Automatic Hilghter of Lengthy Legal Documents

Yanshu Hong
Computer Science Department
Email: yanshuh@stanford.edu

Tian Zhao
Electrical Engineering Department
Email: tianzhao@stanford.edu

1. Introduction

Legal documents are known for being lengthy. To our knowledge, some categories of legal documents contain duplicated information that do not require our attention. However, manually extracting non-duplicate information from documents requires considerable amount of efforts. Thus, we want to use machine learning algorithms to pick up unordinary sentences for us. In this paper, we propose a set of algorithms that filters out duplicate information and returns useful information to the user. We are able to train a learner that can mark unordinary parts of a legal document for manual scrutinization.

First, we pick some legal documents that contain common patterns, e.g. documents about software user agreements, to form a knowledge base for our trainer. We then use *LDA* [1] model to generate a set of common topics across the knowledge base.

At the learning phase, our input is a new piece of legal document in the format of plain text. We first remove common topic words from our test document to increase the differences between sentences in the test document. We then use *Word2Vec* [2], [3] to convert sentences into vectors. We then use *Agglomerative Clustering* and *LOF* [4] algorithms to detect special sentences in the given document. Last, we use *PCA* and *t-SNE* to visualize our result.

2. Related Work

Though text-mining related works are often seen, few of them address the issue of sentence-level anormaly detection. However, we are able to draw inspirations from previous works on unsupervised auto-generation of document summary.

Qazvinian and Radev (2008) presented two sentences selection models based on clusters generated by hierarchical clustering algorithms. *C-RR* finds the largest cluster and rank the importance of sentences by their order of appearance in the cluster. *C-lexrank* selects the most centered sentences within the cluster to be the salient ones. While their clustering strategy performs well on segmenting an article, their strategy on finding the salient sentences does not apply to our case. Based on their design, a sentence with the most common pattern would be chosen as a salient one. However, it is also the case that such sentence would contain the most duplicate information; therefore it should not require much attention from the reader.

Similarly, Nanba and Okumura (1999) discussed ways of citation categorization, which essentially cluster sentences and extract ones closer to the center.

Based on the work of Qazvinian and Radev, Cai (2010) presented a salient sentence extraction strategy by adding a ranking policy for sentences within each cluster. However, Cai did not discuss on how to select the initial condition of centroids. It seems to be the case that the centroids are randomly selected initially, which is not desired because the final clustering would vary a lot as the intial condition changes.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003) presented the *LDA* model to extract common topic words from a set of documents. We find this model very robust, and in fact *LDA* is used widely accross contextual anormaly detection. For example, Mahapatra and Amogh presented a text clustering strategy based on *LDA*. Their model works well; however their feature extraction part can be improved by using more robust algorithms to translate the relation between words into vectors. Having this in mind, we use *Word2Vec* as our major text feature extraction framework.

3. Dataset and Features

We generate our samples by parsing html pages of legal documents. Here is a sample text from iTunes user agreement:

THE LEGAL AGREEMENTS SET OUT BELOW GOVERN YOUR USE OF THE ITUNES STORE, MAC APP STORE, APP STORE, APP STORE FOR APPLE TV, IBOOKS STORE AND APPLE MUSIC SERVICES ("SERVICES"). TO AGREE TO THESE TERMS, CLICK "AGREE." IF YOU DO NOT AGREE TO THESE TERMS, DO NOT CLICK "AGREE," AND DO NOT USE THE SERVICES.

A. TERMS OF SALE

PAYMENTS, TAXES, AND REFUND POLICY

You agree that you will pay for all products you purchase through the Services, and that Apple may charge your payment method for any products purchased and for any additional amounts (including any taxes and late fees, as applicable) that may be accrued by or in connection with your Account. YOU ARE RESPONSIBLE FOR THE TIMELY PAYMENT OF ALL FEES AND FOR PROVIDING APPLE WITH A VALID PAYMENT METHOD FOR PAYMENT OF ALL FEES. For details of how purchases are billed please visit http://support.apple.com/kb/HT5582.

Your total price will include the price of the product plus any applicable tax; such tax is based on the bill-to address and the tax rate in effect at the time you download the product.

All sales and rentals of products are final.

Prices for products offered via the Services may change at any time, and the Services do not provide price protection or refunds in the event of a price reduction or promotional offering.

Figure 1. A sample text

Our training dataset contains 35 legal documents, which include 124,546 words. We use two documents for testing. One is the iTunes user agreement, which contains 1,329 words. We manually label some special sentences in this document. We then run our pipeline on this document, and use the number of labeled sentences highlighted by our pipeline as a measure of how successful the pipeline is.

