Kylie L. Anglin (2019) Gather-Narrow-Extract: A Framework for Studying Local Policy Variation Using Web-Scraping and Natural Language Processing, *Journal of Research on Educational Effectiveness*, 12:4,685-706, DOI: 10.1080/19345747.2019.1654576

Gather-Narrow-Extract: A Framework for Studying Local Policy Variation Using Web-Scraping and Natural Language Processing

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant #R305B140026 to the Rectors and Visitors of the University of Virginia. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

Abstract

Education researchers have traditionally faced severe data limitations in studying local policy variation; administrative datasets capture only a fraction of districts' policy decisions, and it can be expensive to collect more nuanced implementation data from teachers and leaders. Natural language processing and web-scraping techniques can help address these challenges by assisting researchers in locating and processing policy documents located online. School district policies and practices are commonly documented in student and staff manuals, school improvement plans, and meeting minutes that are posted for the public. This paper introduces an end-to-end framework for collecting these sorts of policy documents and extracting structured policy data: the researcher gathers all potentially relevant documents from district websites, narrows the text corpus to spans of interest using a text classifier, and then extracts specific policy data using additional natural language processing techniques. Through this framework, a researcher can describe variation in policy implementation at the local level, aggregated across state- or nation-wide populations even as policies evolve over time.

Introduction

Students are fundamentally impacted by policies made at the district level. The nation's 13,500 districts make policies regarding hiring, resource allocation, and the nature of educational programs (Cohen & Spillane, 1992). Even in the case of national policies such as No Child Left Behind and the Every Student Succeeds Ac*t*, federal legislation incentivizes states to enact reform, states choose how to respond, and then they tend to pass implementation details to districts (Berman & McLaughlin, 1977; Coburn et al., 2016; Cohen & Spillane, 1992; Wong et al., 2018). This elaborate and decentralized system of governance results in remarkable variation in the creation and implementation of policies. Yet, research on policy implementation is scarce compared to policy evaluation (Coburn et al., 2016; Haskins & Baron, 2011; Loeb & McEwan, 2006).

Researchers encounter two challenges in understanding how districts translate state and federal policies. First, administrative datasets may be limited in capturing direct responses to an opportunity or mandate. For example, many states give charter schools considerable latitude in designing their labor force and school environment. Yet, the vast majority of analyses that attempt to understand how charters respond to this flexibility only account for a few inputs like class size, per-pupil expenditures, or the fraction of teachers with an advanced degree (Dobbie & Fryer, 2013). Second, in cases where the researcher collects direct information about teachers and administrative leaders' responses to a policy opportunity or mandate, data collection is expensive and time consuming, with sometimes limited generalizability of results.

Natural language processing (NLP) techniques can help address these challenges by assisting researchers in identifying and processing policy implementation documents located online. School district policies and practices are commonly documented in student and staff manuals, union contracts, school improvement plans, and meeting minutes posted for the public. In the past, these unstructured data have been difficult to convert into analyzable datasets without relying entirely on hand-coding documents, a tedious and error-prone method of extracting data. Recent software innovations in NLP, however, have made the accurate extraction of data from text more accessible to researchers wishing to access new sources of information on educational policies.

This paper introduces an end-to-end framework for collecting policy documents and extracting structured policy data. The process is conducted in three steps: First, the researcher builds a web crawler to *gather* all documents from district websites. Second, she trains a text classifier to *narrow* the collection of documents to those describing local policies of interest. Third, she uses additional NLP techniques to *extract* policy data from relevant spans of text. This process is analogous to catching fish; the fisherman casts a wide net, throws out unwanted debris, and cleans the fish that are worthy of eating.

The gather-narrow-extract framework creates a repeatable pipeline that searches local education agency websites and produces an aggregated, structured dataset of local policies. The primary advantage of this approach is establishing a semi-automated and systematic process of gathering data. Applied at a single point in time, the gather-narrow-extract pipeline is useful for maximizing generalizability (by including the full universe of potential data sources) and replicability (by applying and automating the same data decisions to all data documents). Applied over multiple points in time, the researcher’s pipeline can be used to quickly update both the universe of documents (e.g., assessing how a policy’s legislation promulgates across districts throughout the state) and the specific content of these documents (e.g., assessing how district’s specific policies are modified over time). Data gathered using the gather-narrow-extract framework is useful for examining descriptive patterns in the policy implementation of school entities, as well as for constructing quasi-experimental policy evaluations.

