Corporate Veil Project

Douglas C. Barnard

3/10/2019

1 Overview

1.1 Personal capstone project

In this assignment for the HarvardX PH125.9x online data science course, students were instructed to apply machine learning techniques that go beyond standard linear regression to solve a problem of their choice. Specifically, students were instructed to submit a report that documents their analysis and presents their findings, with supporting statistics and figures, uploaded as both a PDF document and an Rmd file. Students were also instructed to submit their R script file.

As described below, my capstone project explores how natural language processing (NLP) machine-learning techniques can be applied in a typical legal research setting.

Important Note: My R code (which is included in the Rmd file and the R script file) uses the doMC package to enable parallel processing. With parallel processing enabled, the R code executes in ~30 minutes on an iMac Pro (2017) with CPU 3.2 GHz Intel Xeon W, RAM 32 GB 2666 MHz DDR4, and GPU Radeon Pro Vega 56 8176 MB. The R code will require more time to execute without parallel processing, and may exceed memory limits on a computer with less than 32 GB of RAM. The doMC package will only run on Macs and other Unix-type machines with multiple processors, multiple cores, or both. The R code includes a similar important note at the outset, referring to certain lines of code that must be commented out if the script will not be run on a Mac.

1.2 Background

Entrepreneurs who wish to start a new business can limit their personal liability by forming a corporation or limited liability company (often referred to as a business entity) to conduct the business. That way, if their business somehow harms other people and leads to litigation, only the portion of their personal assets that they have committed to the business (as opposed to their homes and the rest of their life savings) will be at risk. Essentially, society has decided to encourage entrepreneurship and the creation of new businesses by limiting what would otherwise be the legal rights of other people – who might be harmed by the new business – to sue the entrepreneur personally.

In addition to these benefits for individual entrepreneurs – who can limit their risk of personal liability by forming a business entity – an already existing business entity can further limit its own liability by forming another entity in turn. For example, a parent entity can form a subsidiary entity to conduct a particularly risky portion of the parent entity's overall business activities.

Society has always been troubled by the possibility that this encouragement of entrepreneurship – through the limitation of personal liability – could lead to an undesired proliferation of overly risky business activities. If this undesired result were to occur, there would be an uneconomic increase in the risk of harm to other people, and worse yet the other people who have been harmed as a result of these overly risky business activities may be left with no ability to recover from the people who harmed them. As a result, judges sometimes decide to disregard the separate identities of the owner and the entity, and permit other people who have been harmed to sue not only the entity that harmed them but also the owner of the entity personally.

In most such cases, however, judges tend to respect the separate identities of the owner and the entity.

For over 100 years, lawyers and law school professors have studied such cases, seeking (i) to identify the factors that judges consider in deciding whether to respect or disregard a limited liability entity structure and (ii) to construct a theoretical framework incorporating these relevant factors for use in predicting how a judge will likely rule in a particular case yet to be decided.

Unfortunately, no clear pattern has emerged in these cases, in part because the judges have tended to explain their decisions by using conclusory language that is synonymous with the outcome (e.g., discussing whether or not the entity was the "alter-ego" of the owner) or by referring to seemingly irrelevant facts related to the observance of corporate formalities (e.g., discussing whether or not the owner and the entity had held shareholder and board meetings, had kept detailed minutes of those meetings, and had maintained separate accounting records and checking accounts). Moreover, in those cases where judges have focused on the more relevant question of whether the owner had put "enough" of the owner's assets in the entity (measured against the risk of harm to other people inherent in its business activities), the judicial inquiry is somewhat at odds with the societal promise of limited liability to entrepreneurs (particularly when the adequacy of the entity's capitalization is evaluated with the benefit of hindsight after an unfortunate incident).

There was something of a breakthrough in this area in 2014, when two professors – Jonathan Macey at Yale Law School and Joshua Mitts at Columbia Law School – published an academic paper applying natural language (NLP) machine learning techniques in an attempt to find useful patterns in what were then ~10,000 potentially relevant cases. Their paper, entitled Finding Order in the Morass – The Three Real Justifications for Piercing the Corporate Veil, is available on the Cornell Law Review website at https://scholarship.law.cornell.edu/cgi/viewcontent.cgi?article=4647&context=clr. As indicated in the title of the paper, cases in which judges decide whether to respect or disregard the limited liability entity structure are often referred to as "veil-piercing" cases (including by judges in the text of their published opinions).

In their paper, professors Macey and Mitts announced that they had developed a theoretical framework — supported by the results of their machine-learning analysis – suggesting that judges tend to disregard the limited liability entity structure only when necessary (i) in order to achieve the purpose of a specific statute or regulation (e.g., protecting the environment or workers); (ii) in order to prevent owners from borrowing money at the entity level by misleading lenders into thinking incorrectly that the owner would be personally responsible for the entity's debt; and (iii) in order to prevent owners from removing assets from an entity in anticipation of the entity's imminent bankruptcy, thereby leaving the entity with insufficient assets to repay its creditors.

Professors Macey and Mitts also announced their finding that judges – who frequently discuss the observance of corporate formalities and the adequacy of the entity's capitalization at length in their opinions – actually pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

Upon reading the professors' published article, I found their conclusions surprising and – at least with respect to the dichotomy they reported between what the judges discuss at length in their opinions and what they actually consider in making their decisions – somewhat improbable. Therefore, I decided to replicate their methodology in this project and see whether I could reproduce their published results.

As discussed below, there are currently ~12,000 potentially relevant cases in the United States courts in which it appears (based on a Google-type search) that the judge may have decided whether or not to respect the limited liability entity structure.

My project replicates the professors' analysis, using the NLP machine learning techniques they described in their published article to examine a subset of these ~12,000 potentially relevant cases, i.e., just those cases that involved *bodily injuries and fatalities* (as opposed to business disputes involving contracts, leases, licenses, and loans). Cases involving bodily injuries and fatalities are commonly referred to as *"tort"* cases.

I chose to focus on tort cases for two reasons. First, cases involving bodily injuries and fatalities are in some ways the most interesting, since they present the greatest tension between society's desire to encourage entrepreneurship – through the limitation of personal liability – and society's concern that this limitation of personal liability could lead to an undesired proliferation of overly risky business activities if left unchecked. And second, as described below, the professors' methodology involves the reading and hand coding of a large sample of potentially relevant cases for use in training predictive algorithms, and I wished to limit the time I would need to invest in this task.

Specifically, I used machine-learning techniques to filter the full set of ~12,000 potentially relevant cases and focus on just a subset of ~500 tort cases involving bodily injuries and fatalities. For example, the people who were injured or killed in these tort cases may have been employees who worked in the entity's plants and warehouses, or members of the public who came into contact with the entity's products or business activities in their daily lives. In addition to a number of routine tort cases in this area, involving, e.g., workplace incidents, medical malpractice, nursing home incidents, and automobile accidents, there have also been several very large cases (e.g., involving asbestos, tobacco, pharmaceuticals, medical implants, and chemicals) in which many people were injured or killed.

In each of these tort cases, the people who had been injured or killed sought to recover from the owner of the entity (as well as from the entity itself), since the entity itself was not in a position to compensate fully the people who had been harmed. (In the case of fatalities, the spouse or an adult child of the deceased will often be the party suing the entity and its owner in the tort case.)

With the tort cases identified, I then used machine-learning techniques in an attempt to determine whether or not the professors' theoretical framework comports with the judges' actual decisions in these cases involving bodily injuries and fatalities. In other words, did the judges in these tort cases tend to respect the limited liability structure unless there was an applicable statute or regulation (the professors' first category) or the entity was in a formal bankruptcy proceeding (the professors' third category), or did the judges instead tend to make their decisions in these cases without regard to the presence or absence of an applicable statute or regulation and without regard to the formal bankruptcy status of the entity? (Note that the professors' second category – preventing fraudulent borrowing – is irrelevant in the context of bodily injuries and fatalities.)

I also used machine-learning techniques in an attempt to determine whether or not I could replicate the professors' finding that judges – who frequently discuss the observance of corporate formalities and the adequacy of the entity's capitalization at length in their opinions – actually pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

1.3 Methodology

Although professors Macey and Mitts described their machine-learning methodology in detail in their academic paper, they have chosen not to make their code available to other researchers who might wish to replicate or build upon their findings. Therefore, I used the professors' description of their methodology from the published paper and the techniques I have learned in HarvardX PH125x in order to perform my own analysis.

The professors' methodology included reading and hand-coding a random sample of ~1,000 judicial opinions (representing 10% of what was then the full set of ~10,000 potentially relevant cases) for algorithm training purposes. Unfortunately, the professors have chosen not to make this training set available to other researchers. As a result, I and other researchers who wish to replicate or build upon their findings need to read and hand code our own training set of judicial opinions.

