ThinkCite

Final Project: Master of Information Management & Systems, UC Berkeley School of Information

Michael Berger
(working with Talia Shwartz, School of Law and Juan Hernandez, DLab)
Advisor: Prof. Deirdre Mulligan
May 8, 2015

Table of Contents

Michael Berger

Final Project, Master of Information Management & Systems
Advisor: Prof. Deirdre Mulligan
May 8, 2015

ThinkCite

1. The Problem

Since the advent of lower cost digitization technologies, a growing number of legal materials are now available in electronic format. Those conducting legal research - practitioners, students, and scholars - are faced with an ever-expanding array of case law, statutes, and other documentary sources of law when searching for legal information. This information is, in many cases, organized and stored in commercial proprietary databases such as those operated by LexisNexis® and Westlaw®. These commercial services employ experts with domain-specific knowledge in order to organize and curate legal documents by hand, so as to better enable later information retrieval by legal researchers. For example, topically-related cases are grouped together into legal categories or topics. This human-enabled curating process can be expensive, as it inefficiently relies on workers with an expensive legal education.

From the perspective of the users - legal researchers in this instance - the act of research and writing can also be fraught with difficulty and expense. When writing a legal document, such as a brief of law or an internal memorandum, the user may need to find a decision, statute, or other document (such as an affidavit) to support the idea they wish to express. In this case, they must interrupt their workflow by switching from the word processor to a search tool, usually on a browser. Once in the search tool, the user must manually enter a query that they have constructed based on what it is they are looking for, and in many cases a natural language query is not supported or does not return useful results; rather, a complicated Boolean-like expression is required. The user may sometimes need to spend time searching through scores of potentially relevant documents before the desired document is found. Finally, when a case is located that the user wishes to cite or quote, the user must again break their workflow to manually copy and paste the cite or quote into the word processor.

2. Objective

Our objective was to build a working prototype of a software system that would attempt to ameliorate these problems. Following a classic software engineering design pattern, we decided to bifurcate the system into two components: a "recommendation engine" and a "graphical user interface (GUI)."

The objective of the recommendation engine would be to address the information organization and retrieval problems presented above. We set out to use machine learning and natural language processing techniques to organize a corpus of legal documents and perform natural language queries against the corpus of documents. With each query entered into the system, the engine would recommend documents that were relevant to the query. The machine learning algorithm would perform the information organization, categorizing documents based on a set of "topics" learned from the entire corpus, obviating the need for human organization. Additionally, the algorithm that we aimed to apply - Latent Dirichlet Allocation - accounts for documents that fall under multiple topics, which we hypothesized would be more naturally suited to legal documents. Finally, we set out to collaborate on a paper of publishable quality that would report on the recommendation engine. We set a goal of completing and submitting the paper for consideration to the 15th International Conference on AI and Law (ICAIL 2015), Workshop on Law and Big Data, with a submission deadline of May 1, 2015.

The objective of the GUI would be to address some of the user experience issues identified with the legal research process and described in the section above. To do so, we set out to build a plugin that would run within Microsoft Word (the most popular word processing tool, used by the vast majority of legal researchers). The plugin would be activated from within a Word writing session; once activated, it would send the writing context (i.e., the last paragraph before the position of the cursor, or a block of text selected by the user) as a natural language query to the recommendation engine. The recommendation engine would then return a set of recommended documents to the GUI for display to the user.

The user would then be able to select which documents were to be cited, and would be assisted in this choice through the provision of the most relevant paragraphs from the documents. If the user wished, they would be able to seamlessly read the entire text of the recommended document, in order to ensure that the document was indeed appropriate for the task. They would be able to select portions of the text in the document and choose to output those portions as quotes. Once the user had made the selection of citations and quotations, the final step would be to output those citations and quotations directly into the

document, without the user needing to copy and paste, or enter the citations manually. The objective of the entire process was to simplify and "black box" the process of searching for, selecting, and citing relevant case material in the course of a legal writing session.

The Team

1. Team Members

There were three members of the team. Talia Shwartz was an LLM student at UC Berkeley, Boalt Hall School of Law. She previously received her LLB and another LLM, and thus brought valuable academic legal research experience and domain knowledge to the team. Juan Hernandez, an MSc student at Northwestern University, McCormick School of Engineering, and an affiliate at the UC Berkeley DLab, brought exceptional data science, machine learning, and computer programming skills. Finally, Michael Berger, a MIMS student at the UC Berkeley School of Information, bridged these two skillsets together, having previously studied and practiced law (JD) and computer science (BSc).

2. Roles

Some tasks were divided into individual action items, while others were shared and performed collaboratively.

Talia took on the task of researching and writing the literature review and abstract for the recommendation engine and related paper. She also designed the digital mockups for the GUI. Juan built and tested the recommendation engine. Michael developed and tested the GUI and associated Microsoft Word plugin.

All three team members worked collaboratively on an ongoing basis to complete all other required tasks. From February to May, we met weekly to plan, work together on the systems, update on progress and results, and assign concrete action items to be accomplished by each team member in the interim times between meetings. During these meetings, we also collaborated on the paper prototyping exercise for the GUI, and together wrote the research paper that was submitted to, and accepted by, the ICAIL 2015 conference. We also tested early iterations of the GUI together for usability.

The Graphical User Interface (GUI)

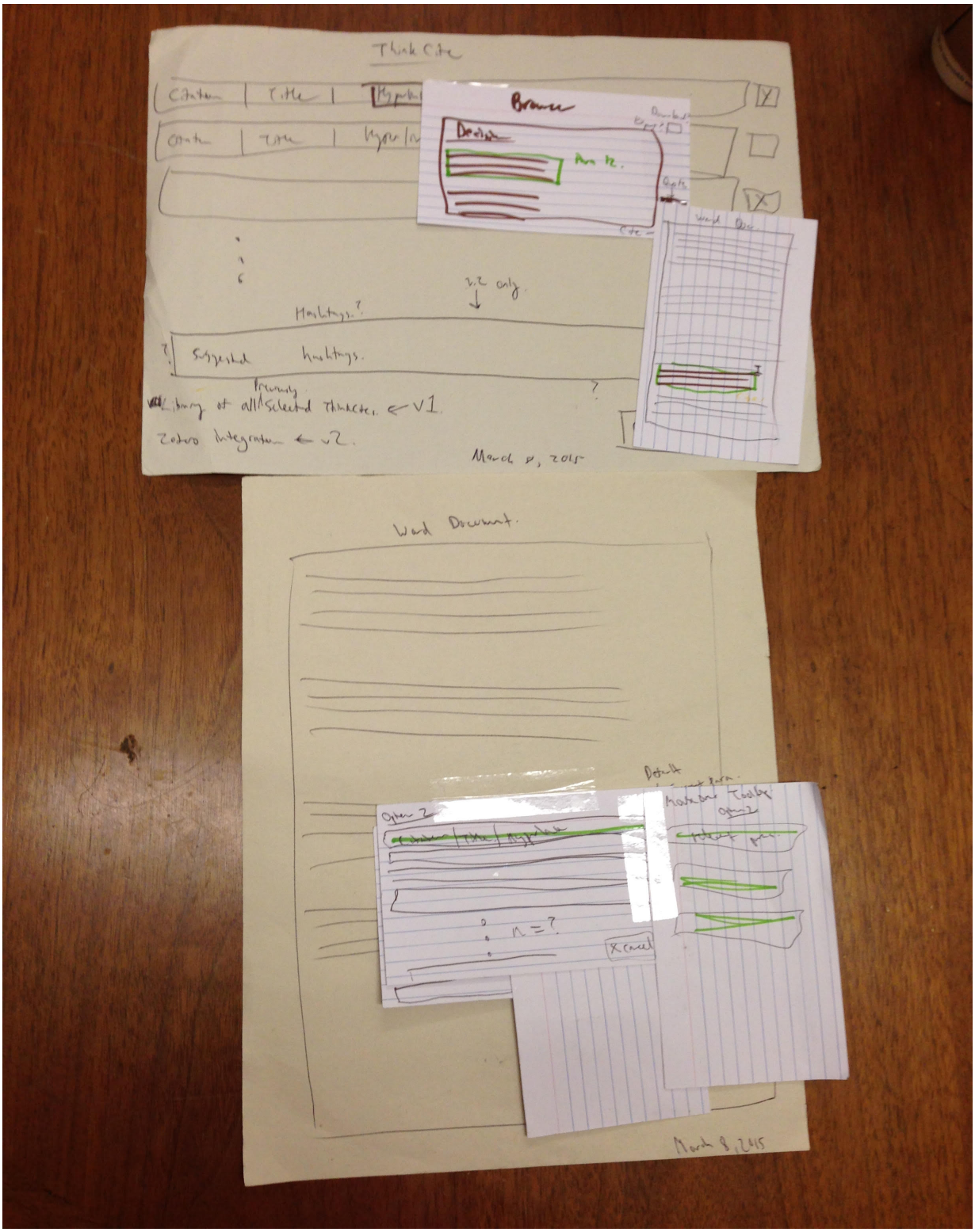
1. Overview - Design Process

Because we were developing a prototype of a relatively new type of software, we adopted an iterative approach for our design process model. In this model, a cyclical approach is taken: beginning with planning, to analysis and design, to implementation, testing, evaluation, and then leading to further planning again. The process may go through several iterations or sub-iterations of this cycle before the software is actually deployed. We used paper prototyping in the initial design phase in order to enable a collaborative design process, and to be able to make design changes based on feedback from other potential users without expending resources redeveloping software. We then formalized the paper designs using Balsamiq mockups, before implementing the working prototype. Even after the prototype was complete, we went through several more iterations of the design after internally testing it, in order to fix usability issues as they arose.