This paper acts as a springboard for researchers hoping to study local policy variation using publicly available policy documents. The first section introduces the gather-narrow-extract framework as a general strategy for how to collect and process policy documents from the Internet using automated techniques. Because implementation of the gather-narrow-extract framework requires a working knowledge of NLP, the second section provides a primer on how computers process text and how this may be leveraged to identify policy documents and classify them as indicative of local policies. The third section illustrates an application of the framework for the collection of district-level policy data in order to demonstrate the set of decisions researchers face using automated techniques to collect policy data. Finally, the paper concludes with a discussion of the strengths and limitations of web-scraping and NLP to collect education policy data.

The Gather-Narrow-Extract Framework

Before discussing how researchers can automate the collection of online policy data, it is useful to first think through how this data may be manually collected. Consider a project that requires a researcher to document the landscape of school uniform policies across some state. This information is almost certainly found in student handbooks posted on school websites, and so the first task is to collect the student handbooks of every school in the state. The researcher makes a list of schools and systematically searches each website to download the student manual. After she has collected the documents, she works through them one by one, skimming until she finds the section that discusses the dress code. If the dress code includes a uniform, she notes this in a spreadsheet, where *1* signifies a school uniform requirement and *0* signifies a dress code that does not require a uniform. If the researcher cannot find the student manual or the manual does not discuss a dress code, she enters the data as missing for that school.

This example is representative of the sorts of challenges researchers face in documenting local policy variation. First, the researcher does not know exactly where the policy information is located online. Policy documents are sometimes located at a single location, such as a department of education website, but they more often exist across multiple websites that may or may not be identified. In this example, student manuals are not housed at a single known location, but can be found somewhere on individual school websites. Visiting and searching the population of school websites for student manuals is time consuming. Second, policy documents contain large portions of text that are irrelevant for a given research question. Student manuals discuss a broad range of policies not related to uniforms or dress code, meaning the researcher is required to weed through the document until she finds the relevant text. Third, because a researcher is hand-coding the policies of each individual school, the sheer number of observations in a sample is prohibitive. As the sample size increases, so does the potential for data entry error. In the student manual example, even if the researcher had every student manual in the state with the relevant dress code text highlighted, the researcher would still need to read each piece of text and key in *1*s and *0*s as appropriate. All told, these challenges result in a manual process that is time consuming, resource intensive, and prone to data entry error.

Each of these three challenges is addressed in the gather-narrow-extract framework. The challenge of indeterminate location can be mitigated by a web crawler that imitates the actions of a researcher, systematically searching a set of websites by following paths of hyperlinks. The crawler is fed a URL, identifies the hyperlinks within that web page, and adds them to an internal list of URLs to visit. It repeats the process for each URL until it runs out of unique pages to visit or reaches some other pre-defined stopping criterion, such as a number of links beyond the original URL. In the context of school policy data, the researcher can feed the web crawler a list of school websites (many states maintain such a list) and code the crawler to search each website on the list and copy the URL to every document it can find (PDF, Word Docs, etc.), thereby ensuring that the researcher has the location of every accessible document posted by each school on the list. The full set of raw text scraped from these URLs is the researcher's text corpus. At this point, the corpus should include text from the population of all relevant documents, but it will also include text from irrelevant documents. This is not a problem but rather a feature of the gather-narrow-extract framework. A key insight of the framework is that the researcher does not need to identify the location of every policy document before scraping. Instead, she scrapes every document and narrows the text later.