As noted above, there are now ~12,000 potentially relevant cases in the full set, i.e., cases that appear to include a judge's decision whether to respect or disregard the limited liability structure. My project focuses on just a subset of these cases, specifically the ~500 potentially relevant *tort* cases that also appear to involve bodily injuries and fatalities. My initial results revealed that a model trained on a random sample of only 10% or 20% of these ~500 potentially relevant tort cases would be somewhat inaccurate. Therefore, I ultimately had to spend three full days reading and hand coding a random sample of ~250 cases (representing 50% of the potentially relevant set) for algorithm training purposes.

As described below in the Analysis section of this report, my overall approach to completing this project proceeded as follows:

1. Use the free CourtListener API and its Google-type search capabilities to identify and download the full text of ~12,000 potentially relevant cases

2. Use the quanteda package and its various natural language processing (NLP) functions in order to perform the following operations:

   - Create a “corpus” containing the full text of the judges’ opinions for all of the ~12,000 downloaded cases.
   - Preprocess the downloaded opinions by (i) converting all of the text to lowercase, (ii) removing numbers, punctuation, and symbols, (iii) removing common “stop words” that lack substantive meaning (using the standard list of stop words from the quanteda package, as well as specialized legal stop words in a list that professors Macey and Mitts have made publicly available), (iv) removing any short words with less than four characters, (v) stripping extraneous white space from the document, and (vi) eliminating word endings such as “ing,” “ed,” and “es” so as to leave only the core “stems” of the words.
   - “Tokenize” the preprocessed text in order to form “bigrams,” i.e., phrases consisting of two consecutive words, since bigrams tend to have greater information content – and therefore greater predictive power – than single words tend to have when considered in isolation.
   - Eliminate bigrams that are found in less than 1% of the ~12,000 opinions and are therefore unlikely to have general predictive power with respect to the full set.
   - Convert the corpus into a “document frequency matrix” (referred to in this report as the full set dfm), recording for each opinion the “counts” (i.e., the number of occurrences) for each of the bigrams (or “features”) that remain in the full set.
   - Based on a manual examination of the bigram features in the full set dfm, identify those features that would suggest a case involves bodily injuries or fatalities, and use these features as a filter in order to create a subset dfm (referred to in this report as the *tort* set dfm) consisting of just the cases that are potentially relevant for the purposes of this project.
   - Based on a similar manual examination of the bigram features, identify those features that would suggest a particular case is at a preliminary procedural phase, and use these features as a filter in order to remove from the tort set dfm those cases that – given their preliminary procedural status – are unlikely to include a judge’s substantive decision whether or not to respect the limited liability structure.

3. Read the full text of ~250 randomly selected opinions (50% of the ~500 potentially relevant opinions in the tort set dfm), and hand code each opinion as to the relevance of the opinion for the purposes of this project and – for those opinions that are found to be relevant – as to the outcome in the case (i.e., the judge’s decision whether or not to respect the limited liability structure).
4. Create two dfms (referred to in this report as the hand-coded set dfm and the hand-coded *relevant* dfm) from the hand-coded data for use as training sets in constructing and validating the machine-learning algorithms described below.
5. Construct *random* relevance and outcome classification models for benchmarking purposes.
6. Construct several *relevance* and *outcome* classification models, trained with 90/10 cross validation repeated 100 times or – for two models that were not accessible via the caret package – trained using 90/10 bootstrap validation repeated 100 times, including:

   - naive Bayes relevance and outcome classification models
   - extreme gradient boosting relevance and outcome classification models
   - random forest relevance and outcome classification models
   - support vector machine relevance and outcome classification models
   - generalized boosted relevance and outcome classification models

7. Based on the cross-validated (or bootstrap-validated) training results, evaluate the accuracy of each of these models (using a confusion matrix that shows the true positives, false positives, false negatives, and true negatives as well as the accuracy, recall, precision, and harmonic average of recall and precision – or F1 – derived therefrom) in predicting the *relevance* of unread cases and, for the unread cases that are predicted to be relevant, predicting the judicial *outcome.*

8. Select the best performing *relevance* classification model, i.e., the model with the highest harmonic average of recall and precision (or F1) in predicting relevant cases, given that the objective in predicting the relevance of unread cases is to identify as many of the actually relevant cases as possible for further analysis while minimizing the number of false positive predictions.

9. Select the best performing *outcome* classification model, i.e., the model with the highest harmonic average of recall and precision (or F1) in predicting judicial outcomes for relevant uread cases, given that the objective in predicting the outcomes of relevant unread cases is to predict both positive and negative outcomes for the relevant unread cases as accurately as possible for further analysis.

10. Use the best-performing relevance classification model to predict the relevance for each of the ~250 potentially relevant opinions in the tort set dfm that have not been read and hand coded, and then use the best-performing outcome classification model to predict the judicial outcomes for the unread cases that are predicted to be relevant.

11. Create a dfm (referred to in this report as the *relevant* set dfm) containing the cases that have been hand coded or machine coded as relevant, as well as the hand-coded or machine-coded outcomes in each instance, but set this dfm aside (for the reasons discussed below in the Analysis section of this report) and use the *hand-coded* relevant dfm instead for the remaining steps 12 through 19 below.

12. Extract the outcome-predictive weights assigned to each of the bigram features considered in the best-performing outcome classification model in order to identify and rank in descending order the bigrams that have the greatest predictive power with respect to judicial outcomes.

13. Manually examine these outcome-predictive bigram features, identify those highly predictive features that are conclusory (e.g., "pierce_veil") or uninformative (e.g., "citat_omit") as opposed to potentially illuminating (e.g., "insur_coverag"), and assign a new outcome-predictive weight of zero to the conclusory and uninformative features so they sink to the bottom in the ranking of outcome-predictive features.

14. Manually examine the outcome-predictive weights of the re-ranked bigram features in an attempt to confirm the professors' finding that judges – who frequently discuss the observance of corporate formalities and the adequacy of the entity's capitalization at length in their opinions – actually pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

15. Use the topicmodels package to create a three-topic unsupervised learning model for the cases in the hand-coded relevant dfm, identify the bigram features that are associated with each of these three topics, and calculate their respective *topic-model* weights in each instance.

16. For each of these topic-predictive bigram features, multiply their respective *topic-model* weights by their respective *outcome-predictive* weights (as determined in steps 12 and 13 above) in order to identify and rank in descending order the bigram features for each of the three unsupervised learning model topics that are not only associated with that *topic* but also highly predictive of judicial *outcomes.*

17. Manually examine the ranked bigram features for each of these three topics, looking for categories of tort cases that should be *considered separately* in attempting to determine how judges decide whether to respect or disregard the limited liability structure in these particular types of cases.

18. Having identified one such topical category, i.e., cases that involve workplace injuries and fatalities (and therefore an applicable workers compensation statute), divide the hand-coded relevant cases into two subsets based on the presence or absence of bigram features associated with this topic, and then examine and compare the outcomes for the cases that are in this category versus the outcomes for cases that are not in this category.

19. Use this information to determine whether the judges' decisions in these cases involving workplace injuries and fatalities comport with the first element of the professors' theoretical framework, i.e., whether the judges were *more likely to disregard* the limited liability structure in this category of cases since they involve an applicable workers compensation statute.

1.4 Results

As described below in the Results section of my report, I was able to replicate the essential results of the professors' machine-learning analysis of the full set of relevant cases through my machine-learning analysis of the subset of relevant *tort* cases. I also reach most, but not all, of the same conclusions the professors did in examining the results of our respective machine-learning analyses.

1.4.1 The Professors' Theoretical Framework

In their published paper, professors Macey and Mitts announced that they had developed a theoretical framework – supported by the results of their machine-learning analysis – suggesting that judges tend to disregard the limited liability structure only when necessary (i) in order to achieve the purpose of a specific statute or regulation (e.g., protecting the environment or workers) . . . .

- My unsupervised learning "topic-model" analysis found that tort cases involving workers compensation statutes do in fact appear to constitute a *separate category* for purposes of predicting whether judges will respect or disregard the limited liability entity structure (which is consistent with the professors' findings).
- My analysis of the hand-coded relevant tort cases also found that judges are in fact *more likely to disregard* the limited liability entity structure in cases involving workers compensation statutes (which is consistent with the professors' theoretical framework).
- I do, however, have one concern regarding the professors' *methodology* and the *conclusions* they drew from the results of their machine-learning analysis, as discussed below in the Results section of this report.