2. Paper Prototyping

On the following page can be found the paper prototype we created collaboratively:

Figure 1 - ThinkCite Paper Prototype



3. Digital Mockups

After receiving suggestions on ways to improve the paper prototype, we created digital mockups using Balsamiq modeling software:

Figure 2 - Main Window Mockup (without preview)

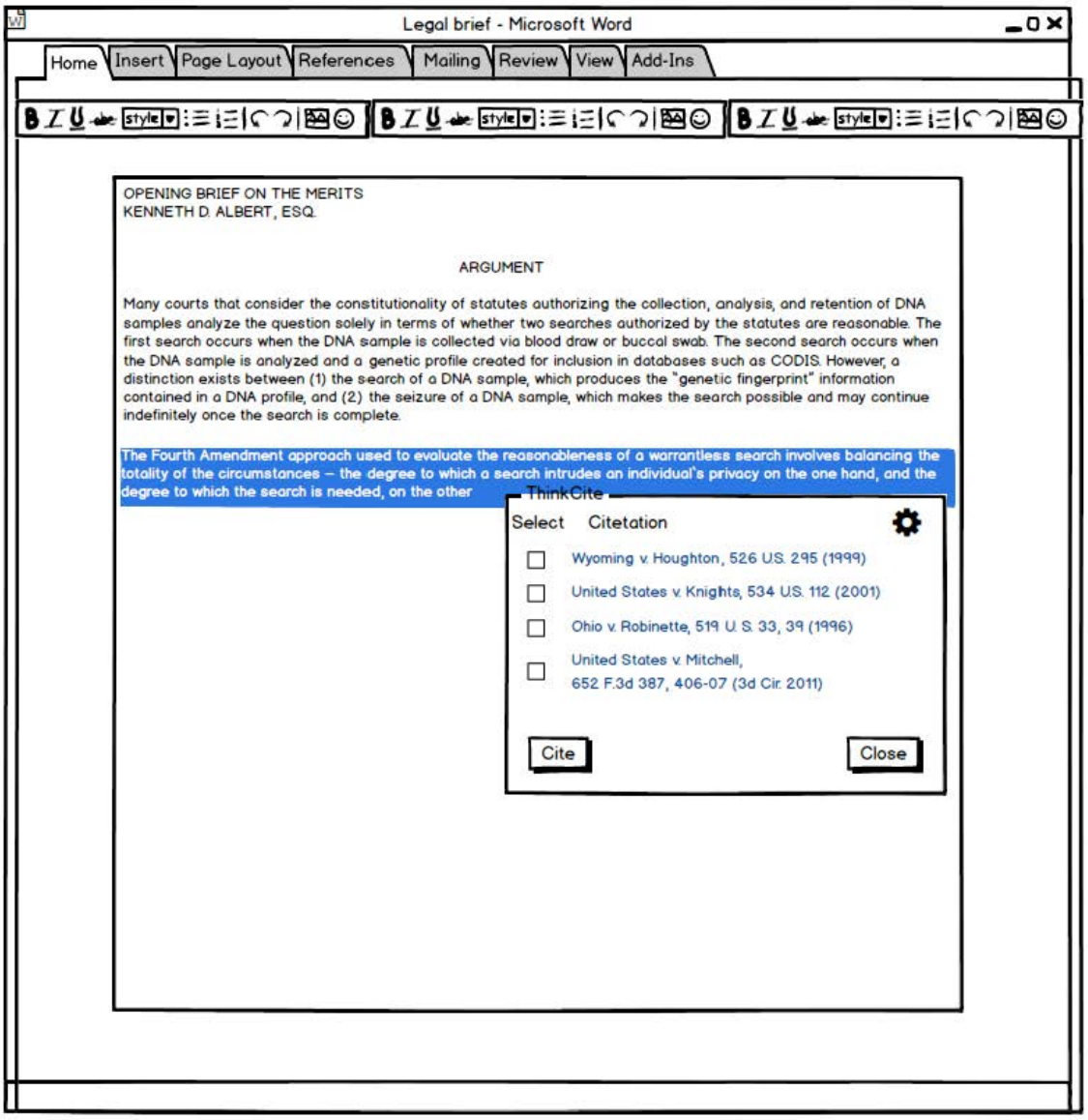


Figure 3 - Main Window Mockups (preview as popup, settings dialog)