The problem of irrelevant text (including irrelevant documents and irrelevant text within relevant documents) can be addressed using text classification. In text classification, text characteristics are used to predict the category to which a document belongs. For example, a document's vocabulary can be used to predict whether the document is a student manual or some other irrelevant document. A researcher can either explicitly define the function between document characteristics and document type or train an algorithm to learn the important characteristics using a set of documents labeled with the appropriate category. Text classification can be used to narrow the text corpus to relevant documents and to narrow document texts to relevant portions. In the school uniforms example, a text classifier would be used to narrow the set of documents to only include student manuals and then to narrow the text of student manuals to paragraphs discussing dress code.

The problem of hand-coding large numbers of documents can again be addressed using

NLP techniques. In this stage, the researcher creates a text classifier to predict each school's policy. The researcher might classify each school's dress code policy based on whether the text contains the word *uniform*. Or she can label a subset of documents and train an algorithm to learn the features of texts that are indicative of requiring a school uniform. This algorithm can then be applied to the full population of schools in order to classify each student manual as either predictive of a uniform requirement or not, and automatically code the school's treatment variable with a 1 or 0.

From there, the researcher has an end-to-end process for collecting policy documents from the Internet and transforming them into structured policy data: she *gather*s all potentially-relevant documents from district websites, *narrows* the text corpus to spans of interest using a text classifier, and then *extracts* specific policy data using additional classifiers or search criteria. The process is generalizable, but can and should be adopted to a researcher's purpose and context. At times, a researcher may have the relevant policy documents in hand, but she may still wish to narrow lengthy texts to relevant portions. Alternatively, she may only need to use the framework to gather policy documents but then may choose to hand-code the documents according to different policies. Regardless, this framework for collecting implementation data relies on NLP for identifying and extracting policy information. By using a text classification model to learn from a few manually annotated documents, a researcher can collect and process a previously infeasible number of documents quickly.

Using Text Classification to Narrow the Text Corpus and Extract Policy Data

In the gather-narrow-extract framework, a researcher uses text classification in both the narrow and extract phases. When narrowing the corpus of gathered documents, the researcher uses text characteristics to predict the relevance of a document. When extracting policy information from a text, the researcher uses text characteristics to predict the presence of some policy or implementation detail. The text classification process can be summarized in three steps. First, the researcher represents the raw text as a set of numerical variables, or **features**. Second, the researcher maps these features to a set of predicted categories using researcher-determined schemas or any of a variety of statistical techniques. Then, the researcher can use the predicted values in subsequent tasks including descriptive or causal analysis (Gentzkow et al., 2017).

This section provides a primer on how researchers may extract features from text and how these features may be used to classify documents. The primer is not meant to act as a comprehensive guide to NLP, but rather to provide the reader with some intuition for how computers process text and how they may leverage NLP in text classification[1].

Feature Extraction

In order to extract meaning from text, a computational approach requires transforming the series of characters that constitute a text into analyzable features. The fundamental unit of text analysis can range from a single character to series of paragraphs. Individual occurrences of these units are termed **tokens**, and the process of breaking down a document into its constituent units is called **tokenization**; most commonly, tokenization occurs at the word level. There are well-accepted automated approaches (**tokenizers**) for splitting texts into their constituent tokens (Bird, Loper, & Klein, 2018). After tokenization, a document is represented as an ordered vector of words and punctuation. This vector may then be transformed into numerical features that characterize the text; features range in complexity from length and word frequencies to word order and patterns.

[1] Readers may turn to Jurafsky and Martin (2018) for more comprehensive coverage of NLP techniques, and to Grimmer & Stewart (2013) and Gentzkow et al. (2017) for a more comprehensive review of text-as-data methods in political science and economics.

A researcher's choice of features depends on the classification task at hand. The most complex classifiers do not require any feature engineering on the part of the researcher. These classifiers, discussed later, learn the characteristics of textual classes using only labeled examples and non-examples, empirically determining the most informative features from raw text. However, these models are often complex and difficult to interpret, and they may require large amounts of computing resources and labeled data to train from scratch. At the other end of the spectrum, a researcher can classify texts using simple schemas and numerical features (such as document length, publication date, and observed frequency of known key words) when she has a particular theory of the distinguishing texts' characteristics fitting a class of interest. If the researcher does not have a strong theory, she can use the text's full vocabulary (for example, document-term frequency matrices) to create features; from there, she can use machine learning methods to determine the most informative features.