1.4.2 Importance of the Factors Judges Discuss in Their Opinions

In their published paper, professors Macey and Mitts also reported finding relatively low *outcome-predictive weights* for the bigram features associated with the observance of corporate formalities and the adequacy of the entity's capitalization, and concluded from this finding of low outcome-predictive weights for these bigram features that judges – who frequently discuss these factors at length in their published opinions – actually pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

- Following the professors' methodology, my machine-learning analysis of the hand-coded relevant tort cases also found relatively low *outcome-predictive weights* for the bigram features associated with the observance of corporate formalities and the adequacy of the entity's capitalization.
- I do, however, have one concern regarding the professors' *methodology* and the *conclusions* they drew from the results of their machine-learning analysis, as discussed below in the Results section of this report.
- Therefore, I am unable to conclude on the basis of the professors' description of their methodology and their reported results that judges *actually* pay relatively little attention to the very factors they discuss at length in their published opinions in deciding whether to respect or disregard the limited liability entity structure.

- In order to reach this somewhat improbable conclusion, it would be necessary to examine the professors' machine-learning code and ideally their training set of hand-coded cases (neither of which they have chosen to make available), replicate their results, and confirm that the relatively low outcome-predictive weights that the professors report for the very factors that judges discuss at length in their opinions are *real* rather than mere artifacts of the professors' methodology.

1.5 Conclusions

The Conclusions section of this report briefly discusses the use of machine-learning techniques in legal research generally, and then describes some aspects of this capstone project that I have identified for future exploration and analysis.

2 Analysis

This section of the report describes in more detail each of the steps listed above in the Overview section. The Rmd version of this report includes the relevant R code for each step, but the code will be hidden in the version of this report knitted to PDF format. The Rscript file submitted with this report contains just my R code, without the additional commentary and analysis contained in this report.

2.1 Download required packages and enable parallel processing

In this preparatory step, my R code loads the packages that the subsequent steps will require (downloading and installing them as necessary).

My R code uses the doMC package in order to enable parallel processing on a Mac machine. Please see the important note in this regard near the beginning of my R code if the script will be executed on a Windows machine.

2.2 Download full text of ~12,000 judicial opinions

In this step, my R code uses the free CourtListener API and its Google-type search capabilities to identify and download the full text of ~12,000 cases that potentially involve judges deciding whether to respect or disregard a limited liability entity structure. The functionality and use of the CourtListener API (and the steps required in order to obtain a personal userid and password) are described on the CourtListener website at https://www.courtlistener.com/api/.

Essentially, my Google-type search phrase uses "wild-cards" to look for opinions on the CourtListener API that contain "pierc*" within three words of "veil" OR that contain "disregard*" within five words of "corpor*". This search phrase takes advantage of the fact that cases involving judges deciding whether or not to respect a limited liability entity structure are commonly referred to as "veil-piercing" cases (including by judges in their published opinions).

Since I am only interested for the purposes of this project in the subset of these ~12,000 cases that involve *bodily injuries and fatalities*, I attempted to restrict my search results in this regard using the Boolean "AND" feature of the CourtListener API. However, I found this feature of the CourtListener API to be unreliable (in fact the number of cases retrieved actually increased when I attempted to apply this restriction). Therefore, I delayed implementing this restriction until the next step in my R code as described below.

I also found that the CourtListener API was dropping every 21st potentially relevant case (as compared to cases that appear on the CourtListener home page when using the same Google-type search phrase). I corresponded with the person maintaining the CourtListener website and API, and he discovered and corrected the "bug" in the API.

Since the CourtListener API “throttles” the free downloading of judicial opinions, it can take several hours to download the full text of these ~12,000 cases. Furthermore, new cases are added to the CourtListener API every day, and I want to “freeze” the full data set so that my results will be reproducible. Therefore, I have disabled this portion of the R code in my Rmd file with an “eval=FALSE” tag (and removed my personal userid and password) so that it will not execute when the Rmd file is knitted to a PDF file. I have similarly commented out this portion of the R code in my R script file.

Instead, I have saved the full set of data I downloaded from the CourtListener API in a file named “data/courtlistener-data.RData” on my public GitHub site, available at https://github.com/dcbarnard/corporate-veil.

Since this portion of the R code in my Rmd file has been disabled, my Rmd file also includes R code that will load the saved CourtListener API data.

The saved data includes the full text of 12398 potentially relevant opinions, along with associated metadata (such as the CourtListener URL) for each downloaded opinion.

2.3 Create document features matrices for the downloaded opinions

In this step, my R code uses the quanteda package and its various natural language processing (NLP) functions in order to perform the following operations:

- Create a “corpus” containing the full text of the judges’ opinions for all of the 12398 downloaded cases.
- Preprocess the downloaded opinions by (i) converting all of the text to lowercase, (ii) removing numbers, punctuation, and symbols, (iii) removing common “stop words” that lack substantive meaning (using the standard list of stop words from the quanteda package, as well as specialized legal stop words in a list that professors Macey and Mitts have made publicly available), (iv) removing any short words with less than four characters, (v) stripping extraneous white space from the document, and (vi) eliminating word endings such as “ing,” “ed,” and “es” so as to leave only the core “stems” of the words.
- “Tokenize” the preprocessed text in order to form “bigrams,” i.e., phrases consisting of two consecutive words, since bigrams tend to have greater information content – and therefore greater predictive power – than single words tend to have when considered in isolation.
- Eliminate bigrams that are found in less than 1% of the 12398 opinions and are therefore unlikely to have general predictive power with respect to the full set.
- Convert the corpus into a “document frequency matrix” (referred to in this report as the full set dfm), recording for each opinion the “counts” (i.e., the number of occurrences) for each of the bigrams (or “features”) that remain in the full set.
- Based on a manual examination of the bigram features in the full set dfm, identify those features that would suggest a case involves *bodily injuries or fatalities*, and use these features as a filter in order to create a subset dfm (referred to in this report as the *tort* set dfm) consisting of just the cases that are potentially relevant for the purposes of this project.
- Based on a similar manual examination of the bigram features, identify those features that would suggest a particular case is at a *preliminary procedural phase*, and use these features as a filter in order to remove from the tort set dfm those cases that – given their preliminary procedural status – are unlikely to include a judge’s substantive decision whether or not to respect the limited liability structure.

After performing these operations, the full set dfm contains 12398 cases, the tort set dfm contains 474 cases, and both the full set dfm and the tort set dfm have the same 2307 bigram features.

The full set dfm and tort set dfm are *dense* matrices, as shown in the following table depicting the upper left corner of the tort-set dfm:

document	argu_decid	argu_caus	present_whether	within_mean	manag_compani
388125	1	0	0	0	0
482542	0	0	0	2	0
2173013	0	0	0	0	0
503793	1	0	0	0	0
2403597	0	0	0	1	0

The first column of each dfm created in this step contains the CourtListener document id for each of the 12398 (or 474) individual opinions in the dfm, and the remaining 2307 columns contain the counts (the number of occurrences) for each of the 2307 bigram features for each such opinion.

The following table shows the top 10 bigram features for the full set dfm and for the tort set dfm, ranked in descending order of their frequencies of occurrence:

Full Set	Tort Set
corpor_veil	corpor_veil
pierc_corpor	pierc_corpor
corpor_entiti	worker_compens
caus_action	caus_action
person_jurisdict	corpor_entiti
disregard_corpor	punit_damag
corpor_form	parent_corpor
breach_contract	disregard_corpor
veil_pierc	corpor_form
parent_corpor	person_injuri

Since the tort set dfm is limited to opinions involving *bodily injuries and fatalities*, certain bigram features such as “worker_compens,” “punit_damag,” and “person_injuri” that are associated with such cases have risen to the top 10 in the tort set dfm ranking.

2.4 Create hand-coded dataset for random 50% sample of opinions

In this step, my R code randomly selects 237 opinions (50% of the 474 opinions in the tort set dfm) without replacement, exports a spreadsheet with URLs for the selected opinions, and then pauses so that I can read the full text of each of these opinions on the CourtListener website.

- I used this spreadsheet to record my findings as to the relevance of each opinion I read and – for those opinions that were relevant – as to the outcome in the case (i.e., the judge’s decision whether or not to respect the limited liability structure).
- Once I had completed my hand coding of the 237 randomly selected opinions, I saved the filled-in spreadsheet and returned to my R code for further processing of the hand-coded data.

My R code then reads the hand-coded data from the fill-in spreadsheet, and stores it in a data frame for further analyses in the steps below.

Since it is unlikely that readers of this report will want to replicate my three-full-day effort to read and hand code each of these 237 opinions, I have disabled this portion of the R code in my Rmd file (with an “eval=FALSE” tag) so that it will not execute when the Rmd file is knitted to a PDF file. I have similarly commented out this portion of the R code in my R script file.

Instead, I have saved my hand-coded data in a file named “data/hand-coded-data.RData” on my public GitHub site, available at https://github.com/dcbarnard/corporate-veil.

Since this portion of the R code in my Rmd file has been disabled, my Rmd file also includes R code that will load the saved hand-coded data.

2.5 Create document features matrices from the hand-coded data

In this step, my R code creates two dfms (referred to in this report as the hand-coded set dfm and the hand-coded *relevant* dfm) from the hand-coded data for use as training sets in constructing and validating the machine-learning algorithms described below.