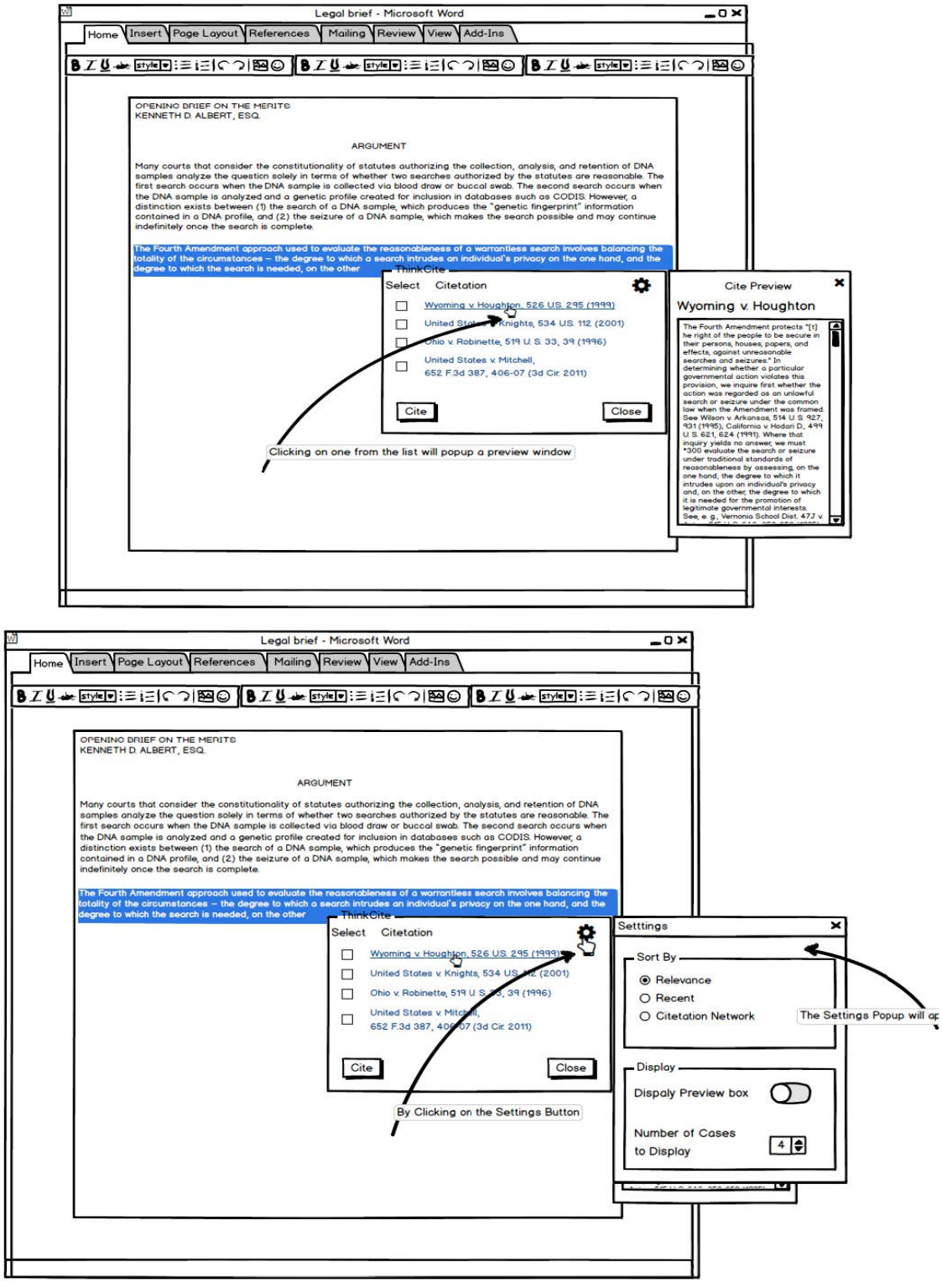


Figure 4 - Word Document with Citation Mockup

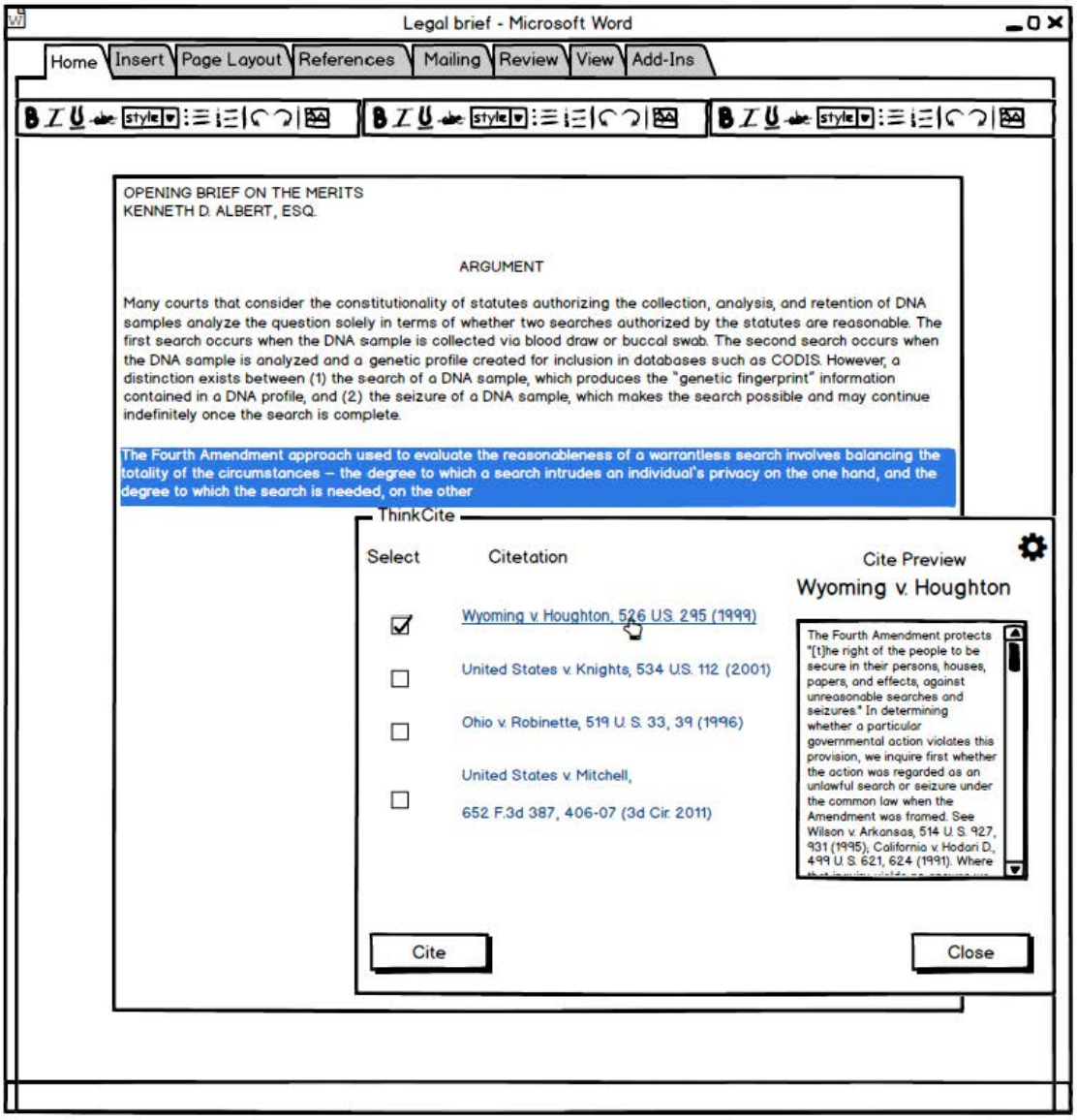
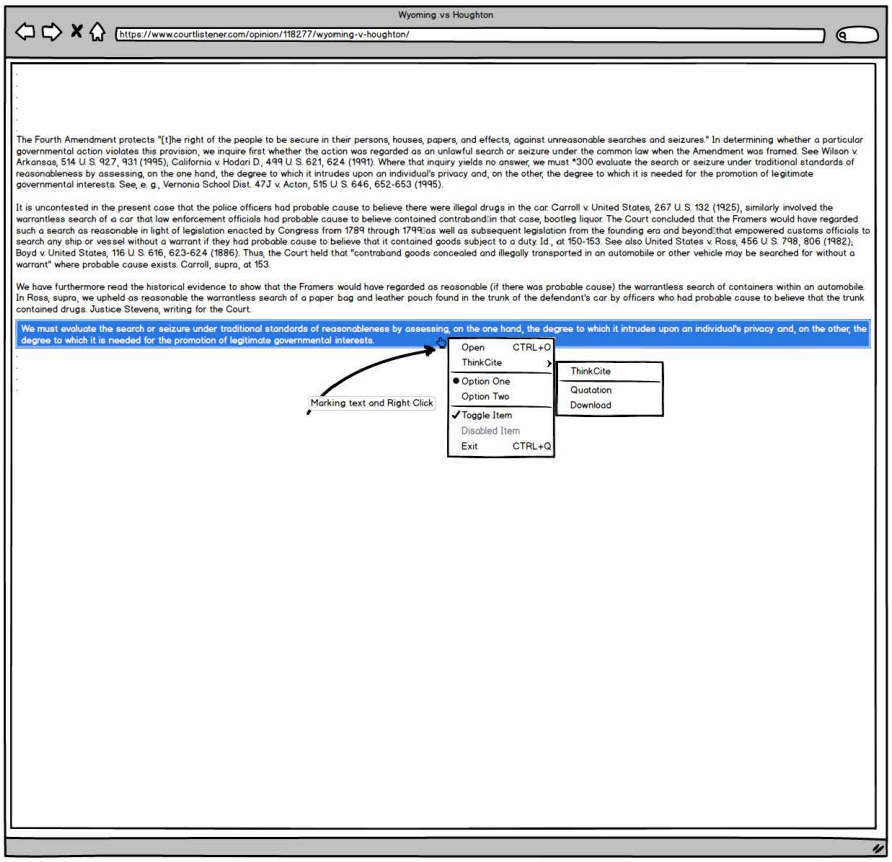


Figure 5 - Browser Window Mockup



4. Implementation

a) System Design

The constraints of our development environment affected our system design. For example, we were running MacOS, and thus did not have the ability to develop a heavyweight plugin for Microsoft Word that would rely on advanced features of Visual Basic .NET. Instead, we designed the Word plugin to be as lightweight as possible. It would simply be activated by a user key combination, would formulate a query based on the active paragraph or selection, and then pass control and information to the ThinkCite application and GUI. The ThinkCite application would then query the recommendation engine, and after results were returned by the engine, the GUI would display them and allow the user to further interact with them. Finally, once the user had chosen to output their selections back to the Word document, control and information would pass back to the Word plugin, which would then insert the information into the document.

b) Microsoft Word Plugin

We built the Microsoft Word Plugin as a macro using Visual Basic for Applications. The macro is activated by way of a keyboard shortcut (Ctrl+space). It retrieves the current paragraph or selection and then, in a shell, activates the ThinkCite GUI and passes the text query into the GUI on the command line, for further processing.

c) Graphical User Interface

The GUI was written as a standalone application in python, using PySide as a graphics/windowing library. The PySide package provides a set of python wrappers to Qt, a C++ graphics library. When cases are opened in a browser window

d) Recommendation Engine

The recommendation engine was written in python as well, using the gensim library to perform topic-modeling functions using LDA. A corpus of all United States Supreme Cour Cases was used, based on data from the Free Law Project and CourtListener.com. Several models were generated and one was chosen to use in the engine. Once the corpus had been analyzed by the LDA algorithm, resulting cases and their topic vectors were stored in a MySQL server for later retrieval during the query search operation.

Recommendation Engine

1. Overview

The recommendation engine was a key part of the project. We built it concurrently with the GUI. After receiving limited, but promising results on an automated test of our own design, we wrote the following paper describing the engine in detail. The paper was submitted on May 1 and accepted on May 6 for presentation at the ICAIL 2015 Workshop on Law and Big Data.

2. Research Paper

The following is the research paper that we wrote and submitted. Please note, it is not in final form so should not be cited or quoted without permission of the authors:

A Legal Citation Recommendation Engine Using Topic Modeling and Semantic Similarity

Talia Schwartz, LL.B., LL.M.
LL.M. Candidate
University of California, Berkeley
School of Law
taliashwartz@berkeley.edu

Michael Berger, B.Sc., J.D.
M.I.M.S. Candidate
University of California, Berkeley
School of Information
mjb@ischool.berkeley.edu

Juan Hernandez, B.A.
M.Sc. Candidate
Northwestern University
McCormick School of Engineering
juanhernandez2016@u.northwestern.edu

ABSTRACT

Topic models are statistical models that detect themes in text corpora. They can be used in information retrieval to find documents that are "similar" to a query, based on the similarity of the themes in the query to the documents in the retrieval database. Applying such models to the domain of legal research might help in improving the efficacy and accuracy of legal research and writing processes that currently rely, to a large extent, on specialized domain knowledge to conduct human-supervised information organization, query formulation, and document retrieval. In this paper we suggest a novel approach that incorporates automatic content analysis methods into the legal sphere and applies Latent Dirichlet Allocation (LDA) to assist in case retrieval when drafting legal documents. We develop a prototype, built for a popular word processor, that runs on a fixed corpus of sixty-four thousand United States Supreme Court cases. The tool is called while the user is developing a document, using the document itself to formulate a query. The tool detects the user's writing context to automatically formulate a query, and uses topic modeling methods to recommend relevant legal cases for citation and quotation within that context. The paper offers an initial evaluation of the method by testing performance using paragraphs from existing legal cases and their associated citations, showing that our algorithm provides an overall effective recommender system compared with the traditional manual-human legal querying method.

Keywords

Text analysis; Topic modeling; Latent Dirichlet Allocation (LDA); Legal citation; Legal Corpus; United States Supreme Court; Legal research; Information Retrieval; Document Similarity;

1. INTRODUCTION

This paper reports on a novel implementation of a recommendation algorithm that utilizes LDA in the legal context to provide a legal citation recommendations. Using natural language data processing while benefiting from the multi-topic nature of LDA, the proposed tool is well-suited to legal research and is capable of augmenting the abilities of domain-knowledgeable users.

In common law, precedent-based legal systems, case law is a primary source of law. Legal practitioners, academics, and students all routinely conduct legal research and turn to case law in the pursuit of the most relevant set of facts and applicable legal rules in a court decision or other legal document. Once a relevant document is found, it can be cited in support of an argument, or distinguished as inapplicable. The legal recommendation system reported in this paper, was thus motivated by the desire to use text analysis techniques to complement traditional legal research with an effective (accurate and relevant) and efficient (low time and effort) search tool. The extensive amount of digitized legal text available for search presents the problem of an ineffective legal search, a problem with which scholars have been trying to cope for several decades [8]. Another issue that the presented tool is intended to address is the human "tendency toward the familiar", as was also reported in the legal context [25]. Indeed, the incorporation of text analysis methods into Law and Artificial Intelligence has been recognized as a pressing need [2]. A study conducted in 2007 on the American case law network found a highly skewed distribution of authorities in the Web of Law; 2% of U.S. Supreme Court citation network comprise 56% of all citations. [27]. The ultimate meaning of this "rich get richer phenomenon", results in legal practitioners analyzing the legal sphere via a highly limited lens [9].

Previous attempts at using automated means for legal research have sometimes been met with difficulty due to technological limitations in both hardware and software. But with recent advances in both hardware capabilities and new algorithmic techniques, the technology is ripe for further exploration of legal search tools. Such tools have the additional potential of normalizing the distribution of case law usage and authority in the Web of Law. The need for an efficient legal citation tool is complemented and reinforced by the existence of advanced data mining and text analysis techniques as well as a more open, digitized environment from which a legal corpus of data can be drawn. The LDA algorithm is of particular interest as it can produce higher-dimension topics/clusters than other topic modeling and clustering algorithms that have been applied to this problem in the past.

To meet this end, a system was developed which recommends a case that is relevant to a text query that has been written in natural language. The described system runs on a data set of the United States Supreme Court decisions, which consists of 63,547 documents.

The rest of the paper unfolds as follows. The second section provides an overview of previous work utilizing text analysis techniques in various field—and in particular, with regard to legal information extraction and retrieval systems and methods. Next, the methods used in our proposed model are described, namely, the application of topic modeling / Latent Dirichlet Allocation to a legal corpus. The fourth part provides our results, including

results from a novel automated testing method devised by the authors to test the practical performance of the tool built. In the last part, conclusions and further research are discussed.

