Leveraging Document Structure for Better Classification of Complex Legal Documents

Alex Ratner
Stanford University / 353 Serra Mall, Palo Alto, CA
ajratner@stanford.edu

Abstract

Document classification is a machine learning application that has been as impactful as it has been successful in a myriad of domains and applications. However, when the documents being classified are large and highly-complex, and when the set of potential classes is large as well, these models could be improved by incorporating more information about the documents' overall structure. Most approaches use bag-of-words type models that discard local structure and focus on types of words or n-grams used. In this paper, we examine several models and attempt to leverage both local (e.g. n-gram) and global (e.g. structure and organization) document features. We apply these approaches to a new dataset of legal documents.

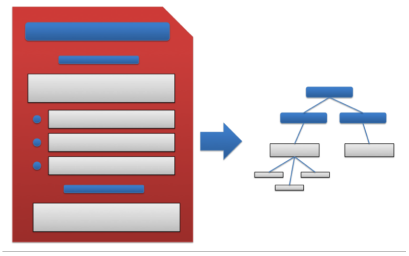
1 Introduction

Text classification is an important component of many modern applications such as information retrieval, information extraction and domain-specific content processing systems. To date, many text classification systems have achieved performance success using simple features that mostly or completely ignore word ordering, document structure and organization, and other such features, and then use sophisticated generative (e.g. LDA) (Blei et. al., 2003) or discriminative (e.g. SVM) models to classify documents. Many approaches preserve some local structure by looking at subsequences of words ("ngrams") or by incorporating some dependency or parse tree information. Lately, several "deep learning" models have attempted to preserve even more local structure by learning high-dimensional representations of large variable-length strings using recursive neural net architectures (Le and Mikolov, 2014).

We hypothesize that for some very large, complex document sets with nuanced classification schemes, even a lot of this local structure might be the same across different document classes, and some awareness of the overall structure and organization of the document might help with the classification task, as previous work leveraging document structure has shown (Chen et. al., 2008).

For this paper, we are specifically interested in classifying legal contract documents. In terms of general motivation, the automation of certain legal processes is a compelling challenge since it is widely acknowledged today that access to legal services and representation is majorly skewed towards those with enough money to afford better lawyers. In large part, this is due to the high costs of manually-performed tasks such as information retrieval, extraction, classification, and anomaly detection, that could be partially or fully automated.

In terms of technical motivation, contract documents are of interest because they are large, complex documents that nonetheless often have some shared structures such as titles, sections, subsections, etc. In our dataset, not only are there many different categories but they are often similar (ex: "Account Receivables Financing Agreement" vs. "Account Receivables Purchase Agreement"), use similar words and phrases, etc.



We explore some discriminative algorithms with some simple feature extraction and feature space reduction techniques, then move on to a custom generative model which attempts to more ex-

plicitly model the document structure we observe.

2 Dataset

We collected our dataset from OneCLE.com, an aggregator site that collects publicly-disclosed legal contract documents from the SEC and categorizes them manually. We began by crawling onecle.com[1] and scraping 27,979 contract documents in 326 categories ranging from "Arbitration Agreement" to "Manufacturing Contract".[2]. In most cases, we restrict our consideration to categories with > 50 documents, which results in a reduced dataset of 13,844 documents. Additionally, in most cases we further randomly subsample to 10,000 documents for slightly more balanced classes. See Results and Discussion.

3 Features and Preprocessing

In order to capture some of the document organizational structure that we wished to leverage, we used a small set of heuristic rules to extract individual sections and their titles, and the overall document title. As will be discussed further, the dataset was extremely noisy with regard to structure- at least 7 different word document-to-html processes appeared to have been used- so we limited our parsing of structure to these high-level components.

Once the documents were parsed into high-level structural components, we preprocessed further using minimum word-length thresholding (min=3), minimum and maximum corpus frequency thresholding (we kept words w that appeared in more than 3 but less that $0.8 * |D|$ of the documents D), Porter stemming (a form of suffix-removal that results in normalized forms of words), and removal of certain non-content words ('stop-word' removal).