The first column of each dfm created in this step contains the CourtListener document id for each of the individual opinions in the dfm, and the remaining columns contain the counts (the number of occurrences) for each of the bigram features with respect to each such opinion. In addition, each dfm created in this step contains *document variables* recording the hand-coded relevance of each opinion and, for each relevant opinion, the hand-coded outcome in that case.

In examining the hand-coding data for the random sample of 237 potentially relevant opinions, it turns out that only 42 percent of the *potentially* relevant hand-coded opinions were *actually* relevant for the purposes of this project, and that the judge disregarded the limited liability entity structure in only 33 percent of the opinions that were found to be relevant.

Given these prevalences, it appears that the training data is somewhat unbalanced, particularly with respect to case outcomes.

In addition, although there are 237 relevant and irrelevant hand-coded opinions available for purposes of training the *relevance* classification models, there are only 99 *relevant* hand-coded opinions available for purposes of training the *outcome* classification models in the steps below.

2.6 Construct random classification models for benchmarking purposes

For benchmarking purposes, my R code first constructs *random* relevance and outcome classification models.

Specifically, the random *relevance* classification model makes random predictions as to the relevance of the opinions in the hand-coded set dfm, and the random *outcome* classification model makes random predictions as to the outcomes for the *relevant* opinions in the hand-coded relevant dfm. These random predictions are compared to the hand-coded classifications, and the numbers of true positives, false positives, false negatives, and true negatives (as well as the accuracy, recall, precision, and harmonic average of recall and precision – or F1 – derived therefrom) are recorded. This process is repeated 10,000 times, and the final performance metrics are calculated based on the average results for all 10,000 iterations.

In making their random predictions:

- the random *relevance* classification model takes into account the 42 percent prevalence for opinions in the hand-coded set dfm that were *relevant*, and
- the random *outcome* classification model takes into account the 33 percent prevalence for relevant opinions in the hand-coded relevant dfm in which the judge *disregarded* the limited liability entity structure.

The following table summarizes the performance metrics for the random *relevance* classification model in predicting the relevance of the hand-coded opinions on average over the course of all 10,000 iterations:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.174	0.243	0.243	0.339	0.513	0.418	0.418	0.418

Similarly, the following table summarizes the performance metrics for the random *outcome* classification model in predicting the outcomes of the hand-coded relevant opinions on average over the course of all 10,000 iterations:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.111	0.222	0.222	0.444	0.556	0.334	0.334	0.334

Note that the recall, precision, and F1 of 42 percent for the random *relevance* classification model is approximately equal to the prevalence of 42 percent for opinions in the hand-coded set dfm that were *relevant*, and that the recall, precision, and F1 of 33 percent for the random *outcome* classification model is approximately equal to the prevalence of 33 percent for relevant opinions in the hand-coded relevant dfm in which the limited liability entity structure was *disregarded*, which is what one would expect in each case if the prevalence-weighted classification predictions were truly random.

Since I will be comparing the performance of my five classification models in predicting (i) the relevance of unread opinions and (ii) the outcomes of relevant unread opinions, I will be adding models to these two tables in the steps below.

2.7 Construct and evaluate several classification models

In this step, my R code constructs the following *relevance* and *outcome* classification models:

- naive Bayes relevance and outcome classification models
- extreme gradient boosting relevance and outcome classification models
- random forest relevance and outcome classification models
- support vector machine relevance and outcome classification models
- generalized boosted relevance and outcome classification models.

In each case, the relevance classification models were trained using the hand-coded set dfm with 90/10 cross validation repeated 100 times (or, in the case of two models that were not accessible via the caret package, with 90/10 bootstrap validation repeated 100 times). The outcome classification models were similarly trained, except this time using the hand-coded *relevant* dfm instead of the full hand-coded set dfm since the judicial outcomes are of interest only with respect to the relevant cases.

During each of the 100 cross-validation (or bootstrap-validation) passes, the particular classification model was *trained* on a randomly selected 90% sample of the hand-coded opinions and then *tested* by comparing the classification model *predictions* with the actual *classifications* for the remaining 10% of the hand-coded opinions which had been held out for testing purposes.

Based on the cross-validated (or bootstrap-validated) training results, my R code then evaluates the accuracy of each of these models (using a confusion matrix that shows the true positives, false positives, false negatives, and true negatives, as well as the accuracy, recall, precision, and F1 derived therefrom, measured on average over the 100 cross-validation or bootstrap-validation passes) in predicting the relevance of unread opinions and, for the unread opinions that are predicted to be relevant, predicting the judicial outcome.

I included the naive Bayes classification model because it is the one that professors Macey and Mitts used for this purpose in their published paper. I selected and included the four other classification models after reviewing an academic paper by four computer science professors, Manuel Fernandez-Delgado, Eva Cernadas, Senen Barro, and Dinani Amorim, entitled Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?, available at http://jmlr.org/papers/volume15/delgado14a/delgado14a.pdf.

2.7.1 Naive Bayes classification models

For this step in my analysis, I used the naive Bayes classification model included in the quanteda natural language processing (NLP) package, rather than the naive Bayes classification model included in the caret package.

According to a post on Stack Overflow by Kenneth Benoit, the author of the quanteda package, caret (which uses naive_bayes from the naivebayes package) assumes a Gaussian distribution, whereas quanteda::textmodel_nb() is based on a more text-appropriate multinomial distribution. Professor Benoit also points out in his post that the caret version is "mind-numbingly" slow, since it works on *dense* matrices, as compared to the quanteda version, which works on the *sparse* dfm. Professor Benoit's Stack Overflow post is available at https://stackoverflow.com/questions/54427001/naive-bayes-in-quanteda-vs-caret-wildly-different-results/54431055#54431055.

For this project, I tried both the caret and the quanteda versions, and confirmed that the quanteda version is indeed much faster than the caret version.

Unlike the caret version, the quanteda version of a naive Bayes classification model does not include any cross-validation or bootstrap-validation capabilities. Therefore, my R code includes my own 90/10 bootstrap-validation code, which repeats 100 times.

Also, unlike the caret version, the quanteda version of the naive Bayes classification model does not include any parameters that may be varied for tuning purposes.

The following table summarizes the performance metrics for the naive Bayes classification model in predicting the *relevance* of unread opinions on average over the course of my 90/10 bootstrap validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.174	0.243	0.243	0.339	0.513	0.418	0.418	0.418
naive Bayes	0.328	0.188	0.082	0.402	0.730	0.801	0.636	0.709

Similarly, the following table summarizes the performance metrics for the naive Bayes classification model in predicting the *outcomes* of relevant unread opinions on average over the course of my 90/10 bootstrap validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.111	0.222	0.222	0.444	0.556	0.334	0.334	0.334
naive Bayes	0.128	0.204	0.202	0.466	0.594	0.388	0.386	0.387

It appears from these performance metrics that the naive Bayes *relevance* classification model performs reasonably well in predicting the relevance of unread opinions (with an F1 of 71 percent), but that the naive Bayes *outcome* classification model performs rather poorly (with an F1 of only 39 percent) in predicting the outcomes for relevant unread opinions.

2.7.2 Extreme gradient boosting classification models

For this step in my analysis, I used the extreme gradient boosting classification model included in the xgboost package, rather than essentially the same model included in the caret package.

For some reason, the version of this model included in the caret package was much slower than the native version included in the xgboost package, and I was not able to rectify this very significant difference in processing times.

Although the native version of the extreme gradient boosting classification model includes a cross-validation function, this function can only be used for the purpose of selecting the optimal number of boosting rounds. Therefore, although my R code uses this cross-validation function for its intended purpose, my R code also includes my own 90/10 bootstrap-validation code, which repeats 100 times.

Since I did not use the caret version of this model, I was unable to use caret for model tuning purposes. Instead, I experimented with several combinations of model parameters, and then included the best combinations of parameters in my R code for both the relevance classifier and the outcome classifier.

The following table summarizes the performance metrics for the extreme gradient boosting classification model in predicting the *relevance* of unread opinions on average over the course of my 90/10 bootstrap validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.174	0.243	0.243	0.339	0.513	0.418	0.418	0.418
naive Bayes	0.328	0.188	0.082	0.402	0.730	0.801	0.636	0.709
extreme gradient boosting	0.252	0.111	0.158	0.479	0.732	0.616	0.695	0.653

Similarly, the following table summarizes the performance metrics for the extreme gradient boosting classification model in predicting the *outcomes* of relevant unread opinions on average over the course of my 90/10 bootstrap validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.111	0.222	0.222	0.444	0.556	0.334	0.334	0.334
naive Bayes	0.128	0.204	0.202	0.466	0.594	0.388	0.386	0.387
extreme gradient boosting	0.152	0.184	0.198	0.466	0.618	0.434	0.452	0.443

It appears from these performance metrics that the naive Bayes *relevance* classification model and the extreme gradient boosting *outcome* classification model have the highest F1 scores based on a comparison of the two algorithms evaluated thus far.