Length. The simplest feature describing a text is generally its length, which is the number of tokens in a document. This is often a useful piece of information for predicting document relevance to a given task. For example, a text is unlikely to be a student manual if it only consists of a few hundred words. Text length can also provide interesting insights about the document's author. Beattie et al. found, for example, that students who write longer responses to survey questions about goals tend to have higher college GPAs than their peers (Beattie et al., 2018). Similarly, policy researchers may be interested in determining if the length of a school improvement plan is related to a school's commitment to change, or if the length of a teachers' union agreement is related to the power of the local union over school policies.

Key Word Occurrence. Beyond length, a vector of tokens may also be searched for instances of key words identified by the researcher as providing information related to a research question. This method is applied by Bettinger et al. (2016) in their analysis of the effect of online course interactions on students' academic success. The researchers create a dictionary of student names and loop through every post in the forum to identify whether the post refers to another student by name. The occurrence of a peer's name is then coded as an instance of peer interaction. In the case of policy research, analysts may choose to use key word searches to weed out irrelevant documents (for example, discarding all documents without the phrase *student manual*) or identify policies (for example, searching student manuals for the term *uniform*), though such choices should be tested for validity. When the key words of interest are better represented as a pattern, regular expressions can be helpful. Regular expressions are a standard syntax for representing string patterns using a series of characters. Each character in a regular expression is either a literal or a special search instruction (for example, the regular expression `a.` will produce a match for the letter `a` followed by any character other than a line break). Regular expressions are particularly helpful in searching for entities with standard formats, such as dates, addresses, and currencies[2].

Document-Term Matrix. If a researcher is interested in a text's full vocabulary, they may represent each text using a vector that counts the number of times each unique word in the text corpus vocabulary occurs in each document *N*. So, each document $(i = 1, \ldots, N)$ is represented by a vector $W_i = (W_{i1}, W_{i2}, \ldots W_{im})$, where W_{im} counts the number of times the *m*th word occurs in the *i*th document. This collection of vectors, $W_i \ldots W_{m,}$ is referred to as a **document-term matrix** and can be used to compare word frequencies across documents and categories.

Document-term matrices can quickly grow to large dimensions, as each unique word is

[2] For more information on regular expressions, I recommend *Mastering Regular Expressions* (Friedl, 2002).

its own column and a corpus can contain hundreds or even thousands of unique words. So, a key challenge in NLP is to determine which and how many terms to analyze. It is particularly helpful to ignore words like *a, an, it, the,* and *further*, which are found in many documents but convey little information. Consequently, these words may be treated as **stop words** – commonly used words that an automated approach should be coded to ignore. Many software packages maintain pre-defined lists of stop words that are automatically excluded from textual analysis[3].

Document-term matrices can be further improved by treating all derivatives of a word as a single entity – for example, treating the words *organize, organizes*, and *organizing* as occurrences of the root word, *organize*. This can be accomplished through either **stemming** or **lemmatization**. Stemming is the task of stripping a word of any **affixes** – an element placed at the beginning or end of a root word (Manning et al., 2009). **Lemmatization** also removes inflectional endings and returns the root form of a word, known as the **lemma** (Manning et al., 2009). A document's vocabulary, lemmatized and cleaned of stop words, can provide meaningful information. For example, by comparing word frequencies between college students who out- and under-perform expectations (based on high-school GPA), Beattie et al. (2018) find that over-performers tend to express more philanthropic goals than under-performers. For each word in the cleaned text corpus (student responses to a goal-setting questionnaire), the authors compared the proportion of under- and over-performers using that word and identified the terms *human*, *people*, *provide,* and *helpful* as predictive of over-performance. Likewise, a policy researcher might compare the document-term matrices of improvement plans from schools with a steep increase in student achievement to those with flatter trends in order to identify potentially important features of a school improvement plan. This is similar to the approach taken by Sun et

[3] For example, Python's Natural Language Toolkit (NLTK) maintains a list of 179 stop words (Bird et al., 2018).

al. (2019), who identified the statistical relationship between the topics found in reform strategy measures and student achievement.