2. PREVIOUS WORK

The use of probabilistic topic models to identify systematic patterns and commonalities in data has applications in diverse fields, from medicine [1] to political science [12][17]; and is implemented for such purposes as text mining [19], social network analysis [6], customer purchasing behavior [14], language modeling [3], and others. The application of collaborative topic modeling to a recommendation system for scientific articles [31] is a utilization of LDA that shares some features with the approach taken in this paper. The application of text analysis methods in the legal sphere is limited, albeit not novel. Automated content analysis and semantic interpretation of legal text has been implemented for the purposes of case summary [20][21] legal outcome prediction [2]; identification of justices' ideology [16] or quantifying the "complexity" of a written opinion [22]; determining consensus among justices [23]; attributing authorship in unsigned court opinions [18]; and as a litigation support system in the field of Online Dispute Resolution (ODR) [10].

The presented recommendation system involves several lines of research within Law and AI, namely, knowledge representation, legal retrieval systems, and information extraction using various text analysis methods. The SALOMON system detects relevant legal text by word-frequency techniques and performs retrieval based on similarity measures, while acknowledging the potential of learning topics by the grouping of text [20][21]. Other models extract semantic information and classify legal text by pre-defined concepts [5], or suggest retrieval based on the k-Nearest Neighbors approach [2][24]. Natural Language Processing techniques have been also implemented in automatic classification of legal case factors [2][32] and in extracting ontologies from legal text [15]. A recent work by El Jelali et al. introduces an information retrieval system that uses machine learning and natural language processing techniques to match disputant case descriptions with court decisions [10]. Text mining and analysis have been implemented outside the academic realm as well, in aiding legal work from a practitioner's perspective. There are several designated Information Retrieval systems in the field of Online Dispute Resolution (ODR)[1].

As has been done in more recent work [10], the tool reported here uses natural language inputs rather than pre-defined, constructed, template-based methods, which were more common in older systems. Critically, the model presented here differs from previous work in the application's methods and goals. To the best of our knowledge, there is no previous published work that utilizes LDA in the context of the legal domain in order to build a legal document recommendation system. The advantage of LDA over many other methods that have been attempted before lies in its modeling of documents as distributed over multiple topics, rather than a single topic, as is assumed in clustering approaches such as K-means and K-medoids. In modeling legal decisions as topic distributions, and topics as distributions over words, LDA accounts for the multi-topic nature of a court case (a single case, or even a single part of a case, may naturally address several legal topics.) Another advantage of LDA is that it learns the relevant item representation in a fully unsupervised, automatic fashion, eliminating the need for manually curated topics or human-enabled feature extraction. Moreover, a significant portion of former work on legal information processing and retrieval focused on semantic inferences that could be gleaned from the legal text. While the suggested topic model algorithm does embody semantic similarity measures [26] rather than a word-based approach [10], in its simplicity, it overlooks the complexity of judicial language. In other words, the proposed tool uses text-analysis techniques that take into account similarity measures while being "agnostic" to words' linguistic meanings, differently from other work [8][10][15][19][32]. Another feature that distinguishes the work reported here is a practical testing technique, which automatically analyzes 1000 queries extracted from a United States Supreme Court Case database.

3. METHODS

Our aim was to design a tool for one type of use-case that is common among legal researchers—i.e. the task of finding a case to cite that is relevant to a portion of text that has already been written in a legal document. Although this is not the only conceivable use-case for a legal research tool, it is a fairly common one, and has the potential to be modified to support other uses.

The methods we employed are directed toward this use-case, and can be summarized as a development phase and a testing phase: first, we build a recommendation engine using LDA that suggests legal citations that are most relevant to a given query from a user (that query consisting of one or more words of natural language text); and second, we test the recommendation engine using an automatic process that simulates the expected use-case. In this model, "relevance" is defined by topic similarity: the most relevant document is the document that is most similar in topics to the text query. In the first phase, we build several different models by varying certain parameters, such as K (the number of topics) and alpha (a hyperparameter used in LDA, described below).

In this section we outline the data used, the steps used for processing data, the method for approximating LDA, and our testing framework.

3.1 Latent Dirichlet Allocation

LDA is a hierarchical Bayesian generative model, in which documents are represented as a mixture of a limited set of topics, where each topic is characterized by a distribution over words [3]. The model, which allows the representation of documents in a reduced feature space consisting of K dimensions (rather than a larger space that contains as many dimensions as the number of unique words in a given corpus) is thus used as a method of unsupervised machine learning to discover latent topics that are hidden in a text.

[1] For an overview see Carneiro D, Novais P, Andrade F, Zeleznikow J, Neves J.,Online dispute resolution: an artificial intelligence perspective. Artif Intell Rev 41(2):211–240 (2014).

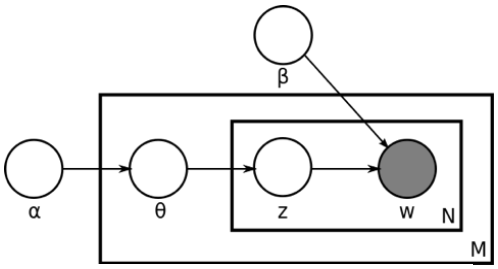


Figure 1. ***Blei et al. 2003, page 997***

LDA can be represented by plate notation to describe the generative process wherein each document in the corpus is generated by picking a multinomial topic distribution $\theta \sim \text{Dirichlet}(\alpha)$ for the document d. Each word of the document is assigned a topic from the topic distribution. Given the topic, a word is drawn from the multinomial word distribution $\beta k \sim \text{Dirichlet}(\eta)$ from the dictionary for that topic.

The method used in our model for approximating the LDA is the Online Learning algorithm, described by Hoffman et al., a variation on the Expectation Maximization approach [13]. The Online Learning algorithm is streamed and runs in constant memory with regard to the number of documents and can make use of a distributed system, which allows it to be implemented on a much larger corpus.

3.2 Data

3.2.1 Data Source

Our data set is the entire corpus of United States Supreme Court decisions available on courtlistener.com,[2] which consists of 63,547 court decisions. The data span a period of over two centuries of deliberations by the United States Supreme Court, from 1754 to 2014. For the purpose of creating our models we sampled fifty percent of the mentioned corpus, meaning, about 32,000 documents, to form a modeling corpus. Once a model was learned from the modeling corpus, topics were inferred on the entire corpus using the trained model, and these topic distributions were used to make the recommendations that we then tested, as described further below.

3.2.2 Data Preprocessing

As part of preprocessing, data cleaning for LDA consisted of removing punctuation and stop words, in which each document in the corpus was transformed into a "bag of words" representation. Moreover, we created two different corpora to test whether or not stemming helped to reduce noise and produce LDA models that represented topic distribution more accurately.

[2] CourtListener, a legal research website sponsored by the non-profit Free Law Project, was established in 2010 by Brian W. Carver and Michael Lissner (University of California, Berkeley School of Information). Michael Lissner, CourtListener.com: A platform for researching and staying abreast of the latest in the law, Masters Thesis, University of California, Berkeley, School of Information, (May 7, 2010). For the Supreme Court database see https://www.courtlistener.com/?q=&court_scotus=on&order_by=score+desc (last visited April 16, 2015).

3.3 Creating and Evaluating Models

The modeling corpus is divided into a training set - consisting of 80% of the corpus selected randomly (approximately 26,000 documents) - and an evaluation set consisting of the remaining 20% of the corpus. To create each model, a set of K topics are first learned from the training set by applying Latent Dirichlet Allocation [3] to learn topics. Each model was then evaluated for perplexity based on the remaining evaluation set, as described in section 4.2. The LDA algorithm uses two user-determined parameters, α and K, which alter the sparsity of the topic distribution and the number of topics, respectively. The best model for which we had the computational resources to implement was then chosen using the perplexity measurement described in the Analysis section below. The best model built on the modeling corpus is then applied among the entire corpus of documents to infer the topic distribution of each document therein, representing each document as a vector of length K. This model underlies the recommendation engine - the topic vector of each document, as computed by the model, is stored in a database.

3.4 Retrieving Relevant Citations

In order to perform the information retrieval task, the engine receives a query consisting of a text input automatically extracted from the user's draft document. It then searches for relevant documents as follows: (1) the user's text is transformed into a K-dimensional vector representing that query's topic distribution using the same LDA model learned from the training corpus; (2) the similarity of the query topic vector to the topic vectors of each of the documents in the database is computed, and (3) the documents are retrieved as citation recommendations in descending order of their semantic similarity, as measured by a distance measure between the topic distributions.

We compare four different distance measures: Kullback-Leibler (KL) Divergence,

$$KL(p,q) = \sum_{i=1}^{T} p_i \log \frac{p_i}{q_i}$$

Information Radius (IR),

$$IR(p,q) = \sum_{i=1}^{T} p_i \log \frac{2 \times p_i}{p_i + q_i} + \sum_{i=1}^{T} q_i \log \frac{2 \times q_i}{p_i + q_i}$$

Hellinger Distance (HD)

$$HD(p,q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{1}^{T} (\sqrt{p_i} - \sqrt{q_i})^2}$$

and Manhattan Distance (MD).

$$MD(p,q) = 2 \times (1 - \sum_{1}^{T} \min(p_i, q_i))$$

In all the above formulae, p and q are the topic vectors to be compared, and p_i and q_i are each of the co-ordinate elements of the respective topic vectors.

These distance measures were used to create a document-to-document similarity coefficient following the example of Dagan et al. (1997) and Rus et. al (2013)

$$SIM(p,q) = 10^{-\delta IR(c,d)}$$

Another version of the retrieval algorithm weighs the similarity coefficient alongside a measure of the retrieved documents' popularity (i.e. the number of times each document has been cited before). The weighting is accomplished by multiplying the similarity by a coefficient of between 0.8 and 1.2, scaled to the total distribution of document citations. The coefficient boundaries were selected empirically to be conservative in order to prevent overweighting—*i.e.*, to prevent the popularity measure of a document from having an excessive effect on the ranking and overpowering the similarity measure.