2.7.3 Random forest classification models

For this step in my analysis, I used the random forest classification model from the ranger package, as incorporated into the caret package.

I selected the random forest algorithm from the ranger package since, according to the package notes and the package vignette, it is a fast implementation of random forests, particularly suited for high-dimensional data. This sounds appropriate for my project, since my dataset is relatively high dimensional with 2307 bigram features serving as potential predictors.

Since the caret package includes a cross-validation function, my R code performs a 90/10 cross validation repeated 100 times.

In the course of this cross validation, my R code also performs an initial tuning of the model parameters for the random forest relevance and outcome classification models.

The following table summarizes the performance metrics for the random forest classification model in predicting the *relevance* of unread opinions on average over the course of my 90/10 cross validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.174	0.243	0.243	0.339	0.513	0.418	0.418	0.418

model	tp	fp	fn	tn	acc	rcl	prec	f1
naive Bayes	0.328	0.188	0.082	0.402	0.730	0.801	0.636	0.709
extreme gradient boosting	0.252	0.111	0.158	0.479	0.732	0.616	0.695	0.653
random forest	0.244	0.109	0.173	0.473	0.718	0.585	0.692	0.634

Similarly, the following table summarizes the performance metrics for the random forest classification model in predicting the *outcomes* of relevant unread opinions on average over the course of my 90/10 cross validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.111	0.222	0.222	0.444	0.556	0.334	0.334	0.334
naive Bayes	0.128	0.204	0.202	0.466	0.594	0.388	0.386	0.387
extreme gradient boosting	0.152	0.184	0.198	0.466	0.618	0.434	0.452	0.443
random forest	0.074	0.037	0.260	0.629	0.703	0.221	0.664	0.332

It appears from these performance metrics that the naive Bayes *relevance* classification model and the extreme gradient boosting *outcome* classification model have the highest F1 scores based on a comparison of the three algorithms evaluated thus far.

2.7.4 Support vector machine classification models

For this step in my analysis, I used the support vector machine classification model from the kernlab package, as incorporated into the caret package.

I selected the support vector machine algorithm from the kernlab package since, according to the package notes and package vignette, it works with *sparse* matrices and is therefore considerably more efficient in terms of memory usage and processing times.

Since the caret package includes a cross-validation function, my R code performs a 90/10 cross validation repeated 100 times.

In the course of this cross validation, my R code also performs an initial tuning of the model parameters for the support vector machine relevance and outcome classification models.

The following table summarizes the performance metrics for the support vector machine classification model in predicting the *relevance* of unread opinions on average over the course of my 90/10 cross validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.174	0.243	0.243	0.339	0.513	0.418	0.418	0.418
naive Bayes	0.328	0.188	0.082	0.402	0.730	0.801	0.636	0.709
extreme gradient boosting	0.252	0.111	0.158	0.479	0.732	0.616	0.695	0.653
random forest	0.244	0.109	0.173	0.473	0.718	0.585	0.692	0.634
support vector machine	0.284	0.155	0.134	0.427	0.711	0.680	0.647	0.663

Similarly, the following table summarizes the performance metrics for the support vector machine classification model in predicting the *outcomes* of relevant unread opinions on average over the course of my 90/10 cross validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.111	0.222	0.222	0.444	0.556	0.334	0.334	0.334

model	tp	fp	fn	tn	acc	rcl	prec	f1
naive Bayes	0.128	0.204	0.202	0.466	0.594	0.388	0.386	0.387
extreme gradient boosting	0.152	0.184	0.198	0.466	0.618	0.434	0.452	0.443
random forest	0.074	0.037	0.260	0.629	0.703	0.221	0.664	0.332
support vector machine	0.221	0.211	0.112	0.456	0.677	0.664	0.512	0.578

It appears from these performance metrics that the naive Bayes *relevance* classification model and the support vector machine *outcome* classification model have the highest F1 scores based on a comparison of the four algorithms evaluated thus far.

2.7.5 Generalized boosted machine classification models

For this step in my analysis, I used the generalized boosted machine classification model from the gbm package, as incorporated into the caret package.

I selected the generalized boosted machine algorithm from the gbm package after reviewing the package notes and package vignettes for several of the available generalized boosted machine algorithms incorporated into the caret package.

Since the caret package includes a cross-validation function, my R code performs a 90/10 cross validation repeated 100 times.

In the course of this cross validation, my R code also performs an initial tuning of the model parameters for the generalized boosted machine relevance and outcome classification models.

The following table summarizes the performance metrics for the generalized boosted machine classification model in predicting the *relevance* of unread opinions on average over the course of my 90/10 cross validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.174	0.243	0.243	0.339	0.513	0.418	0.418	0.418
naive Bayes	0.328	0.188	0.082	0.402	0.730	0.801	0.636	0.709
extreme gradient boosting	0.252	0.111	0.158	0.479	0.732	0.616	0.695	0.653
random forest	0.244	0.109	0.173	0.473	0.718	0.585	0.692	0.634
support vector machine	0.284	0.155	0.134	0.427	0.711	0.680	0.647	0.663
generalized boosted	0.253	0.096	0.165	0.486	0.740	0.606	0.726	0.660

Similarly, the following table summarizes the performance metrics for the generalized boosted machine classification model in predicting the *outcomes* of relevant unread opinions on average over the course of my 90/10 cross validation repeated 100 times:

model	tp	fp	fn	tn	acc	rcl	prec	f1
random benchmark	0.111	0.222	0.222	0.444	0.556	0.334	0.334	0.334
naive Bayes	0.128	0.204	0.202	0.466	0.594	0.388	0.386	0.387
extreme gradient boosting	0.152	0.184	0.198	0.466	0.618	0.434	0.452	0.443
random forest	0.074	0.037	0.260	0.629	0.703	0.221	0.664	0.332
support vector machine	0.221	0.211	0.112	0.456	0.677	0.664	0.512	0.578
generalized boosted	0.125	0.104	0.208	0.563	0.688	0.376	0.546	0.445

It appears from these performance metrics that the naive Bayes *relevance* classification model and the support vector machine *outcome* classification model have the highest F1 scores based on a comparison of the five algorithms I evaluated.

2.8 Select the best performing relevance classification model

Based on this comparative analysis of the five different *relevance* classification models, it appears that the *naive Bayes* model has the highest F1 in predicting relevant opinions. Since the objective in predicting the relevance of unread opinions is to identify as many of the actually relevant opinions as possible, while minimizing the number of false positive predictions, I have selected the naive Bayes model for this use in the further analyses described below.

2.9 Select the best performing outcome classification model

Based on this comparative analysis of the five different *outcome* classification models, it appears that the *support vector machine* model has the highest F1 in predicting judicial outcomes for relevant cases. Since the objective in predicting the outcomes of unread opinions is to predict both positive and negative outcomes for the relevant opinions as accurately as possible, I have selected the support vector machine model for this use in the further analyses described below.

2.10 Predict relevance and outcome for unread tort opinions

In this step, my R code uses the naive Bayes *relevance* classification model to predict the relevance for each of the 237 potentially relevant opinions in the tort set dfm that have *not* been read and hand coded, and then uses the support vector machine *outcome* classification model to predict the judicial outcomes for the unread opinions that are predicted to be relevant.

2.11 Create relevant set dfm with hand-coded and machine-coded opinions

In this step, my R code creates a dfm (referred to in this report as the *relevant* set dfm) containing the opinions that have been hand coded or machine coded as relevant, as well as the hand-coded or machine-coded outcomes in each instance.

2.11.1 The classification model predictions

If you assume that all of the 237 opinions I read were accurately hand-coded, then it is clear that the relevance and outcome classification model *predictions* for the 237 *unread* opinions are not nearly as accurate as my hand-coded classifications.

The bootstrap-validation performance metrics for the naive Bayes *relevance* classification model included the following true and false classification rates, where a *relevant* classification was defined as a *positve* classification for this purpose:

- True positive rate: 0.328
- False positive rate: 0.188
- False negative rate: 0.082
- True negative rate: 0.402

This suggests that approximately 36 percent (i.e., the false positive rate divided by the *sum* of the true positive rate and the false positive rate) of the *machine-coded* opinions in the *relevant* set dfm are actually *irrelevant*.