Term Frequency-Inverse Document Frequency. Even when applied on lemmatized texts, word frequencies as described above can suffer from an unintelligent weighting system: although all terms are considered equally important, many terms have little or no discriminating power in determining document relevance or identifying local policies. As a motivating example, a collection of school improvement plans is likely to feature the term *school* in almost every document, effectively rendering it no more useful than a stop word. To correct this, a word's relative importance to a piece of text can be calculated using a **term frequency-inverse document frequency** (**tf-idf**) weighting scheme. Formally, tf-idf weights are determined by the following formula:

$$tfidf_{t,d} = tf_{t,d} * \log \frac{N}{df_t},$$

where *tf* represents the term-frequency for a single term, *t*, in a single document; *N* represents the number of documents in the corpus; and *df* represents the frequency of the term across all documents. The tf-idf weighting scheme assigns highest weight to a term when it occurs many times in a small number of documents, lower weights to terms that occur fewer times in a document or in many documents, and lowest weight to terms occurring fewest times in almost every document.

Context-dependent features. None of the previous features make any effort to consider the context of a token's occurrence. They assume documents can be represented as an unordered **bag of words**. If the researcher fears that word meaning is highly dependent on context, she can choose to retain some of the information contained in word order using **bigrams** (token pairs) or **trigrams** (token triples). For example, a bigram would be capable of distinguishing the difference in meaning between the word pair *school uniform* and the single word *uniform* (meaning invariant, rather than an item of clothing).

Finally, to retain some of a token's semantic meaning, a researcher may use **word embeddings**, which are numerical vectors of some pre-defined length (often 300) optimized such that words that appear in similar contexts will be mapped spatially close to one another in the vector space (Mikolov et al., 2013). The underlying proposition of word embeddings is that "a word is characterized by the company it keeps" (Firth, 1957; Manning & Schütze, 1999). Because related words are often used in similar contexts (e.g., *student*, *child*), related words will be assigned spatially close vector representations by a good embedding model. Researchers may choose to use publicly available pre-trained word embeddings (for example, Google offers a large set of word embeddings trained on a Google News dataset) or train the word embedding algorithm on their own corpus. Note that the semantic relationships between words in a word embedding will depend on the context in which the model was trained. Word embeddings trained on a financial corpus may identify the terms *principal* and *investment* to be semantically close, while word embeddings trained in an educational context will identify *supervisor* as a semantic cousin of *principal*. Good word embeddings can improve a researcher's ability to identify key concepts from policy documents by taking the relationship between words into account.

Finally, researchers may also use more advanced NLP techniques to code features like parts of speech, named-entity tags (like person, location and date), contrast, elaboration, and topic change. For example, Kelly et al. (2018) use these more nuanced features to code the authenticity of teachers' question-asking behaviors from transcribed recordings (Kelly et al., 2018; Olney et al., 2017). Policy researchers may find these features particularly useful in transcriptions like school board meeting minutes, where topics and speakers change frequently.

Classifiers

After variables have been extracted from a text, their relationship to a document's type or topic is determined by a classifier. A researcher's choice of classifier will depend on the task at hand. Broadly speaking, there are three categories of classifiers: dictionary-based classification, machine learning classification, and deep learning classification. Dictionary-based methods require prior information on the relationship between features and classes; it is therefore most appropriate when prior information on classes is strong and where information in the text is comparatively weak (Grimmer & Stewart, 2013). Machine learning techniques are generally a good choice when there is little theory guiding the choice of dictionary terms, but the researcher is able to create a set of labeled training documents. If the researcher does not need to interpret the resulting trained classification function, then she can turn to more complex pre-trained deep learning classifiers for an easy-to-implement, high-performing, and context-sensitive approach to classification. In the gather-narrow-extract framework, the researcher does not often need to interpret the relationship between text features and text classification during the narrow phase. It is not necessary to know the features of a student manual, for example; it is only necessary that the researcher is confident they have collected all of the student manuals of interest. On the other hand, when a researcher extracts policy information, interpretation becomes more important for defining the policy of interest, and so uninterpretable classifiers would not be recommended.