3.5 Testing the Recommendation Engine

Besides the standard evaluation methods for topic models, we set up the following practical testing scenario as another method to determine the optimal parameters, as well as test the relevant performance of our two[3] corpora: We randomly selected 1000 paragraphs from cases in the US Supreme Court database, each of which contains some text and a citation to another Supreme Court decision. The citations were then separated from the paragraphs, and each of the paragraphs were used as query text to be input into the recommendation engine. Using the above-described algorithm, we retrieve a sorted list of recommended citations for each test paragraph. Of course, we removed any citations that were not dated earlier than the document from which the test paragraph was taken. Our metric for performance was the position of the actual cited document in the recommendation ranking. Ideal performance would be when the number one recommendation in the ranking matches the actual citation that had been originally cited in the paragraph. We compared how well each of our models, similarity measures, and weighting with popularity ranks the citation found in the paragraph. The optimal combination of parameters will be used in the tool implementation.

4. RESULTS AND ANALYSIS

4.1 Model Parameter Choices

We developed two sets of LDA models with 15, 30, 50, 100, 200, 300, 400, 500, 600, 700, 800, and 900 topics, one set using stemming and another set without stemming words. In addition to comparing the number of topics, we tested the difference in results from selecting α as 50/k (a rule of thumb following Griffiths and Steyvers, 2004) and dynamically learning α from the data.

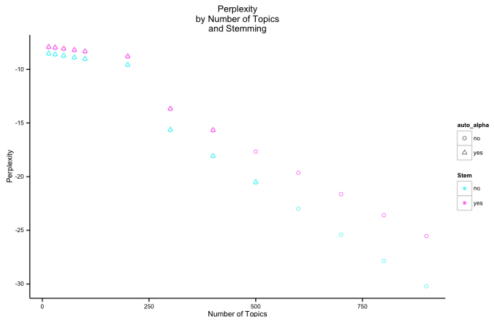
4.2 Model Evaluation

As described in the Methods section above, each model was evaluated in an initial evaluation stage. This was done by calculating the per-word likelihood and perplexity, using the 20% holdout of the modeling corpus as an evaluation set. Intuitively, perplexity is an indicator of the ability of a given model to predict the next word in a given document, and the lower the perplexity, the better the model is at doing so [30]. More formally, perplexity is derived from log-likelihood of unseen documents and is calculated as follows, where w is the set of unseen documents (and w_d refers to each document in the set), $\mathbf{\Phi}$ is the topic matrix, and α is Dirichlet prior of the topic distribution:

$$\mathcal{L}(w) = \log p(w|\mathbf{\Phi}, \alpha) = \sum_d \log p(w_d|\mathbf{\Phi}, \alpha).$$

$$\text{perplexity(test set } w) = \exp\left\{-\frac{\mathcal{L}(w)}{\text{count of tokens}}\right\}$$

The figure below shows the results of our evaluation of the perplexity of the models described above. Since automatic alpha parameter setting was computationally more expensive and yielded no relative benefit, we discontinued it.

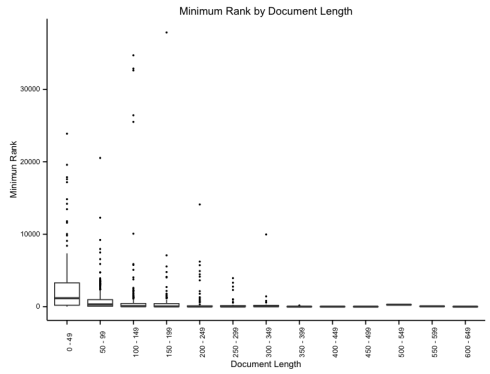


Although the apparently superior models contained over 500 topics, computing resource restrictions permitted us only to do the next phase of the testing with a 200-topic model by the publication deadline. Additional testing with the models of greater than 200 topics is currently being performed, with more results expected in the coming weeks.

4.3 Recommendation Engine Testing Results

Using the 200-topic model in the engine, we tested the recommendation performance of the engine using the procedure described in the Methods section above. This additional procedure measured the ability of the recommendation engine to retrieve the citation that we knew was in some way relevant to a given text query paragraph (because it had been cited by the Supreme Court in relation to that paragraph). In the graph below, we show how the length of the query paragraph ("Document Length") affected the ranking of the "correct" citation returned by the recommendation engine ("Minimum Ranking").

[3] (1) Corpus with stemming, and (2) Corpus without stemming.



The graph above shows that performance stabilizes as the query length exceeds 250 words. These results have implications for recommendation tool design: using a 200-topic model, a recommendation tool built on this engine should extract 250 words of context from the user's document in order to retrieve better results.

We investigated the outlier points in the boxplot and found that they represented documents from the data set where the parser failed to parse a query paragraph correctly. Including query cases of every length, as well as outliers, our initial overall quantile results were as follows, where the top row shows the percentile and the bottom row shows the minimum ranking of the cited document:

10 %	20 %	30 %	40%	50%	60%	70%	80%	90%	100%
2	8	24	53.6	135.5	286.8	632.9	1223.8	2924.9	37854

Focusing on query cases that were of length greater than 250 words, we report the quintile results below:

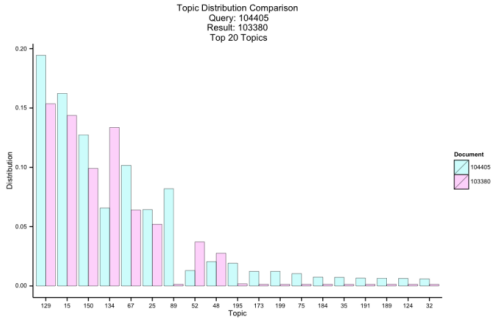
10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0.6	2	3	6.4	13	27.2	59	174.6	703.2	9970

Thus, for queries of length greater than 250 words, the recommendation engine built on the 200-topic model weighted with popularity returned the original citation associated with that query in its top 3 results 30% of the time, and in the top 59 results at least 70% of the time. These results are very encouraging for an initial test, and we are now iterating on this design by engaging additional computing resources to test the models we created that had a larger number of topics, K.

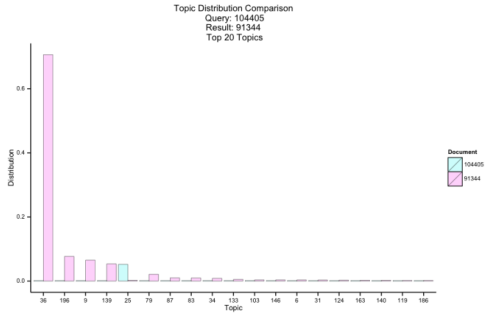
4.4 Illustrative Query Results

To illustrate more intuitively how the recommendation engine was able to find the matching document to many of the query paragraphs, we provide an example of both a matching and non-matching document to a single query.

The graph below shows the probability distribution of the top 20 topics in an example query (blue), and compares them with the distribution over the same 20 topics in the highest ranked document returned by the engine (pink). As can be seen, the topic distributions are fairly well matched:



Compare this with the next graph, which shows the same query compared with a low-ranked document returned from the engine. As can be seen, the top 20 topics in the distribution for the comparison document (pink) are barely found in the probability distribution for topics in the query (blue):



5. CONCLUSIONS

To the best of our knowledge, this was the first attempt of its kind to create a reliable legal recommendation system based on LDA topic modeling, and the first attempt at testing such a system by automated means. As described above, initial results have been encouraging, and further research and testing is currently being undertaken.

The tool's admittedly circumscribed ability to detect the cited document among its top recommended results is nevertheless worthy of further exploration, given the limited amount of topics in the model we were able to test with our set of computing resources. Additional models with higher number of topics and lower perplexity are currently being tested to see if they improve performance on the automated test set. While results points to the need for more topics in the models, an optimal number of topics should be learned from additional tests.

Additional modifications to the data set in the pre-processing stage, as well as further changes to the ranking method, are also candidates to be considered for improving the system's overall performance. Moreover, for the purpose of this research, the data set was intentionally left unfiltered. Further research might

consider filtering out some of the documents based on noise (documents with low to non-alphabetic content) and legal relevance (for example, the corpus consisted of a large number of decisions dismissing writs of certiorari with very little content.) Another route for further research and development could consider using existing documents and their citations to learn (using supervised machine learning) the optimal weights for popularity and other metadata available in our corpus to improve recommendations.

This work also raises additional interesting avenues of research exploration in the domain of testing. Whereas the automated testing techniques described rely on judicial language, additional testing could be performed to evaluate the tool's performance given text queries generated by laypeople, students, or legal practitioners with different linguistic styles. A user experience study could be conducted among legal practitioners to determine the tool's efficiency (low time and effort), by comparing the legal research user experience with the aid of the suggested tool versus traditional legal research and writing methods. Furthermore, testing is required among testers with domain specific knowledge to evaluate the effectiveness of the system, in terms of the recommendation accuracy and relevance. By collecting user data, this information could also be used to optimize the recommendation, using supervised learning, in a similar way to the calibration framework suggested above.

Yet another direction of future research could be the substitution of alternative and newer methods of topic modeling, such as Heirarchical Dirichlet Process (HDP) and Nested Chinese Restaurant Proces (nCRP) [4][28] in an attempt to contribute to the sophistication of the model and its accuracy, by implementing/learning topic hierarchies, rather than flat lists, within the legal texts.

Finally, from a normative and policy perspective, additional research should consider the implications of the discussed technology on the legal industry and profession, such as its potential narrowing effect on legal research and writing skills, or its potential in widening the distribution of case law networks.

6. ACKNOWLEDGMENTS

The authors wish to thank Brian Carver and the Free Law Project for their generous provision of data. This resource was invaluable in conducting our analysis. We also acknowledge the UC Berkeley DLab, its director Justin McCrary, the Computational Text Analysis Working Group led by Nicholas Adams and Brooks Ambrose, for their excellent support and assistance. Finally, we thank the UC Berkeley School of Information and its Computing & Information Services Unit for their technical and application support, access, and hosting.

7. REFERENCES

[1] Arnold, C. El-Saden, S. Bui, A. Taira, R. 2010. Clinical case-based retrieval using latent topic analysis. In: *Proc. AMIA* 26.

[2] Ashley, K.D. Brüninghaus, S. 2009. Automatically classifying case texts and predicting outcomes. *Artificial Intelligence and Law* 17(2), 125.

[3] Blei, D.M., Ng, A. and Jordan, M.I. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*. 3, 993.

[4] Blei, D.M., Griffiths, T.L. and Jordan. M.I. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)* 57(2), 7.

[5] Brüninghaus, S. Ashley, K.D. 2001. The role of information extraction for textual CBR. In: *Proceedings of the 4th international conference on case-based reasoning* (ICCBR-01, Vancouver, CA). Springer Lecture Notes in Artificial Intelligence, Springer, Berlin.