This inaccuracy in predicting the *relevance* of unread opinions also means that it is difficult to estimate the accuracy of the support vector machine *outcome* classification model in predicting outcomes for the machine-coded opinions in the relevant set dfm. Note that the cross-validation results for the support vector machine classification model only assess its accuracy in predicting outcomes for unread opinions that are assumed to have been *accurately* classified as relevant beforehand. Therefore, it is not clear what outcome the support vector machine outcome classification model would predict for an *irrelevant* opinion that has been incorrectly machine-coded as relevant, and in any event the predicted outcome for an irrelevant opinion would not be meaningful for the purposes of my further analyses in the steps below.

2.11.2 The hand-coded classifications

However, my experience in reading and *hand coding* 237 randomly selected opinions leads me to believe that my own classifications as to relevance and outcome were not 100% accurate either.

In many instances, it was actually difficult for me to decide upon the relevance and outcome of a particular opinion I had just read:

- In a number of the opinions I read, the judge ruled *against* the owner of an entity, but on the basis of a slightly different *theory* (e.g., by finding that the entity was the *agent* of the owner when it harmed another person, and therefore the owner was responsible for the harm its agent caused, or by finding that *both* the entity and the owner had *independently* harmed the other person). Sometimes in these opinions the judge claimed he or she had *respected* the limited liability entity structure in the process of ruling *against* the owner of an entity, and other times the judge did not address this specific question at all.
- In reading such opinions, I struggled in my classifications as to relevance (i.e., in deciding whether the case was *really* a relevant "veil-piercing" case) and as to outcome (i.e., in deciding whether the judge had *respected* or *disregarded* the limited liability entity structure in the process of ruling *against* the owner, particularly when the judge did not address this specific question in his or her opinion).
- As I read more and more opinions, my understanding of the relevant legal framework increased, and I sometimes felt compelled to go back and re-code opinions that I had already coded differently in order to reflect my increased understanding of the underlying legal framework.

I suspect that another person reading the same 237 opinions might classify some of them differently than I did as to relevance and outcome, and I am not even certain that I would classify all of the opinions I read the same way if I were to read them again several months from now when they are no longer fresh in my mind.

Moreover, it is likely that professors Macey and Mitts (and any law students working under their supervision) hand coded the ~1,000 opinions they read more accurately than I hand coded the 237 opinions that I read, particularly in light of their excellent legal minds and their focus on this particular legal issue in writing a paper for publication in a peer-reviewed academic journal.

2.11.3 Decide which data to use in the further analyses below

With the limitations of my own hand coding in mind, I am less troubled by the relative inaccuracies of the naive Bayes relevance classification model and the support vector machine outcome classification model.

Nonetheless, I still believe that my hand-coding results – while unlikely to be 100% accurate – are more accurate (perhaps significantly more accurate) than my machine-coding results, if for no other reason than the fact that my classification algorithms were trained using my somewhat inaccurate hand-coded data and then introduced their own further inaccuracies in turn.

Therefore, I have decided to use the *hand-coded* relevant dfm (which includes only the hand-coded relevant opinions and their hand-coded outcomes), rather than the relevant set dfm (which includes both the hand-coded and the machine-coded relevant opinions and their hand-coded and machine-coded outcomes), for purposes of my further analyses in the steps below.

2.12 Calculate outcome-predictive feature weightings

In this step, my R code extracts the outcome-predictive weights from the support vector machine outcome classification model (which was trained on the *hand-coded* relevant dfm containing my own classifications as to the outcomes for the opinions I had found to be relevant) in order to identify and rank in descending order the bigrams that have the greatest predictive power with respect to judicial outcomes.

Based on a manual examination of these bigram features – identifying those features that are conclusory (e.g., “pierce_veil”) or uninformative (e.g., “citat_omit”) as opposed to illuminating (e.g., “insur_coverag”) – my R code then assigns a new outcome-predictive weight of *zero* to the conclusory and uninformative features so that they sink to the bottom in the ranking of outcome-predictive features.

The following table shows the outcome-predictive features that remain in the top 25, ranked in descending order of their importance (on a scale of 100) in predicting whether the judge will disregard the limited liability entity structure.

feature	importance
subsidiari_corpor	75.587
product_liabil	75.587
sole_sharehold	71.149
parent_subsidiari	69.060
individu_sharehold	68.538
parent_corpor	67.363
corpor_sharehold	61.619
board_director	61.358
vicari_liabil	58.616
wholli_subsidiari	55.744
insur_coverag	53.133
strict_liabil	51.175
respondeat_superior	50.914
wholly-own_subsidiari	48.825
separ_distinct	48.303
corpor_structur	47.650
insur_polici	47.520
exercis_control	45.822
also_contend	43.603
parent_compani	43.342
corpor_name	43.081
common_ownership	43.081
corpor_sham	43.081
equit_principl	43.081
vicari_liabl	42.037

It appears from the bigram features in this table that judges focus on the following questions in deciding whether to respect or disregard the limited liability entity structure:

- whether the entity was owned by an individual entrepreneur, or was instead owned by another entity such as a parent corporation (“subsidiari_corpor,” “parent_subsidiari,” “individu_sharehold,” “parent_corpor,” “corpor_sharehold,” “wholli_subsidiari,” “wholly-own_subsidiari,” “corpor_structur,” “parent_compani,” and “common_ownership”)
- whether the people harmed were members of the public (“product_liabil”) rather than employees of the entity itself
- whether the owner and the entity had respected the limited liability entity structure themselves, and

whether the owner had controlled and participated in the operations of the entity to such an extent that the owner and the entity should be considered as a single person ("sole_sharehold," "vicari_liabil," "strict_liabil," "respondeat_superior," "separ_distinct," "exercis_control," "corpor_name," "common_ownership", "corpor_sham," and "vicari_liabl")

- whether the entity had insurance coverage ("insur_coverag" and "insur_polici")
- whether there were *multiple* reasons for the judge to consider disregarding the limited liability entity structure ("also_contend")
- whether achieving a just and fair outcome required the judge to consider basic equitable principles in deciding whether to disregard the limited liability entity structure ("equit_principl")

2.13 Compare outcome-predictive results to professors' published findings

In this step, I manually examine the outcome-predictive weights of the ranked bigram features in an attempt to confirm the professors' published finding that judges – who frequently discuss the observance of corporate formalities and the adequacy of the entity's capitalization at length in their opinions – actually pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

Note in this regard that few if any of the top 25 outcome-predictive features in the above table appear to involve the observance of corporate formalities or the adequacy of the entity's capitalization, at least explicitly.

My findings and thoughts in this regard are set forth below in the Results section of this report.

2.14 Create topic model for hand-coded relevant opinions

In this step, my R code uses the topicmodels package to (i) create a three-topic unsupervised learning model for the opinions in the *hand-coded* relevant dfm, (ii) identify the bigram features that are most associated with each of these three topics, and (iii) calculate their respective *topic-model weights* (i.e., the strength of their association with the topic) in each instance.

For each of these topic-predictive bigram features, my R code then multiplies their respective *topic-model* weights by their respective *outcome-predictive* weights (which were extracted from the support vector machine outcome classification model in the step above) in order to identify and rank in descending order for each of the three unsupervised learning model topics the bigram features that are not only the most *associated* with that topic but also the most *predictive* of judicial outcomes.

Since an outcome in which the judge *disregards* the limited liability entity structure was defined as the *positive* outcome in the support vector machine outcome classification model, a higher outcome-predictive weight for a bigram feature means that a judge is more likely to disregard the structure and rule against the owner in a case where that factor is present.

2.14.1 A problem with the Rstudio knit functionality

In generating this report several times (without changing anything in the saved Rmd file), I discovered that the R code for this step generates a *different* result each time the same Rmd file is knitted. Therefore, as an experiment, I executed the *R code itself* (rather than knitting it) several times, and each time it generated the *same* result. The R code includes a local seed-setting parameter – which is why the R code itself creates reproducible results – but for some reason the R code executes differently each time when the Rmd file containing the same R code is knitted.

In order to address this issue – which may be a "bug" in Rstudio based on my review of various related postings on Stack Overflow – I have disabled this portion of the R code in my Rmd file (with an "eval=FALSE" tag) so that it will not execute when the Rmd file is knitted to a PDF file. I have similarly commented out this portion of the R code in my R script file.

Instead, I have saved the results of executing the R code itself in a file named "data/topic-model-table.RData" on my public GitHub site, available at https://github.com/dcbarnard/corporate-veil.

Since this portion of the R code in my Rmd file has been disabled, my Rmd file also includes R code that will load the saved results.