Dictionary-based Classification. The most intuitive methods of classification are dictionary methods, which use the occurrence (and/or rate of occurrence) of key words to classify documents into categories. The dictionary of student names found in Bettinger et al. (2016) is an illustrative example of how dictionaries may be used to classify texts: texts containing a word in the dictionary (here, a roster of student names) are classified as indicative of peer interactions while those without a dictionary occurrence are classified as lacking a peer-interaction. Dictionary methods tend to be theory- or intuition-driven rather than determined by the text data at hand. For dictionary methods to work well, their key words need to be well aligned with the construct of interest. It is for this reason that Grimmer and Stewart (2013) argue that a key principal of text analysis is “validate, validate, validate” (p. 5) and that Bettinger et al. (2016) take the time to verify that student names are indicative of response forum posts.

Machine Learning Classifiers. While dictionary methods of classification require researchers to identify words that separate categories ahead of time, supervised learning techniques use the text at hand to determine the relationship between text features and classification. In supervised learning problems, human coders label a representative subset of data (here, plain text documents) with their appropriate classifications. This training set is then used to train an automated classifier, which learns a function between features and classes from the training set. To avoid overfitting the model to noise in the training sample, the researcher also provides a set of labeled texts for validation. In the validation phase, the model’s predictive capability is tested on previously unseen data; a researcher can optimize a classifier by iteratively comparing different specifications to their validation dataset performance. There are many accepted classification algorithms that one might use to categorize text; the rest of this section provides a representative sample.

Researchers are likely familiar with **logistic regression**, which predicts the log-odds probability that an input belongs in one of two categories (e.g., yes or no, relevant or irrelevant, treated or untreated). Logistic classifiers are interpretable and easy to conceptualize, but they suffer from data sparsity problems when considering word frequencies—the number of words tends to far outnumber the number of documents, drastically reducing statistical power. In order to effectively use logistic regression, a researcher will need to either select features using theory or use some method of data-driven feature selection.

One popular strategy of feature selection is the estimation of penalized linear models, in particular using Least Absolute Shrinkage and Selection Operator, or **LASSO** (Hastie et al., 2009). LASSO regression uses a penalty term to shrink regression coefficients towards and to zero. By shrinking coefficients towards zero, LASSO discourages more complex models in order to avoid overfitting the model to statistical noise. By shrinking some coefficients to zero, the algorithm also performs variable selection. The extent to which LASSO shrinks coefficients is determined by the penalty term, which is optimized by minimizing the sum of squared errors in the regression equation. Thus, LASSO may be used to reduce over-fitting the model noise and to select the most informative text features for classification.

Another common text classification model is **support vector machines** (SVM), which treats each labeled observation as a set of coordinates in an n-dimensional vector space, where n is the number of features exposed to the model. Then, a hyperplane is chosen to maximally differentiate the labeled classes in that space, and new unlabeled observations are classified according to the side of the hyperplane they occupy when plotted. Compared to logistic regression, SVMs are better-tuned to the particular challenges of text classification, namely high-dimensional feature spaces where each distinct word in a corpus of documents corresponds to a feature. An SVM's ability to learn is not necessarily hindered by high dimensionality: if training data are separable by a wide margin, results from an SVM can generalize even in the presence of many features (Joachims, 1998).

Deep Learning Classifiers. Some of the newest and most complex classifiers are **convolutional neural networks** (CNN). Unlike the previously mentioned models, CNNs are capable of taking account of a token's location in a text by recognizing patterns in the data using layered non-linear functions called **neural networks**. Well known for their modern applications to digital images for visual classification tasks such as facial recognition (Redmon et al., 2016), CNNs filter data into a series of increasingly complex patterns. Because the convolutional filter preserves spatial relationships between elements of an input vector, it has built-in support for context; an individual element's value (such as a token) is considered in the presence of its neighbors' values, rather than strictly on its own. When a CNN is applied to text data where each word is encoded as a pre-trained word embedding, the model can learn and detect high-level features for context-sensitive content (Kim, 2014). For example, a classification CNN on school clothing regulations might contain a low-level feature for the bigram *dress code,* and a series of high-level filters for negative restrictions (e.g., *students cannot wear shirts with logos*) and positive restrictions (e.g., *students must wear closed-toe shoes*).