The other document is the Atlassian's user agreement, which contains 7,934 words. Due to the size of this document, we cannot label all the sentences. Instead, we use this document for generalization test. We run our pipeline on Atalassian's user agreement, read the highlighted sentences, and subjectively check if these sentences are unordinary. Since the problem we defined is unsupervised, we can only evaluate the final results by subjective means. To increase the credibility of our evaluation, we also use quantitative methods for evaluating the robustness of our pipeline. We will discuss more about these methods in section 5.

We remove prepositions, numbers, punctuations because they add noise to our samples. For example, "I have iphone" and "I have an iphone" are treated as the same sentence in our pipeline.

We also realize that some words with different suffixes may have the same meaning. For example, "mature" and "maturity" should be understood as the same word. In preprocessing step, we run an algorithm called *Porter Stemming* [11] to convert words with the same stem and similar suffixes into one word.

We have two different sets of feature extraction and selection strategies. First, we use *LDA* to construct common topic words across the training dataset. The feature extraction within *LDA* is done by taking all documents as a single corpus and then by applying the variational inference method discussed in the original paper.

After running *LDA* on all the documents, we get a list of topic words. A topic is a list of words, e.g. *trademark, services, may, use, application, agreement, content*. We observe that these topic words do not contribute to the "specialty " of a sentence. Therefore, we remove the topic words from the test document.

Second, for the test document, we use *Word2Vec* to translate a word into a vector. *Word2Vec* is a two-layer neural net that processes text. It takes text corpus as input and its output is a set of feature vectors for words in that corpus. For each word, we generate 100 features. These word vectors allow for numeric operations on words. For example, the vector of "king" added by the vector of "woman" would return the vector of "queen". After running *Word2Vec*, we establish a mapping from words to vectors.

```
>>> model['apple']
array([  3.39468718e-01,  -9.81548280e-02,   2.06640467e-01,
        -2.21591026e-01,  -4.72269282e-02,   1.05696611e-01,
```

Figure 2. Part of word vector of "apple"

We assume that the feature vector of a sentence is the sum of feature vectors of all the words in this sentence. We then run clustering algorithms on the vector representations of all the sentences.

4. Methods

We use *EM* algorithm to learn the hidden parameters of the *LDA* model. We then use *Agglomerative clustering* to learn the distribution of sentences. We also use *LOF* to learn the distributions of unordinary sentences in the document. We will discuss these methods in the following subsections.

4.1. *EM* for estimating *LDA* parameters

The *LDA* model uses a word w as a basic unit. A document is a sequence of N words denoted by $\mathbf{w} = (w_1, \cdots, w_N)$. A corpus contains M documents is denoted by $\mathbf{D} = \{\mathbf{w}_1, \cdots, \mathbf{w}_M\}$.

LDA takes the following generative steps for each document $\mathbf{w}$ in a corpus $\mathbf{D}$:

(a) Choose $N \sim$ Poisson(ξ).
(b) Choose $\theta \sim$ Dir(α).
(c) For each of the N words $\mathbf{w}_n$:
  i Choose a topic $z_n \sim$ Multinomial(θ).
  ii Choose a word w_n from $p(w_n|z_n, \beta)$, where p is a multinomial probability conditioned on z_n and β.

The probability of seeing an N word document, $P(w|\alpha, \beta)$, is given as:

$P(w|\alpha,\beta) = \int(\Pi_{n=1}^{N} P(w_n|\theta,\beta))P(\theta|\alpha)d\theta$.

To estimate the two hidden parameters, (α,β), from a corpus **D**, one needs to maximize the log likelihood:
$\log P(\mathbf{D}|\alpha,\beta) = \sum_{d=1}^{M} \log P(w^{(d)}|\alpha,\beta)$,
which is not tractable. To resolve this issue, the LDA paper recommends to use variational inference method. The key of this method is to apply the Jensen's inequality to obtain a lower bound of $\log P(w|\alpha,\beta)$. Let $q(z,\theta)$ be the joint pdf of z,θ. Then:
$\log P(w|\alpha,\beta) \geq \int \sum_z q(z,\theta) \log \frac{P(w,z,\theta|\alpha,\beta)}{z,\theta} d\theta$.
Let z,θ be parameterized by ϕ,γ respectively. Let $L(\phi,\gamma;\alpha,\beta)$ denotes the lower-bound of $\log P(w|\alpha,\beta)$. Then:
$\log P(\mathbf{D}|\alpha,\beta) \geq \sum_{d=1}^{M} L(w^{(d)};\phi^{(1,d),\cdots,\phi^{N_d,d}},\gamma^{(d)};\alpha,\beta)$
$= \mathbf{L}\ (\Phi,\Gamma;\alpha,\beta)$,
where $\Phi = [\phi^{(n,d)}], \Gamma = [\gamma^{(d)}]$.