For our discriminative algorithms (Logistic regression and SVM), we then transformed each component into a vector of TF-IDF weighted indicator features; in other words, given a training set vocabulary $V = \{w_1, ..., w_{|V|}\}$ from N documents, we represented each text component as a length-$|V|$ vector $\vec{x}^{(d)}$ where $x_i^{(d)} = \alpha_{i,d} * f(i,d)$, with $f(i,d)$ being the count of word w_i in document d, and $\alpha_{i,d}$ being the TF-IDF weight,

$$\alpha_{i,d} = \left(0.5 + \frac{0.5 * f(i,d)}{max(\{f(j,d) : w_j \in d\})}\right) * \log\left(\frac{N}{|\{d \in D : w_i \in d\}|}\right)$$

Even with these preprocessing steps, we still ended up with $|V| \approx 40,000$ [3], meaning our feature vectors were this length as well. We tried three types of feature dimensionality reduction: χ^2 test thresholding (select the K best features according to this statistic of informativeness); principle component analysis (PCA); and latent semantic analysis (LSA; similar to PCA, this is the term for truncated SVD used on feature vectors such as ours). Detailed review of these methods is not included as they uniformly appeared to dramatically lower classifier performance in testing, and were thus not utilized.

Finally, we handled structure in the following model-specific ways: for Multinomial NB, no information about document structure was kept; for logistic regression and SVM, we used separate vocabularies and feature vectors for the body text and titles respectively, and then concatenated these vectors; for *Cross-section* Multinomial NB, we represented structure as described in the following section.

4 Models

4.1 Logistic Regression

Our baseline discriminative model was L2-norm logistic regression, which minimizes the following cost function:

$$J(w,c) = \frac{1}{2}w^T w + C\sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$$

where C is a hyper parameter determining the balance between fitting the model and penalizing overfitting with the L2-norm.

4.2 SVM

As reviewed in class, an SVM minimizes the following cost function:

$$J(w, b\gamma) = \frac{1}{2}w^T w + C\sum_{i=1}^{n} \gamma_i$$

[1] Permissible according to their robots.txt file

[2] Permissible as these are unmodified versions of the public-domain documents provided by the SEC

[3] Approximate due to the random sub-sampling done to balance class membership sizes, noted in the previous section

subject to:

$$y_i(w^T x_i + b) \geq 1 - \gamma_i$$
$$\gamma_i \geq 0 \forall i$$

In the dual form of the problem, we can write the optimization problem in terms of $< x, x' >$ and then replace these with arbitrary (Mercer) kernel functions $\phi(x, x')$. We use two different kernel functions, a linear one- $\phi(x, x') =< x, x' >$- and a radial basis function (RBF) kernel- $\phi(x, x') = \exp(|x - x'|^2)$.

4.3 Multinomial Naive Bayes

Multinomial Naive Bayes is a generative model which uses the "Naive Bayes" approximation to assume independence between every pair of features. Thus we can model the probability of a document is $p(y) \prod_{i=1}^{n} p(x_i|y)$, where $x_1, ..., x_n$ are the words in the document of class y, and where our model is parameterized by $\theta_{i|y} = p(x_i|y)$ and $\phi_y = p(y)$, and where we use Laplace smoothing to redistribute some probability mass from observed word statistics to allow for words in the test set that were not seen in the training set.

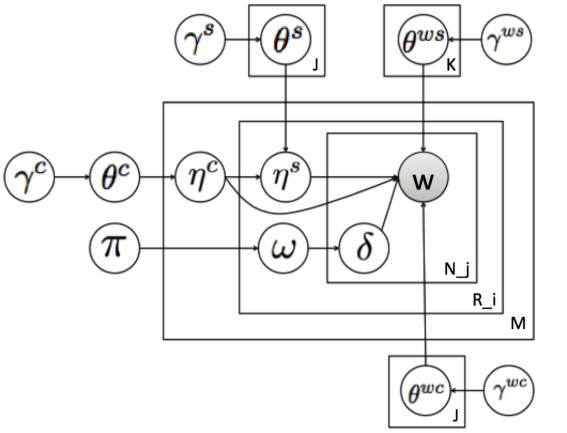
We implement Multinomial NB in two ways- first using MLE to find the optimal parameters, and then using Gibbs sampling (see next subsection), mostly in order to set a relative baseline for the other Gibbs sampling model used.

4.4 *Cross-section* Multinomial Naive Bayes

Our final model is a potentially novel[4] attempt to more explicitly account for document structure, which we term the "Cross-section" version of Multinomial NB. Specifically, we hypothesize that in each section of a contract, there is some language that is very specific to that contract and its *contract class* as a whole, and some language that is more specific to a certain type of *section class* used across multiple types of contracts. For example, multiple types of contracts might have a section having to do with limiting liabilities; we hypothesize that this section might have some words strongly related to the contract type, and some words related more to the general concept of liabilities. Moreover, we hypothesize that the sections of the contract might be a good set of segmentations to reflect these factors.

[4]In a small, application-specific way... we know there are lots of plate models out there!