A CNN's ability to learn context-dependent features offers high performance on classification tasks and eliminates the need for hand-engineered text features. Further, many generalizable end-to-end CNN pipelines geared toward NLP tasks like text classification are available as open-source software, making application of this class of models to new tasks straightforward for researchers. However, CNNs may require a larger training data set than their simpler machine learning counterparts (like LASSO and SVM) and do not provide an interpretable function between inputs and classification (Erickson et al., 2018).

Validation

Classifier performance is usually evaluated by one or more of four measures: accuracy, precision, recall, and the F-measure. The simplest measure that can be used to evaluate a classifier is *accuracy* – the percentage of observations in the test set that the classifier labels correctly. Accuracy is a weak measure by which to optimize when a researcher needs to distinguish a few positives from many negatives (for example, to identify a few relevant documents from many irrelevant documents). In such a case, the accuracy score for a model that simply labels all observations as negative would be close to 100%, despite its failure to identify any positives. Precision, recall, and F-measure improve on accuracy by taking into consideration the relationships between *true positives* (relevant items correctly classified as relevant), *true negatives* (irrelevant items correctly classified as irrelevant), *false positives* (irrelevant items incorrectly classified as relevant), and *false negatives* (relevant items incorrectly identified as irrelevant). Precision represents the proportion as true positives out of all documents identified as relevant, while recall represents the proportion of relevant documents correctly identified as relevant. An F-measure balances precision and recall by computing their harmonic mean where the balance between precision and recall is weighted by β^2. When precision and recall are given equal weight ($\beta^2 = 1$), this is commonly referred to as the F1 measure (Manning et al., 2009).

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$F_\beta - Measure = \frac{(1 + \ \beta^2) * Precision * Recall}{(\beta^2 * Precision) + Recall}$$

$$F_1 - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Researchers commonly optimize a classifier by iteratively comparing the performance of different specifications on their validation dataset, as described by any of these metrics. The choice of metric depends on the scenario and whether false positives or false negatives are costlier. When using a classifier to identify relevant documents, for example, false positives present a greater risk of incorrect data. Returning to the school uniform example, if a lunch menu is inappropriately classified as a student manual, an algorithm will likely not identify a school uniform policy in the lunch menu, even though the true student manual may contain a uniform policy. In this scenario, researchers will want to maximize precision to ensure data integrity.

Precision, recall, and f-measures are most commonly associated with machine learning and deep learning classifiers, but it is equally important for researchers to validate dictionary-based classifiers. As in the machine learning case, this is done by comparing the results of the dictionary classifier to the human gold standard (Grimmer & Stewart, 2013). For example, in their analysis of the impact of online peer interaction on course performance, Bettinger et al. (2016) feared that using class rosters to identify references to peers would overlook any nicknames used in the forum. To test this, they hand coded 300 forum posts and calculated an accuracy rate of 96%.

Applied Example of the Gather-Narrow-Extract Framework

To illustrate how a researcher may use web-scraping and NLP to collect and extract information from diverse and unstructured policy documents, I will walk through an applied example studying variation in education policy implementation in Texas. In June 2015, the Texas legislature passed House Bill 1842 - Districts of Innovation, granting public schools the ability to exempt the majority of state education regulations, including teacher certification requirements, maximum class sizes, and minimum instruction time (Texas Education Code). The law does not require that districts seek approval for exemptions, but districts must make policy changes transparent by posting a District of Innovation plan (DIP) specifying the exact regulations it plans to exempt. Given the number and scale of school policies that can be waived, the deregulation effort in Texas has the potential to dramatically change the day-to-day operations of many districts in the state.