[6] Cha, Y. and Cho, J. 2012. Social-network analysis using topic models. *SIGIR '12*.

[7] Dagan, I. Lee, L. and Pereira, F. Similarity-based methods for word sense disambiguation. 1997. *Proceedings of the 35th ACL/8th EACL*, 56.

[8] Daniels, J.J. Rissland, E.L. 1997. Finding legally relevant passages in case opinions. In: *Proceedings of the 6th International Conference on Artificial intelligence and Law*, ACM. New York, NY, USA 39.

[9] Delgado R. and Stefancic, J. 2007. Why Do We Ask the Same Questions? The Triple Helix Dilemma Revisited, 99 *LAW LIBR. J*. 307.

[10] El Jelali, S. Fersini, E. and Messina, E. 2015. Legal retrieval as support to eMediation: matching disputant's case and court decisions. *Artificial Intelligence and Law* 23(1), 1.

[11] Griffiths, TL. Steyvers, M. 2004. Finding Scientific Topics. In *Proceedings of the National Academy of Sciences of the United States of America*, 101, 5228.

[12] Grimmer, J. 2010. A Bayesian Hierarchical Topic Model for Political Texts: Measuring Expressed Agendas in Senate Press Releases. *Political Analysis* 18(1), 1.

[13] Hoffman, M. Blei, D. and Bach, F. 2010. Online learning for latent Dirichlet allocation. *Neural Information Processing Systems*.

[14] Iwata, T. Watanabe, S. Yamada and T. Ueda, N. 2009. Topic tracking model for analyzing consumer purchase behavior. In *IJCAI*.

[15] Lame, G. 2004. Using NLP techniques to identify legal ontology components: Concepts and relations. *Artificial Intelligence and Law* 12(4), 379.

[16] Lauderdale, B. and Clark, T. 2012. The Supreme Court's Many Median Justices. *The American Political Science Review* 106(4), 847.

[17] Laver, M. Benoit, K. and Garry, J. 2003. Extracting Policy Positions from Political Texts Using Words as Data. *The American Political Science Review* 97 (2), 311.

[18] Li, W. Azar, P. Larochelle, D. Hill, P. Cox, J. Berwick, R. C. and Lo, A. W. 2013. Using algorithmic attribution techniques to determine authorship in unsigned judicial opinions. 16 *Stanford Technology Law Review*, 503.

[19] McCarty, L.T. 2007. Deep semantic interpretations of legal texts. In *Proceedings of the eleventh international conference on artificial intelligence and law*, 217.

[20] Moens, M.F. Gebruers, R. and Uyttendaele, C. 1996. SALOMON: Final Report. Technical Report ICRI, K.U. Leuven.

[21] Moens, M.F., Uyttendaele, C. and Dumortier, J. 1999. Abstracting of Legal Cases: The Potential of Clustering

Based on the Selection of Representative Objects. *Journal of the American Society for Information Science* 50(2), 151.

[22] Owens, R. and Wedeking, J. 2011. Justices and Legal Clarity: Analyzing the Complexity of U.S. Supreme Court Opinions. *Law & Society Review* 45(4), 1027.

[23] Rice, D. and Zorn, C. J. 2014. The Evolution of Consensus in the U.S. Supreme Court. Available at SSRN: http://ssrn.com/abstract=2470029.

[24] Riloff, E. 1996. An Empirical Study for Automated Dictionary Construction for Information Extraction in Three Domains. *Artificial Intelligence* 85, 101.

[25] Ruhl, J. B. and Katz, D. M. 2015. Measuring, Monitoring, and Managing Legal Complexity. Iowa Law Review, 100, Vanderbilt Public Law Research Paper No. 15-1.

[26] Rus, V., Niraula, N. and Banjade, R. 2013. Similarity Measures based on Latent Dirichlet Allocation. In: *Proceedings of the 14th International Conference on Intelligent Text Processing and Computational Linguistics* (CICLing 2013). Springer Berlin Heidelberg, Samos, Greece, 459.

[27] Smith, T. A. 2007. The Web of the Law, *44 San Diego Law Review*, 309.

[28] Teh,Y. W. Jordan, M. I. Beal, M. J. and Blei, D. M. 2006. *Journal of the American Statistical Association*, 101 (476), 1566.

[29] Wallach, H. M. 2006. Topic modeling: beyond bag-ofwords. *ICML '06*. ACM.

[30] Wallach, H.M. Murray, I. Salakhutdinov, R. and Mimno. D. 2009. Evaluation methods for topic models. In L. Bottou and M. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*.

[31] Wang, C. and Blei, D.M., 2011. Collaborative Topic Modeling for Recommending Scientific Articles. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 448.

[32] Wyner, A. and Peters, W. 2010. Lexical semantics and expert legal knowledge towards the identification of legal case factors. In Radboud Winkels, editor, *Proceedings of Legal Knowledge and Information Systems (JURIX 2010)*. IOS Press. 127.