The idea of EM is to start from initial α,β, and iteratively improve the estimate:
E-step: Given α,β,
$(\Phi,\Gamma) := \arg\max_{(\Phi,\Gamma)} \mathbf{L}(\Phi,\Gamma;\alpha,\beta)$;
M-step: Given Φ,Γ,
$(\alpha,\beta) := \arg\max_{(\alpha,\beta)} \mathbf{L}(\Phi,\Gamma;\alpha,\beta)$.
These two steps are repeated until **L** converges. We won't show the complete derivation here, but we use the following updating equations for getting (α,β).
In *E-step*, the update equations for Φ and Γ are:
$\phi_i^{(n,d)} :\propto \beta_{i,w_n} \exp(\digamma(\gamma_i^{(d)} - \digamma(\sum_j \gamma_j^{(d)}))$
where $\digamma$ is the digamma function;
$\gamma_i^{(d)} := \alpha_i + \sum_{n=1}^{N_d} \phi_i^{(n,d)}$.
In *M-step*, the update equation for β is:
$\beta_{ij} :\propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_i^{(n,d)} \mathbf{1}(w_n^d = j)$.
There is no analytical solution to α, but it can be obtained by maximizing:
$L_\alpha = \sum_d (\log \bar{\Gamma}(\sum_{i=1}^{K} \alpha_i) - \sum_{i=1}^{K} \log \bar{\Gamma}(\alpha_i)$
$+ \sum_{i=1}^{K} (\alpha_i - 1)(\digamma(\gamma_i^{(d)}) - \digamma(\sum_{j=1}^{K} \gamma_j^{(d)})))$
by using Newton's method.

4.2. Agglomerative Clustering

For a new test document, we first get the feature vector of each sentence. We then use *Agglomerative clustering* to segment the document. We first specify that we want to have N clusters. We then put each sentence into its own cluster, and iteratively merge the closest clusters until only N clusters remain. We then look at these clusters. If there exists a cluster that contains only one sentence, then we know that this sentence is a special one, as it is never merged into other clusters.

We define the cosine similarity between two sentences as the dot product of their normalized vector representations with zero mean. We then calculate cosine distance from cosine similarity and use this distance as our metric.

The distance between any two clusters A and B is taken to be the mean distance between elements of each cluster [12].

In section 5, we will discuss on how to choose the right number of clusters to achieve the best clustering structure.

It is worth mentioning that we find *K-means* and *K-means++* performing much more unstable than *Agglomerative Clustering* in our case. The final outcome of *K-means* and *K-means++* depends heavily on the initial settings of the centroids. However, since both methods choose initial centroids in a semi-random fashion, it is not guaranteed that the final clustering would remain the same from run to run. In contrast, *Agglomerative Clustering* does not depend on initial assignments of centroids. As a result, it creates more stable outcomes.

Before running *LOF*, we recommend to run *Agglomerative Clustering* first, and then remove the anormaly sentences discovered by the clustering method. The intuition is that we want to decrease the local density around unordinary sentences so that these sentences can be more easily picked up by *LOF*.

4.3. Local Outlier Factor

In our feature space, each sentence is represented as a point. Therefore, if we find a point with low local density, then this point is an outlier, and its corresponding sentence is unordinary.

To help explain how *LOF* works, we introduce two terms:
$\mathrm{N}_k(A)$: Set of k nearest neighbors to A.
k-D(A): distance from A to its k^{th} nearest neighbor.
$\mathrm{RD}_k(A,B) = \max\{\text{k-D}(A,B), \text{distance}(A,B)\}$,
where RD denotes reachability distance. We use the maximum of two distances to get a stable result.
The local reachability density of A is computed as:
$\mathrm{lrd}(A) = 1/(\frac{\sum_{B\in \mathrm{N}_k(A)} \mathrm{RD}_k(A,B)}{|\mathrm{N}_k(A)|})$.
It is then compared with its neighbors using:
$LOF_k(A) = \frac{\sum_{B\in \mathrm{N}_k(A)} \mathrm{lrd}(B)}{|\mathrm{N}_k(A)|} / \mathrm{lrd}(A)$,
which is the average local density of neighbors divided by the object's own local density. If this value is greater than 1, it tells that the local density around this object is lower than its neighbors, and thus is an outlier.