We use a plate diagram to illustrate the model proposed:



This model defines the following generative process for creating a contract, given J contract classes and K section classes to choose from:

- A contract class is sampled: $\eta^c \sim Multinomial(\theta^c)$
- For each section:
  - A section class is sampled: $\eta^s \sim Multinomial(\theta^s)$
  - A Bernoulli parameter is sampled: $\omega \sim Beta(\pi)$
  - For each word:
    - A binary value is sampled: $\delta \sim Bernoulli(\omega)$
    - If $\delta = 0$, a word is sampled conditional on the contract class, $w \sim Multinomial(\theta^w c)$
    - If $\delta = 1$, a word is sampled conditional on the section class, $w \sim Multinomial(\theta^w s)$

The Multinomial parameters θ all have corresponding Dirichlet priors, however for simplicity we use symmetric and uniform $(= 1)$ priors so this essentially works out to Laplace smoothing.

We then use Gibbs sampling, which roughly is the technique of sampling all the involved parameters and labels one at a time, conditioned on all the other parameters/values as set in the previous sampling iteration. We calculate the conditional distributions required for sampling; for example, for sampling a new document class for document i, the probability of a certain contract class c con-

ditioned on the other parameters is:

$$P(c|\vec{C}^{(-i)}, \vec{S}, \theta^c, \theta^s, \vec{\omega}, \theta^{wc}, \theta^{ws}, ...)$$
$$= \frac{P(c) * P(\vec{C}^{(-i)}, \vec{S}, \theta^c, \theta^s, \vec{\omega}, \theta^{wc}, \theta^{ws}, ...|c)}{P(\vec{C}^{(-i)}, \vec{S}, \theta^c, \theta^s, \vec{\omega}, \theta^{wc}, \theta^{ws}, ...)}$$

Since we are calculating these probabilities for the classes c in order to sample from a Multinomial, we can disregard the denominator which has no dependence on c:

$$P(c|\vec{C}^{(-i)}, \vec{S}, \theta^c, \theta^s, \vec{\omega}, \theta^{wc}, \theta^{ws}, ...)$$
$$\propto P(c) * P(\vec{C}^{(-i)}, \vec{S}, \theta^c, \theta^s, \vec{\omega}, \theta^{wc}, \theta^{ws}, ...|c)$$
$$\propto \left(\frac{count(c)+1}{N_{docs}+n_{classes}}\right) \prod_{j=1}^{n_{sections,i}} P(s_j|c)$$
$$* \prod_{i=1}^{n_{words,i,j}} [\delta_{ij} P(w_{ij}|c) + (1-\delta_{ij}) P(w_{ij}|s)]$$
$$= \left(\frac{count(c)+1}{N_{docs}+n_{classes}}\right) \prod_{j=1}^{n_{sections,i}} \theta^s_{s_j|c}$$
$$* \prod_{i=1}^{n_{words,i,j}} \left[\delta_{ij} \theta^{wc}_{w_ij|c} + (1-\delta_{ij}) \theta^{ws}_{w_ij|s}\right]$$

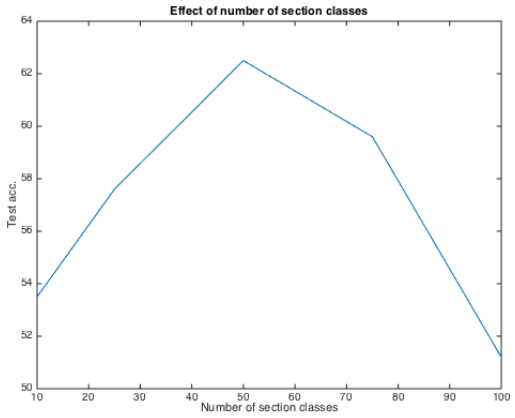
And similarly for the other conditional distributions needed for the sampling iterations.

5 Results

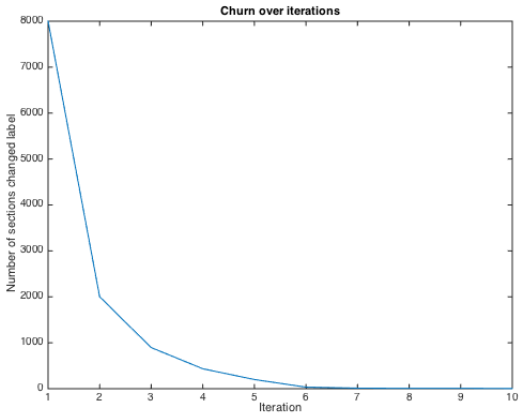
We use stratified 5-fold cross validation[5] on a randomly-subsampled down set of 10,000 documents in order to balance class membership counts, selecting from 77 classes which had at least 50 documents.

