context for the second token and consequently its label is not factored into the undersampling, resulting in more original labels appearing across both tokens in the bigrams. While 87% of the tokens were original in the all sentences training data, 74% and 37% were original in the edited sentences training data and the randomly undersample bigrams training data respectively.

Results

When trained on all sentences, all three models had over 90% accuracy on the testing examples driven by identifying over 98% of the original tokens correctly. Highlighting the clustering of accurately classified tokens seen in Figure 2, accuracy for the tokens within a particular sentence ranged widely from 6-100% for each of the models. In addition, 71%, 71%, and 80% of sentences were classified 100% correctly by the HMM, SP, and CRF models respectively.

Although these high-level results are strong, a better indicator for these models' performance is their classification accuracy on edited tokens only because these labels represent the editing processing and appear infrequently in the data. The CRF is revealed to be a significantly better classifier based on edited token accuracy, which reflects its more flexible feature set that

		Hidden Markov Model			**Structured Perceptron**			**Conditional Random Field**		
		Overall Accuracy: 90.8 %			Overall Accuracy: 91.9 %			Overall Accuracy: 94.6 %		
		Edited Token Accuracy: 37.5%			Edited Token Accuracy: 34.9%			Edited Token Accuracy: 65.5%		
	Support	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Original	18,767	0.93	0.98	0.95	0.92	0.99	0.96	0.96	0.98	0.97
Inserted	1,458	0.85	0.49	0.62	0.89	0.46	0.61	0.94	0.78	0.85
Deleted	943	0.34	0.20	0.25	0.64	0.17	0.27	0.61	0.46	0.52
Total	21,168	0.89	0.91	0.90	0.91	0.92	0.90	0.94	0.95	0.94

Table 1. Summary of Classification Results for Training On All Sentences (Edited token accuracy is defined as (number of true positives for inserted tokens + number of true positives for deleted tokens) / (total number of inserted tokens + total number of deleted tokens). Boundary errors are not taken into account when calculating precision and recall.)

		Hidden Markov Model			**Structured Perceptron**			**Conditional Random Field**		
		Predicted			Predicted			Predicted		
		Original	Inserted	Deleted	Original	Inserted	Deleted	Original	Inserted	Deleted
Actual	Original	18,312	106	349	18,621	70	76	18,447	58	262
	Inserted	720	711	27	768	674	16	299	1,142	17
	Deleted	730	23	190	761	17	165	491	21	431

Table 2. Confusion Matrices for Training On All Sentences

takes more sentence context into account when tagging. The CRF correctly identified 66% of edited tokens while the HMM and SP correctly identified 38% and 35% respectively.

Among edited tokens, inserted tokens were classified more precisely and with higher recall than deleted tokens. Across all three models, precision ranged from 85-94% for inserted tokens and from 34-64% for deleted tokens. Recall was above 46% for inserted tokens (reaching as high as 78% for the CRF) while recall was below 46% for deleted tokens (dropping as low as 17% for the HMM). As seen in the confusion matrices in Table 2, the weaker performance on deleted tokens is largely due to the models being more likely to misclassify deleted tokens for original tokens than inserted tokens for original tokens.

Classification improved by making adjustments for the unbalanced data set in training, particularly for the HMM and SP models. As expected, HMM and SP benefited from a higher frequency of edited tokens, which placed more weight on correctly classifying these tokens in training. This meaningfully increased recall at the cost of reduced precision and low overall accuracy given a large number of original tokens were tagged as edited. The best example of improvement is the SP where edited token accuracy increased by 15% to 49% for training on edited sentences only and by 30% to 65% for training on randomly undersampled bigrams (compared to training on all sentences).

Most notably for the other models, when trained with randomly undersampled bigrams, HMM's edited token accuracy improved by 14% to 52% and CRF's dropped 12% to 53%. The drop in classification accuracy can likely be explained by the CRF's reduced ability to decipher context when considering only two tokens at a time.

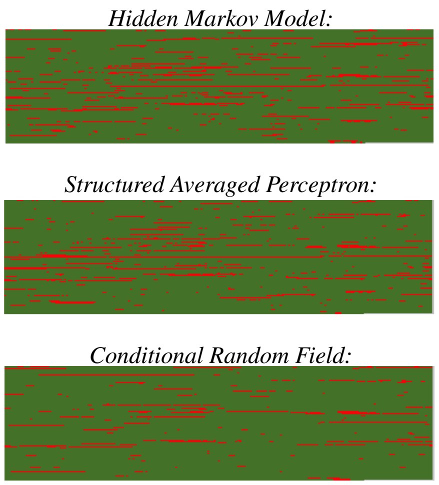
Because this is a preliminary look at applying machine learning to legal editing, these results require several caveats:

- The NDAs are sourced from a single firm and consequently the models identify that specific firm's preferred edits. Other firms may have different editing preferences, which these models may not be able to recognize or may find more difficult to learn in the training process, leading to poorer classification results than shown in this paper.
- Because there are only 105 documents, the text reflects a limited subset of the edits made, the language used, or situations addressed in an NDA. Additional data would be required to further generalize the three models and to further mitigate unbalanced data sets.
- Given that identifying edits is highly context dependent, treating sentences as observations may remove important context available earlier in the paragraph or earlier in the document. Models incorporating prior sequences into predictions (e.g. Recurrent Neural Networks) and longer training observations (if additional NDAs are available) should be evaluated.
- Text processing, feature selection, and model training could be further customized for legal documents, hopefully leading to improved classification results. This has already proven to be a successful strategy with the enhanced performance of models trained on edited sentences or randomly undersampled bigrams.
- The CRF can have a tendency to overfit the training data with complicated features that reflect nuances in one or two training examples. The feature set was deliberately restricted for the model in this paper, narrowing a potentially large gap between training and testing accuracy to 3% (98% training accuracy vs. 95% testing accuracy).

		Hidden Markov Model			Structured Perceptron			Conditional Random Field		
		Overall Accuracy: 88.8 %			Overall Accuracy: 90.6 %			Overall Accuracy: 91.3 %		
		Edited Token Accuracy: 39.4%			Edited Token Accuracy: 49.5%			Edited Token Accuracy: 69.1%		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Edited	Inserted	0.71	0.50	0.58	0.74	0.65	0.69	0.81	0.80	0.80
Sentences	Deleted	0.24	0.24	0.24	0.33	0.26	0.29	0.36	0.53	0.43
		Overall Accuracy: 83.2 %			Overall Accuracy: 34.7 %			Overall Accuracy: 27.1 %		
		Edited Token Accuracy: 51.8%			Edited Token Accuracy: 65.0%			Edited Token Accuracy: 53.4%		
RUS	Inserted	0.45	0.60	0.52	0.10	0.94	0.18	0.13	0.89	0.22
Bigrams	Deleted	0.28	0.45	0.35	0.14	0.41	0.20	0.14	0.24	0.18

Table 3. Summary of Classification Results After Adjustments for Unbalanced Data

Testing Example Tokens Tagged By Each Model



Example Paragraph Tagged By Each Model

Hidden Markov Model:
<del>Within ten days <\del> <ins>Promptly,<\ins>after being so requested by the Company or [NAME] <del >[NAME]<\del><ins>[NAME] in writing,<\ins> except to the extent you are advised by legal counsel that complying with such request would be prohibited by law or regulatory authority, you will return or destroy at your cost all Evaluation <del >Material at the option<\del><ins>Material.

Structured Averaged Perceptron:
<del>Within ten days<\del><ins>Promptly,<\ins>after being so requested by the Company or [NAME] <del >[NAME]<\del><ins>[NAME] in writing,<\ins>except to the extent you are advised by legal counsel that complying with such request would be prohibited by law or regulatory authority, you will return or destroy at your cost all Evaluation <del >Material at the option<\del><ins>Material.

Conditional Random Field:
<del>Within ten days<\del><ins>Promptly,<\ins>after being so requested by the Company or [NAME] <del >[NAME]<\del><ins>[NAME] in writing,<\ins>except to the extent you are advised by legal counsel that complying with such request would be prohibited by law or regulatory authority, you will return or destroy at your cost all Evaluation <del >Material at the option<\del><ins>Material.

Figure 2. Visualized Classification Results for Training On All Sentences (Green represents correct classifications while red represents incorrect classification. Names redacted for this paper are indicated with [NAME].)

Conclusion

Despite caveats, this paper's classification results should be treated as a positive sign that applying machine learning to editing legal documents is feasible and worthy of additional research. Approximately 90% overall accuracy and 70% edited token accuracy in the best models for this simplified classification task show that supervised learning models can handle the complexities of sparse, unbalanced, context-rich legal document data as long as the right text and training processes are in place.

Many additional steps need to be taken before machine learning lightens lawyers' workloads. The long-term ideal model that automates editing original NDAs will need to identify deleted tokens similar to the tagging done by the models in this paper. Yet, deleted tokens had the poorest classification performance and were at best identified correctly 60% of the time. Preliminary testing of models trained on edited documents to find words that should be deleted in original documents were unsuccessful. In addition, the long-term ideal model will have to generate text to insert into the original document. With more options on how many and what particular words are inserted, this will be the largest and most rewarding challenge.

References

1. Colgrove, C., Tibshirani, J. & Wong, R. Predicting edit locations on wikipedia using revision history. *Stanford Undergraduate Research Journal* 80–83 (2011).
2. Bird, S., Loper, E. & Klein, E. Natural language processing with python. (2009).
3. Finkel, J. R., Grenager, T. & Manning, C. Incorporating non-local information into information extraction systems by gibbs sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)* 363–370 (2005).