Final Design

1. Screenshots of Final Design

Figure 6 - ThinkCite Main Window After Activation

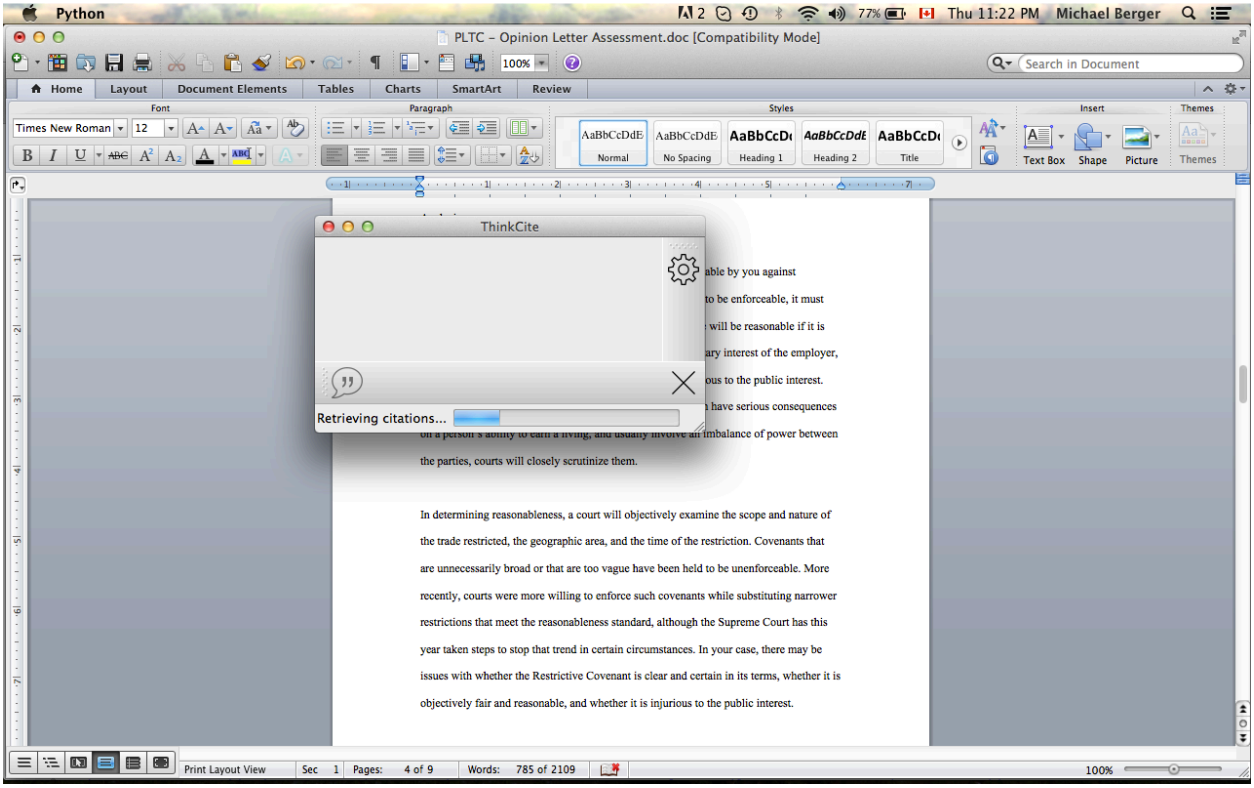


Figure 7 - ThinkCite Main Window Showing Query Results

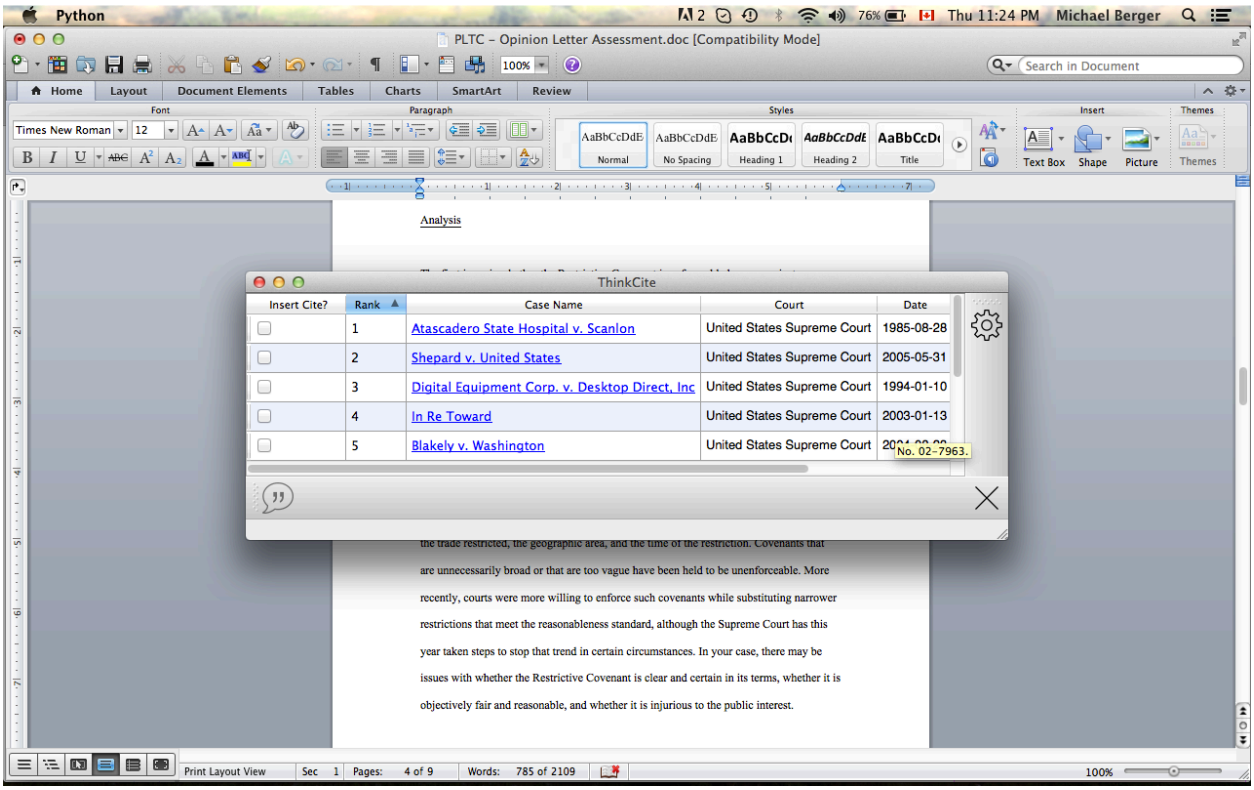


Figure 8 - Selecting Citations for Insertion

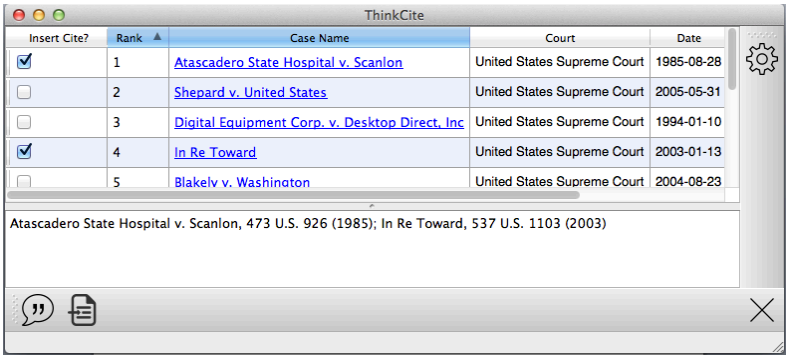
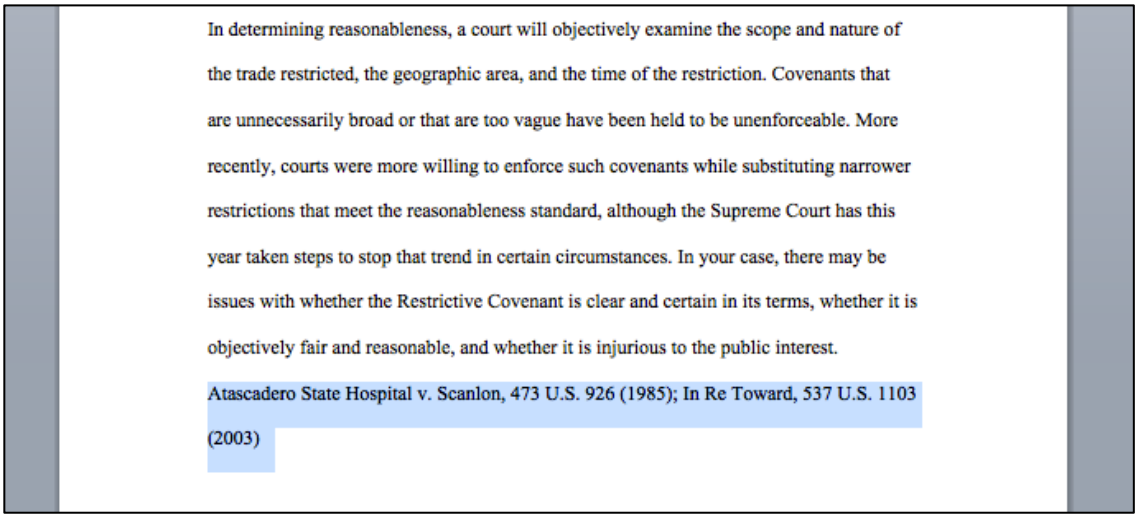


Figure 9 - Citations after Insertion into Word Document



In determining reasonableness, a court will objectively examine the scope and nature of the trade restricted, the geographic area, and the time of the restriction. Covenants that are unnecessarily broad or that are too vague have been held to be unenforceable. More recently, courts were more willing to enforce such covenants while substituting narrower restrictions that meet the reasonableness standard, although the Supreme Court has this year taken steps to stop that trend in certain circumstances. In your case, there may be issues with whether the Restrictive Covenant is clear and certain in its terms, whether it is objectively fair and reasonable, and whether it is injurious to the public interest.

Atascadero State Hospital v. Scanlon, 473 U.S. 926 (1985); In Re Toward, 537 U.S. 1103 (2003)

Figure 10 - ThinkCite Browser Window Displaying Case Text from CourtListener.com

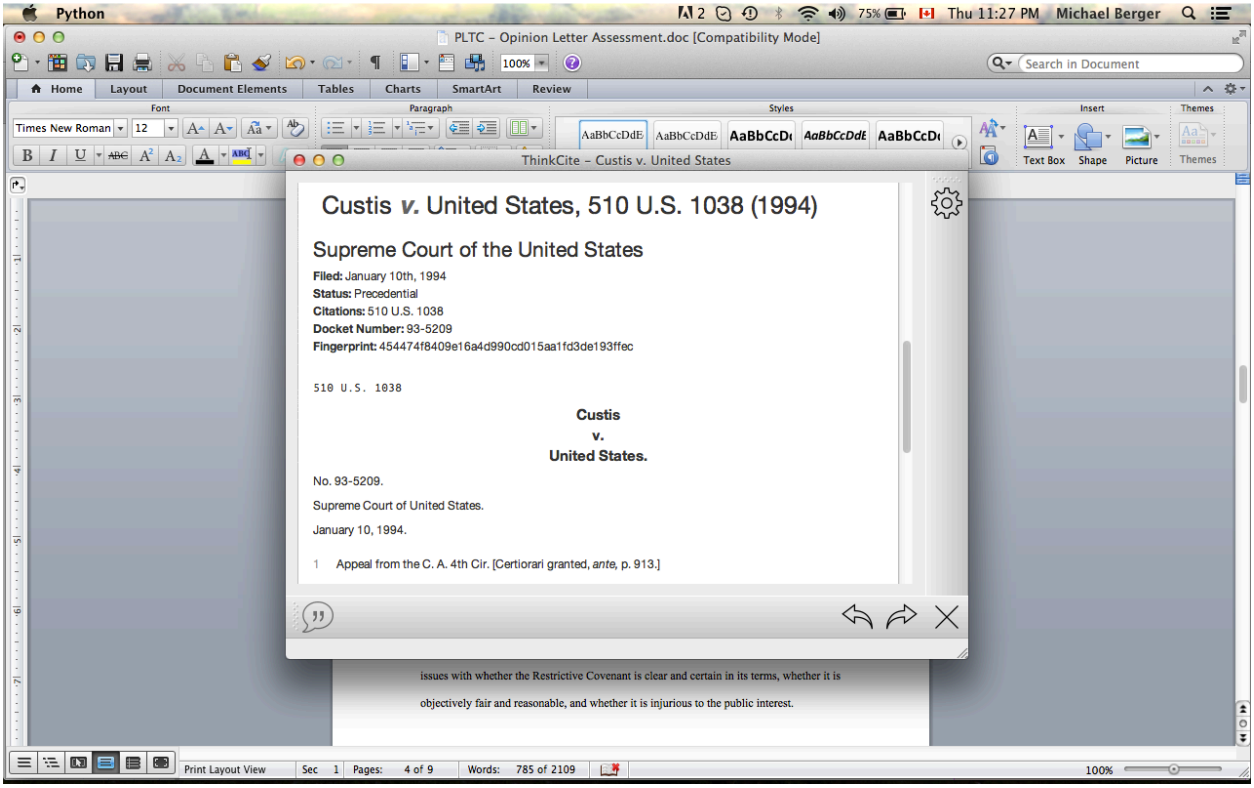


Figure 11 - Selecting a Quotation in Browser Window for Insertion

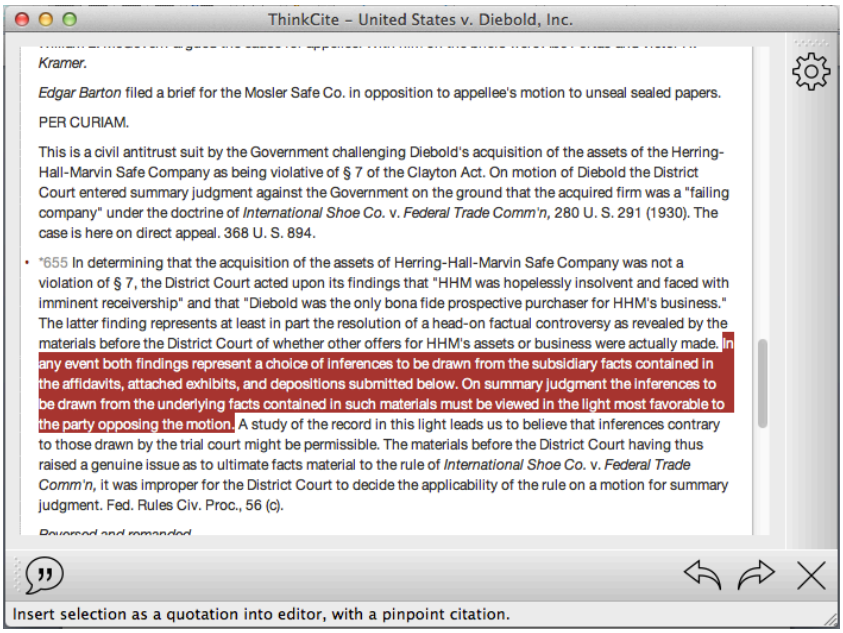


Figure 12 - Quotation with Pinpoint Citation Ready for Insertion into Word Document