Model	**Test Acc.**	Train Acc.
Logistic Regression	**82.5%**	95.9%
SVM (linear kernel)	**82.0%**	98.5%
SVM (RBF kernel)	**25.0%**	27.8%
Multinomial NB (MLE)	**70.4%**	90.3%
Multinomial NB (Gibbs)	**60.5%**	N/A
Cross-section NB	**62.5%**	N/A

For our Gibbs sampling models, we used a custom implementation in python. We used burn-in and thinning to reduce spurious effects from the initial state space exploration and to avoid autocorrelation respectively. Nevertheless, our implementation was somewhat limited in terms of computational resources, which limited our ability to explore the hyper parameter space. We did examine the optimum number of section classes to set, plotted below[6].



We also observed that the system seemed to converge to a steady state local optima very quickly, perhaps much faster than we would want for ideal performance; an illustrative trial is plotted below:



6 Discussion

In this paper we attempted to (a) explore the range of practical options for dealing with a classification task of interest, and (b) investigate some ways to leverage document structure for better classification. Specifically, we attempted to learn classification models for long, complex documents, with a high number (77+) of total classes.

One initial finding of interest was that simple attempts at feature dimensionality reduction had terrible effects on performance; perhaps because a large volume of specific words were actually very relevant to contract class distinctions. Addition-

[5] Stratified meaning that we keep the proportions of classes constant in each fold

[6] Although caution should be used when viewing this plot due to the high variance in our method

ally, we note that the SVM with RBF kernel performed extremely badly; this is especially interesting compared to the very high performance of the SVM with linear kernel. We predict that performance could be improved with more careful calibration of the RBF hyper-parameters, however this disparity is still interesting for what it might apply about RBF kernel performance in a very high dimensional, sparse feature space in a classification problem with a large number of classes.

With respect to the generative models explored, we see that not only did generative MLE approaches tend to do worse than the discriminative algorithms, but that Gibbs sampling methods did significantly worse still. It is widely known that there is some 'black magic' in the implementation of Gibbs sampling methods, so perhaps- due in part to our computation-limited implementation- we simple did not explore the configuration / hyper-parameter space thoroughly enough (for example, we only used symmetric uniform Dirichlet priors). Another factor was the fast rate of convergence to steady-state observed; we hypothesize that if we could get the sampler to explore the state space more then performance would be much higher. Multiple random initialization runs might be a tack for this, given a faster sampler implementation. In general, we observed that the sampler was highly sensitive to initialization methodology as well.

Finally, though we observe that our *Cross-section* model performed slightly better than basic Multinomial NB, we do not know if this is a statistically significant difference given the variance in our method. Additionally, there are several ways that our *Cross-section* model could effectively reduce to the basic Multinomial NB case, for example if the Bernoulli priors become very low, which we would still need to investigate more carefully.

However, we do think the relatively decent performance of the newly-proposed model indicates that similar methods might warrant further exploration, for several reasons. First of all, the documents collected were *extremely* messy, and extraction of structure was far more difficult than anticipated. With a better collection of documents, structure could be both more accurately represented and more deeply leveraged. Second of all, the parameters of this model were pushing the limits of the simple implementation used; with a better sampler, we could much more effectively explore this model and more complex ones.

7 Conclusion

In this paper we explore several methods for automatically classifying complex legal documents of a large number of classes. We compare discriminative and generative approaches, including a novel generative model for capturing cross-correlations in the document substructure. We find however that simple discriminative models- such as an SVM with linear kernel and logistic regression- still attain the best performance; however we think further exploration is warranted with some of the generative approaches which explicitly model document structure.

8 Future Work

In the future, we would like to pursue three major directions: (1) Explore generative models further like the ones proposed which explicitly model document structure, however using better quality source documents and a faster Gibbs sampler; (2) Explore deep learning approaches, using word embeddings and recursive auto encoders mapped onto document structure; and (3), explore the failure of SVM with RBF kernel and dimensionality reduction methods that we observed.

References

Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 2 (2011) 2825-2830.

Philip Resnick, Eric Hardisty. 2010. Gibbs Sampling for the Uninitiated. *Technical Report*, http://www.umiacs.umd.edu/ resnik/pubs/LAMP-TR-153.pdf Accesed 12/12/2014.

Pengtao Xie, Eric Xing. 2013. Integrating Document Modeling and Text Clustering. *CUAI*, 2013.

David Blei, Andrew Ng and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (2003) 993-1022.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *Proceedings of the 31st International Conference on Machine Learning*, 2014. JMLR: W&CP volume 32.

Harr Chen, S.R.K. Branavan, Regina Barzilay and David R. Karger. 2009. Content Modeling Using Latent Permutations. *Journal of Artificial Intelligence Research*, 2009c.