2.14.2 Returning to the analysis

The following table shows the top 25 bigram features for each of the three unsupervised learning model topics, ranked in descending order of the *product* of their respective *topic-model* weights and their respective *outcome-predictive* weights as described above:

Topic 1	Topic 2	Topic 3
parent__corpor	strict__liabil	corpor__form
worker__compens	punit__damag	insur__coverag
subsidiari__corpor	product__liabil	sole__sharehold
parent__subsidiari	singl__busi	corpor__sharehold
wholli__subsidiari	wrong__death	individu__sharehold
product__liabil	also__argu	respondeat__superior
subsidiari__parent	reason__care	vicari__liabil
sole__sharehold	whether__corpor	corpor__fiction
parent__compani	commit__fraud	vicari__liabl
workmen__compens	prima__faci	board__director
separ__distinct	evid__corpor	wholly-own__subsidiari
board__director	tort__liabil	corpor__structur
independ__contractor	award__punit	limit__liabil
tort__liabil	also__contend	alter__doctrin
distinct__entiti	actual__damag	insur__polici
employ__employe	evid__reason	domin__control
employe__employ	consid__whether	abus__corpor
corpor__liabl	essenti__element	liabil__theori
corpor__structur	busi__enterpris	corpor__separ
direct__particip	compet__evid	alter__corpor
parent-subsidiari__relationship	sole__proprietorship	employe__corpor
assum__liabil	perpetr__fraud	person__corpor
action__brought	juri__find	theori__liabil
entitl__matter	fail__provid	alter__theori
control__subsidiari	therefor__find	within__scope

It appears from the bigram features in the columns of this table that the three unsupervised learning model topics may relate to the following types of tort cases in each instance:

- Topic 1 may relate to tort cases involving workers who were injured or killed in the workplace ("worker__compens," "workmen__compens," "independ__contractor," "employ__employe," "employe__employ"), with the result that workers compensation *statutes* are likely to be applicable in these cases
- Topic 2 may relate to tort cases in which the owner had some degree of moral culpability ("strict__liabil," "punit__damag," "commit__fraud," "prima__faci," "award__punit," "perpetr__fraud," and "fail__provid")
- Topic 3 may relate to tort cases in which the owner and the entity had failed to respect the limited liability

entity structure themselves ("corpor_form," "board_director," "corpor_structur," "abus_corpor," and "corpor_separ"); in which the owner had controlled and participated in the operations of the entity to such an extent that the owner and the entity should be considered as a single person ("sole_sharehold," "individu_sharehold," "corpor_fiction," and "domin_control"); or in which the owner itself was *directly* liable on one or more legal theories for the harm caused ("respondeat_superior," "vicari_liabil," "vicari_liabl," "alter_doctrin" "liabil_theori," "theori_liabil" and "alter_theori")

2.15 Determine whether judges decide workers comp cases differently

In their published paper, professors Macey and Mitts announced that they had developed a theoretical framework – supported by the results of their machine-learning analysis – suggesting that judges tend to disregard the limited liability structure only when necessary (i) in order to achieve the purpose of a specific statute or regulation (e.g., protecting the environment or workers) . . . .

As noted in the previous section, one of the three unsupervised learning model topics *does* appear to involve tort cases in which workers were injured or killed in the workplace. Since most workers are covered by workers compensation statutes, this is a good example of a setting in which judges would need to consider the purpose of a specific statute or regulation in deciding whether to respect or disregard the limited liability entity structure.

In this step, my R code divides the opinions in the hand-coded relevant set into two categories – i.e., the opinions that involve workplace injuries and fatalities and the opinions that do not – based on the presence or absence of bigram features associated with workers compensation statutes, and then calculates and compares the *respective percentages* of the opinions in which the judge decided to *disregard* the limited liability entity structure for *each of these two categories.*

Based on this categorization of the 99 hand-coded relevant opinions, 50 (or 51 percent) of the relevant opinions involved workers compensation statutes and the other 49 (or 49 percent) of the relevant opinions did not.

- For the 50 opinions that *involved* workers compensation statutes, the judge decided to respect the limited liability entity structure in 32 (or 64 percent) of the opinions and decided to *disregard* the limited liability entity structure in the remaining 18 (or 36 percent) of the opinions.
- For the 49 opinions that *did not involve* workers compensation statutes, the judge decided to respect the limited liability entity structure in 34 (or 69 percent) of the opinions and decided to *disregard* the limited liability entity structure in the remaining 15 (or 31 percent) of the opinions.

Therefore, although the difference is relatively modest, it does appear from the 99 tort opinions I hand coded that judges are *more likely to disregard* the limited liability entity structure in tort cases that *involve* workers compensation statutes (structure disregarded 36 percent of the time) than they are in other tort cases that *do not involve* workers compensation statutes (structure disregarded 31 percent of the time).

2.16 Compare topic-model results to professors' published findings

In this step, I compare the results of my topic-model analysis with the professors' published results, and consider whether our respective results are consistent with and tend to support the professors' theoretical framework.

My findings and thoughts in this regard are forth in more detail below in the Results section of this report.

2.17 Terminate parallel processing

In this final step, my R code terminates parallel processing.

3 Results

Through my machine-learning analysis of the subset of relevant *tort* opinions, I was able to replicate the essential results of the professors' machine-learning analysis of the full set of relevant opinions. I also reach most, but not all, of the same conclusions the professors did in examining the results of our respective machine-learning analyses.

Since I will question some aspects of the professors' methodology and some of the conclusions they reached in the discussion below, I should make clear at the outset that I am under no illusion that my own methodology and conclusions would survive a similar critical analysis unscathed. I greatly respect the professors' groundbreaking application of machine learning techniques to legal research, and I have learned a great deal from reading their published paper, attempting to replicate their methodology and findings, and reaching my own tentative conclusions as to the meaning of the results.

3.1 The Professors' Theoretical Framework

In their published paper, professors Macey and Mitts announced that they had developed a theoretical framework – supported by the results of their machine-learning analysis – suggesting that judges tend to disregard the limited liability structure only when necessary (i) in order to achieve the purpose of a specific statute or regulation (e.g., protecting the environment or workers); (ii) in order to prevent owners from borrowing money at the entity level by misleading lenders into thinking incorrectly that the owner would be personally responsible for the entity's debt; and (iii) in order to prevent owners from removing assets from an entity in anticipation of the entity's imminent bankruptcy, thereby leaving the entity with insufficient assets to repay its creditors.

Since my project focused on a subset of 474 potentially relevant *tort* opinions, rather than the full set of 12398 potentially relevant opinions, I was only able to replicate the professors' methodology in part. For example, the professors' second category – preventing fraudulent borrowing – is irrelevant in the context of bodily injuries and fatalities. Furthermore, there were not enough opinions in the tort set dfm involving entities in a formal bankruptcy proceeding – the professors' third category – for me to replicate the professors' methodology in this respect.

I was, however, able to replicate the professors' methodology with respect to their first category – cases involving a specific statute or regulation – since the hand-coded relevant set dfm contains a number of opinions involving the workers compensation statutes that apply when employees are injured or killed in the workplace. As discussed above:

- My unsupervised learning "topic-model" analysis found that tort opinions involving workers compensation statutes do in fact appear to constitute a *separate category* for purposes of predicting whether judges will respect or disregard the limited liability entity structure.
- My analysis of the 99 relevant tort opinions I hand coded also confirmed that judges are *more likely to disregard* the limited liability entity structure in tort cases that involve workers compensation statutes (structure disregarded 36 percent of the time) than they are in other tort cases that do not involve workers compensation statutes (structure disregarded 31 percent of the time).

I do, however, have one concern regarding the professors' methodology and the conclusions they drew from the results of their machine-learning analysis.

On close reading of the professors' published article, it appears that the professors calculated the *absolute values* of the negative and positive topic-specific, outcome-predictive weights for their various bigram features, thereby removing *directionality* from the results of their machine-learning analysis. See, e.g., footnote 180 on page 148 of their published paper, available on the Cornell Law Review website at https://scholarship.law.cornell.edu/cgi/viewcontent.cgi?article=4647&context=clr.

With the directionality removed, the professors' analytical results only demonstrated that judges treat cases involving the three elements in their theoretical framework *as separate categories* in deciding whether to respect or disregard the limited liability entity structure. Therefore, the professors do not appear to have had sufficient basis for their conclusion that the results of their machine-learning analysis confirm the validity of their theoretical framework, which suggests that judges are *more likely to disregard* the limited liability entity structure when any of their three theoretical framework elements are present in the case.

Instead, I believe that the professors could have divided their hand-coded (and machine-coded) opinions into categories based on the presence of bigram features associated with each of their three theoretical framework elements, and then examined the outcomes for opinions *involving* each of these three elements versus the outcomes for opinions that did not involve any of the three elements. If upon examination the results showed that judges were in fact *more likely to disregard* the limited liability entity structure in cases involving any of the three elements, then the professors would have had sufficient basis for their conclusion that the results of their machine-learning analysis confirmed the validity of their theoretical framework.

In summary, my unsupervised learning "topic-model" analysis found that tort cases involving workers compensation statutes do in fact appear to constitute a *separate category* for purposes of predicting whether judges will respect or disregard the limited liability entity structure (which is consistent with the professors' findings), and my analysis of the hand-coded relevant tort opinions found that judges are in fact *more likely to disregard* the limited liability entity structure in those cases involving workers compensation statutes (which is consistent with the first element of the professors' theoretical framework). As described above, however, I do have one concern regarding the professors' methodology and the conclusions they drew from the results of their machine-learning analysis.