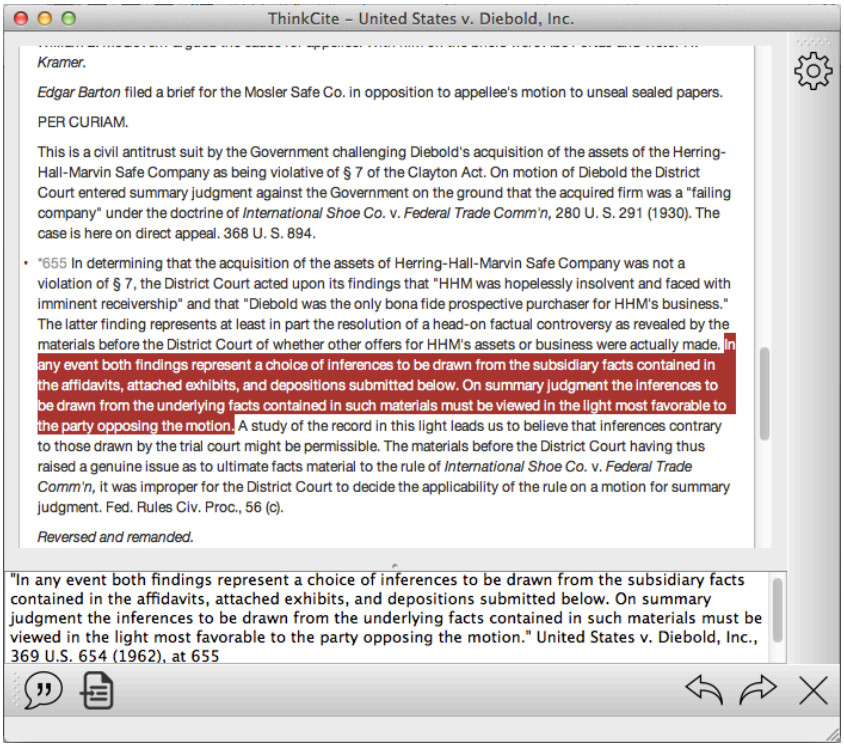
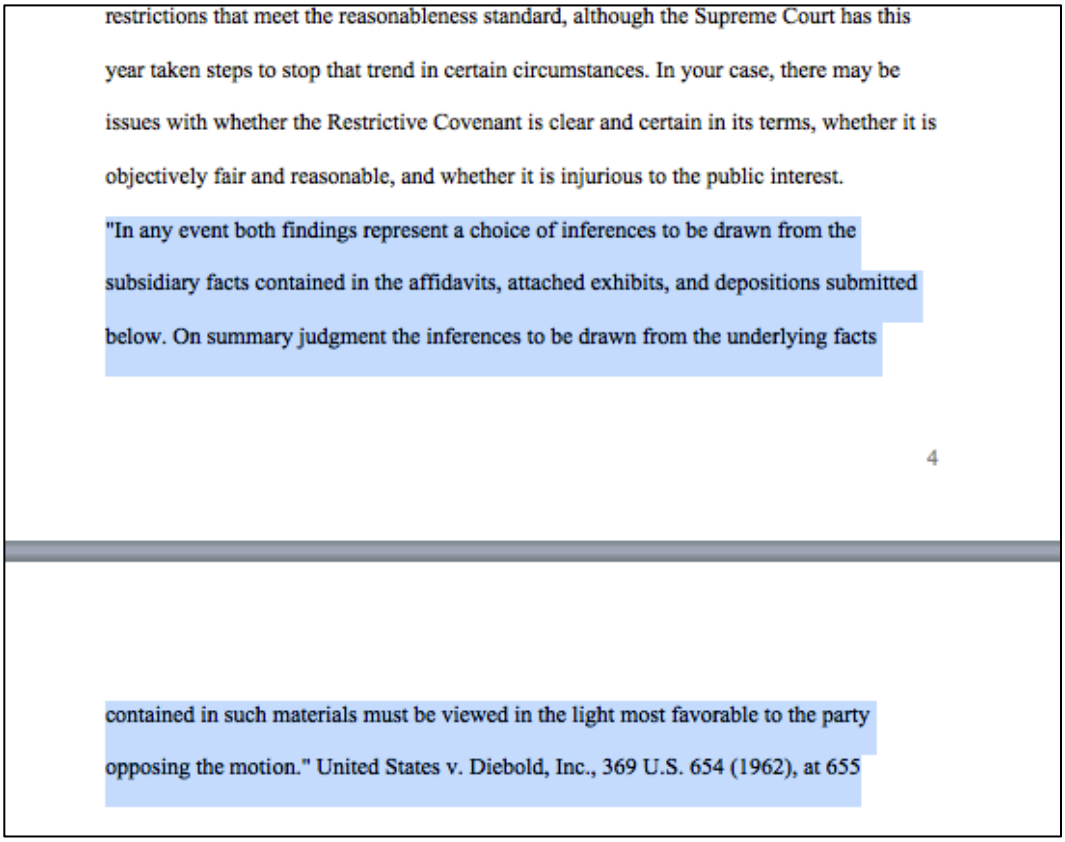


Figure 13 - Quotation with Pinpoint Citation after Insertion into Word Document



Results and Challenges

1. Recommendation Engine

As indicated in the submitted paper, our results for the recommendation engine test were fair, but encouraging enough to warrant further work. More specific results are discussed in detail in the "Results and Analysis" section of the paper. We were somewhat challenged by the computing resources available to us. For example, for the I School server to apply the various permutations of the 100 and 200 topic models to the entire SCOTUS corpus took approximately two weeks of computation. It would have taken even longer to apply the 300 and 400 topic models, and these were the models that showed the most promise during our initial model evaluation.

2. User Interface

We were able to implement most of the planned features in the user interface prototype. One of the most important features - integration with Microsoft Word - was accomplished successfully. The results returned by the recommendation engine were successfully displayed to the user in the GUI, and the user is able to select those results that are most pertinent to their task. The browser feature was also successfully implemented. Clicking on the links in the results table brings up a browser that displays the entire text of the case from CourtListener.com. One of the most challenging features that was successfully implemented was the quotation feature, which allows the user to highlight text in the browser and have it inserted into the document. The most difficult part of this feature to implement was the calculation and rendering of a correct pinpoint citation based on the text that was selected. Finally, the program successfully outputs the desired quotations/citations into the Word document. However, in performing initial user testing of the GUI/engine combination, we noted the following weaknesses:

a) Speed

It currently takes about 3 minutes to perform a query against the full SCOTUS corpus. In our view, this is an inordinate length of time for the user to wait for results after activating the tool, especially since the current implementation of the prototype blocks execution of Microsoft Word while the query is being performed (a cancel button is available, however.) Further, if the prototype were to be expanded to include more jurisdictions, the query time would also expand.

b) Relevance of Query Results

The recommendation engine seems to perform poorly on short queries, returning very noisy results. When coupled with the GUI, which by default only uses the current paragraph as a query, this causes the prototype to return noisy results almost every time it is used in a live test situation. Even when larger blocks of text were selected and tested, none of the cases returned seemed relevant to the contents of the query text on closer inspection.

c) Other Features

There were a few planned features that remain unimplemented. The planned "Settings" panel was not completed - currently the settings of the prototype remain on programmatically-set defaults, and cannot be customized by the user. The "preview tooltip" feature, which was intended to show the most relevant paragraphs of the case as a tooltip to

aid in selecting results to cite, has been partially implemented - the GUI is capable of displaying paragraphs of the cases as tooltips, but the recommendation engine is not correctly calculating the most relevant paragraphs to display. Finally, when outputting the citations to the Word document, the system is currently only able to output inline citations, without any formatting such as italics for the case name.

Directions for Future Work

1. Improving the Recommendation Engine

Further work is currently being performed to improve the recommendation engine (see the copy of the submitted paper above for a more detailed description of methods to be used to improve the recommendation engine.) Using a model with a greater number of topics will likely improve results, based on the indications from the initial model evaluation process. Aside from the automated testing process described in the submitted paper, further manual testing with users who have domain-specific knowledge will greatly aid in improving the recommendation engine. The highest priority will be to filter out noisy results, such as certiorari decisions, from the recommendations. A technique for handling shorter queries (<250 words) will also need to be developed. Additionally, an appropriate balance between topic relevance and citation "popularity" will need to be struck. There are multiple other parameters that can be tuned, such as which "similarity" measure to be used. Finally, once the system is returning good results on the SCOTUS data set, we will need to expand the prototype to include other jurisdictions, continue user testing with the expanded data set, and modify the algorithm according to user feedback on case relevance.

2. Improving the User Interface

To the greatest extent possible, the GUI was designed to be extensible, so that future feature expansion could be accommodated. For example, the display is completely customizable, so that not only cases might be displayed in the future, but also statutes, academic journal articles, even affidavits or other legal documents. The aim would be to have the tool work in conjunction with a law firm's knowledge management system.

The highest priority for improvement, however, is speed and memory optimization. Currently the entire SQL database is loaded into memory each time the ThinkCite program is run. This is the longest-running operation in the sequence. We will need to engineer a way to perform the similarity calculations on the much more memory and CPU-efficient

SQL server, so that only the most relevant results need to be returned to the ThinkCite application and loaded.

Once these initial important changes are made to the GUI and the recommendation engine, we will need to perform more extensive user testing with legal researchers in real-life working scenarios, in order to get feedback on how the tool performs during such tasks. Once the prototype has been improved to a sufficient level, we would then perform a side-by-side test, comparing the user experience of traditional legal research and writing with current tools, and the user experience of working with ThinkCite, both on a subjective level, and using objective measurements such as speed to accomplish research and writing tasks.

Other areas of potential improvement include implementing the planned features discussed in the section above. Finally, after the tool has been deemed ready for distribution, we would need to test its cross-platform capability, and test its deployment in a real situation such as a law practice.

Acknowledgments

We gratefully acknowledge the support of our project advisor, Prof. Deirdre Mulligan. We also wish to thank Prof. Brian Carver for his feedback throughout the process, and the Free Law Project for its generous provision of the Supreme Court case data. This resource was invaluable in building the prototype and conducting our analysis for the submitted paper. We also acknowledge the UC Berkeley DLab, its director Justin McCrary, and the Computational Text Analysis Working Group led by Nicholas Adams and Brooks Ambrose, for their excellent support and assistance. Finally, we thank the UC Berkeley School of Information and its Computing & Information Services Unit for their technical and application support, access, and hosting.