This method studies the local distribution of points, and helps us improve on top of *Agglomerative Clustering*. Basically, it is hard to have insight of how points are distributed within an agglomerative cluster. Using

LOF helps us understand the distribution of points within a cluster.

5. Experiments and Results

In this section, we first discuss how we choose the hyperparameters for *LDA* and for *Agglomerative Clustering*. We will then talk about the performance our pipeline achieves on the test cases.

5.1. Tuning Hyperparameters

In this subsection, we will use the iTunes user agreement as an example to show how our hyperparameter choosing strategy works.

5.1.1. Choosing parameter for *LDA* model. Documents in our training dataset is unlabeled. Therefore, we wish to achieve high likelihood on a held-out testset. In fact, we use the perplexity of a held-out test set to evaluate the models.

Intuitively, a lower perplexity score indicates better generalization performance. The perplexity is equivalent to the inverse of the geometric mean per-word likelihood:

$perplexity(D_{\text{test}}) = \exp(-\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d})$,

using the same notation as in section 4.1.

In our experiment, we use the corpus generated by parsing html pages of legal documents from Atlassian, Github, and Apple. We held out 25% of the data and trained on the remaining 75%.

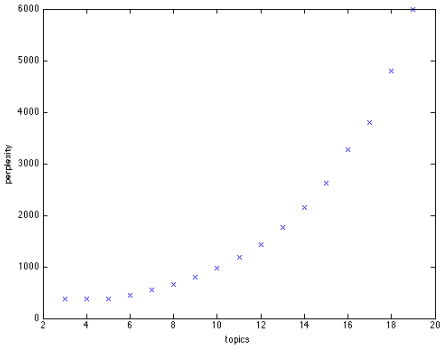

Figure 3. perplexity given number of topics

We found that the perplexity is the lowest when there are around $3 \sim 5$ topics. We found that for the iTunes user agreement, using 5 topics works the best for improving the differences between sentences.

5.1.2. Choosing number of clusters. We use Silhouettes [13] score to evaluate the robustness of clusters. Intuitively, larger Silhouettes score indicates that the distance between each cluster is large, and the maximum distance from the farthest point in a cluster to its centroid is small. The formal definition is given as follows:

For each point i, let $a(i)$ be the average dissimilarity of i with all other points in the same cluster. Let $b(i)$ be the lowest average dissimilarity of i to any other cluster, of which i does not belong to. The cluster with the lowest average dissimilarity is defined to be the neighboring cluster of i. This cluster is also the next best fit for i. Silhouette score is defined as:

$s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$, where $-1 \leq s(i) \leq 1$.

If $s(i)$ is close to 1, it means that the average dissimilarity between i and its neighboring cluster is much greater than the dissimialrity between i and its own cluster. In this case, we say that i is properly clustered. In contrast, if $s(i)$ is close to -1, it means that i would be more properly labeled by its neighboring cluster.

We evaluate the average of $s(i)$ over all sentence vectors of the test document. Since the average $s(i)$ over all points in a cluster is a measure of how tight the cluster is, the average $s(i)$ of the whole dataset is a measure of how properly the whole document has been clustered.

Acknowledgments

The authors would like to thank...

References

[1] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." the Journal of machine Learning research 3 (2003): 993-1022.

[2] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[3] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[4] Breunig, Markus M., et al. "LOF: identifying density-based local outliers." ACM sigmod record. Vol. 29. No. 2. ACM, 2000.

[5] Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of Machine Learning Research 9.2579-2605 (2008): 85.

[6] Radim Rehurek, *Optimizing word2vec in gensim*. Nov 07, 2015

[7] Qazvinian, Vahed, and Dragomir R. Radev. "Scientific paper summarization using citation summary networks." Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. Association for Computational Linguistics, 2008.

[8] Cai, Xiaoyan, et al. "Simultaneous ranking and clustering of sentences: a reinforcement approach to multi-document summarization." Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics, 2010.

[9] Nanba, Hidetsugu, and Manabu Okumura. "Towards multi-paper summarization using reference information." IJCAI. Vol. 99. 1999.

[10] Mahapatra, Amogh, Nisheeth Srivastava, and Jaideep Srivastava. "Contextual anomaly detection in text data." Algorithms 5.4 (2012): 469-489.

[11] Porter, Martin. "The Porter stemming algorithm, 2005." See http://www.tartarus.org/∼martin/PorterStemmer.

[12] Sokal, Robert R. "A statistical method for evaluating systematic relationships." Univ Kans Sci Bull 38 (1958): 1409-1438.

[13] Rousseeuw, Peter J. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." Journal of computational and applied mathematics 20 (1987): 53-65.