3.2 Importance of the Factors Judges Discuss in Their Opinions

In their published paper, professors Macey and Mitts reported finding relatively low outcome-predictive weights for the bigram features associated with the observance of corporate formalities and the adequacy of the entity's capitalization, and concluded from this finding of low outcome-predictive weights for these bigram features that judges – who frequently discuss these factors at length in their published opinions – actually pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

This is a somewhat improbable conclusion, and therefore it must be critically examined.

Since the professors have not chosen to make their code available to other researchers who might wish to replicate and build upon their analysis, and instead have only described their methodology in their published paper, it is not clear that one can use the published results of their machine-learning analysis – in the context of their described methodology – to rule out the *more likely* possibility that judges *actually do base* their decisions whether to respect or disregard the limited liability entity structure on the very factors they discuss at length in their published opinions.

Recall that the professors' methodology includes a step removing common "stop words" in the construction of bigram features. As a result, the word "not" – which could be critically important in this context – may have been removed (e.g., the word "not" is included within the standard list of stop words in the quanteda package).

To illustrate why the omission of this one word – "not" – could be critically important, imagine an extreme hypothetical setting in which judges *always* apply a several-factor test (addressing *only* the observance of corporate formalities and the adequacy of the entity's capitalization) and *never* take other factors into account in deciding whether to respect or disregard the limited liability entity structure.

In this extreme hypothetical setting, the judges' opinions would nearly always contain the bigram features associated with these factors, regardless of whether the judge ultimately respected or disregarded the limited liability entity structure, and with similar prevalences in either outcome. As a result, these factors – which in this setting are *perfect* outcome predictors – would nonetheless have a low outcome-predictive weight (perhaps even zero), unless outcome-switching words such as "not" were taken into account.

Even if one assumes the professors' methodology *considered* these outcome-switching words as potential predictors of outcomes, there is another problematic aspect of the professors' methodology, which goes to their use of two-consecutive-word bigrams. In a setting where negating words such as "not," "never," and "no" can be *outcome-switching*, these words will often fall some distance away from other *outcome-predictive* words in the text. For example, the judge's opinion in one case might contain the phrase, "It does *not* appear that the owner adequately capitalized the entity," and in another case might contain the phrase, "It appears that the owner adequately capitalized the entity," with very different potential implications for the outcomes in the two cases. Therefore, it would be more appropriate in the NLP methodology to use "within X words" predictive features – in order to "pair" the *outcome-switching* words with the associated *outcome-predictive* words – rather than simple bigrams.

Since I intentionally replicated the professors' methodology, I too found relatively low *outcome-predictive weights* for the bigram features associated with the observance of corporate formalities and the adequacy of the entity's capitalization. For the reasons discussed above, however, it does not necessarily follow from these findings that judges – who frequently discuss these factors at length in their published opinions – *actually* pay relatively little attention to these factors in ultimately deciding whether to respect or disregard the limited liability entity structure.

In summary, I am unable to conclude on the basis of the professors' description of their machine-learning analysis and their reported results (or on the basis of my own, consistent results following their methodology) that judges *actually* pay relatively little attention to the very factors they discuss at length in their published opinions in deciding whether to respect or disregard the limited liability entity structure.

In order to reach this somewhat improbable conclusion, it would be necessary to examine the professors' machine-learning code and ideally their training set of hand-coded opinions (neither of which they have chosen to make available), replicate their results, and confirm that the relatively low outcome-predictive weights that the professors report for the very factors that judges discuss at length in their opinions are *real* rather than mere artifacts of the chosen methodology.

4 Conclusions

4.1 Applications of machine-learning in legal research

It appears from my analysis and findings in this capstone project that machine-learning techniques can be quite useful in performing legal research.

In particular, it appears that properly trained classification algorithms can be reasonably accurate in predicting whether or not an unread opinion is *relevant.* Since it would take a considerable amount of time for even a well-trained human reviewer to read a large number of potentially relevant opinions and decide whether each case is actually relevant, the use of a properly trained classification algorithm can save a great deal of human effort (and money) in the course of a large legal research project.

However, these classification algorithms do not appear to be as accurate in predicting the binary *outcomes* of unread relevant opinions. One reason for this relative difference in accuracy goes to the importance of outcome-switching words such as "not," "never," and "no" in this binary-outcome context. Features that are highly predictive of binary outcomes, but that occur with similar prevalence in the opinions with either outcome, will appear to have very little predictive power if outcome-switching words are ignored (e.g., if they are deemed to be "stop words") in the machine-learning methodology. Another reason for this relative difference in accuracy is the fact that these outcome-switching words often fall some distance away from the other outcome-predictive words in the text. If the predictive features are constructed as simple two-consecutive-word bigrams in the machine-learning methodology, then the outcome-switching words will generally not be paired with the other outcome-predictive words, and the accuracy of the classification model will suffer accordingly.

I have come to understand that these classification algorithms must be used with careful attention to the appropriateness of the methodology for the task at hand. Without careful attention to the appropriateness of the methodology, it is all too easy to reach incorrect or unsupported conclusions from the results of the machine-learning analysis.

4.2 Areas for Further Research

In my own mind, and informed by my reading of 99 relevant "veil-piercing" opinions involving bodily injuries and fatalities, I suspect that judges focus primarily on *"equitable" considerations* such as justice and fair play in making their decisions whether to respect or disregard the limited liability entity structure.

- One such equitable consideration is the extent to which the owner and the entity themselves have respected the separateness they are asking the judge to respect. This would explain the extensive discussion by judges in their opinions regarding the observance of corporate formalities, such as holding shareholder and board meetings, keeping minutes of those meetings, and maintaining separate accounting records and checking accounts. It would also explain the extensive discussion by judges in their opinions regarding whether the owner had controlled and participated in the operations of the entity to such an extent that the owner and the entity should be considered as a single person.
- Another equitable consideration is the moral culpability of the owner. This would explain the tendency of judges to disregard the limited liability entity structure in formal bankruptcy proceedings when the owner has removed assets from an entity in anticipation of the entity's imminent bankruptcy, thereby leaving the entity with insufficient assets to repay its creditors. It would also explain the tendency of judges to disregard the limited liability entity structure in situations where the owner has borrowed money at the entity level by misleading lenders into thinking incorrectly that the owner would be personally responsible for the entity's debt. It may also explain the extensive discussion by judges in their opinions regarding the adequacy of the entity's capitalization, since a material shortfall in capitalization (especially when the entity also lacks adequate insurance coverage) suggests that the owner was attempting to avoid personal responsibility for engaging in *overly* risky activities at the limited liability entity level.

With this capstone project completed, I would like to conduct a similar machine-learning analysis, this time in an attempt to confirm my own "theoretical framework" as described above, and this time using "within X words" features rather than bigrams as outcome predictors (since potentially outcome-switching words such as "not," "never," and "no" will often fall some distance away from other outcome-predictive words in the text).

Rather than limiting my analysis to just the subset of 474 opinions involving *bodily injuries and fatalities*, as I did for the purposes of this capstone project, I might instead analyze the full set of 12398 opinions involving decisions by judges whether to respect or disregard the limited liability entity structure.

While this change in scope would necessitate the hand coding of many additional randomly selected opinions, the accuracy of my relevance and outcome predicting models would likely increase, perhaps substantially, in the process.

- For example, it appears from the professors' published paper that their naive Bayes classification algorithm (trained using hand-coded data from a random sample of ~1,000 opinions drawn from the full set of ~10,000 opinions) could make relevance *and* outcome predictions for the unread opinions with an accuracy of 77 percent.
- By way of comparison, my naive Bayes *relevance* classification model, which was trained on a random sample of only 237 potentially relevant tort opinions that were hand coded, has an accuracy of only 73 percent (and an F1 of only 71 percent) in predicting the relevance of unread opinions.
- Similarly, my support vector machine *outcome* classification model, which was trained on only the 99 hand-coded opinions that were found to be relevant, has an accuracy of only 68 percent (and an F1 of only 58 percent) in predicting the outcome of unread relevant opinions.

If I were to hand code additional opinions, I would use my relevance classification model in order to focus my further reading on just the opinions that were likely to be relevant. My initial hand coding of a random selection of 237 potentially relevant opinions downloaded from the CourtListener website API (using its Google-type search capability in order to identify potentially relevant opinions) revealed that only 42 percent of the potentially relevant opinions were actually relevant when I read them. Since my naive Bayes relevance classifier has an accuracy of 73 percent, I could use it to pre-screen the additional, unread opinions that I was about to read and hand-code, thereby making more efficient use of my time.