The Texas Districts of Innovation law provides an ideal context for demonstrating the importance of web-scraping and NLP techniques for two reasons. First, without web-scraping and NLP, an analysis of Texas school district deregulation would be a foreboding task, as a researcher would need to locate and hand-code over 5,000 pages of DIPs. The time-consuming nature of such a task would inhibit a timely description of important events, and, because districts may declare District of Innovation status and amend their documents at any time, the dataset might already be out of date by the time a researcher completes hand-coding exempted laws. Second, DIPs share characteristics with a number of other policy-relevant documents located on district websites: they contain rich data, but they are found in disparate locations, are stored in diverse media, and require the capacity to turn natural language into structured data in order to extract value. Therefore, a demonstration of how DIPs may be collected and analyzed generalizes to a number of relevant educational documents and related research questions.

The following sections contain a step-by-step explanation of the process I followed in pursuit of documenting the regulatory exemptions of each district in the state. These methods were tuned to the specific challenges of DIPs, but the decision-making process is generalizable. Figures 1 and 2 provide a visual overview of the steps and decisions researchers face when they apply gather-narrow-extract for data collection.

<Insert Figures 1, 2 here.>

Step 1: Gather Potentially Relevant Documents using a Web Crawler

The first step to gathering policy documents is to determine a set of seed links for a web crawler to visit (see Figure 1, step 1). Like many states, the Texas Education Agency maintains a list of school district websites; the state also maintains a helpful list of URLs for Districts of Innovation. These links do not typically lead to a DIP; instead, they lead to local district websites containing DIPs someplace within their site hierarchies. This scenario requires a method of retrieving documents without knowing their exact location. I therefore built a web crawler to search each district website for links to documents.

The web crawler followed a loop in which it (1) visited each URL in the list of school district websites; (2) copied every URL that linked to a static document (indicated by the extensions .PDF, .doc, or .docx and the strings “drive.google.com” or “docs.google.com”); and (3) followed any next-level links to additional websites in the site’s hierarchy. The web crawler followed this loop until it copied the URL of every document within three links away from the seed link[4]. While developing the web crawler, I iteratively tested it on a small number of links (see Figure 1, steps 2 through 4), including samples of links leading directly and indirectly to DIPs and links leading to documents in four different storage media: HTML pages, PDF, Microsoft Word format, and Google Docs.

After collecting all of the URLs, I used Apache Tika to scrape the raw text from each HTML page, PDF, Word Document, and Google Doc (Mattmann & Zitting, 2011)[5]. The final result of the web crawler was a dataset of district names, URLs potentially containing DIPs, and extracted plain text.

Step 2: Narrow the Collection of Documents Using a Text Classifier

By design, web crawlers cast a wide net. In this case, every static document was collected, ensuring that no DIPs were inadvertently missed. In total, the web crawler extracted

[4] A number of software libraries are available for simplifying the process of building a web crawler; modules for making HTTP requests and parsing HTML are particularly prevalent. To submit HTTP requests, I used the Requests (Reitz, 2018) library; to parse HTML, I used BeautifulSoup (Richardson, 2017).

[5] Of the 3995 documents scraped from district websites, Apache Tika was able to extract text from 3818.

plain text from 3,743 documents, five documents per district on average. The goal at this stage was to narrow the collection of documents (including DIPs, but also calendars, lunch menus, and other irrelevant texts) to those most likely to be relevant.

I began by labelling a random sample of 385 plain text documents as *true* if the document was a DIP or *false* if it was irrelevant, setting aside 85 for validation and using 300 for training (step 6 in Figure 1). Training data were held in a two-column dataset where the first column contained plain text and the second contained the true/false label. I then applied and compared the performance of four classifiers – a dictionary classifier, a LASSO regression, an SVM, and a CNN. Appendix A provides more information on the specifications and performance of each classifier. For my final model, I chose to implement the CNN, as this was the model with the highest F_1-measure when trained on 300 documents. The CNN does not require any feature engineering on the part of the researcher. From the training text, the CNN created a function predicting DIP status using a set of pre-trained word embeddings and convolutional filters[6]. After training the CNN, I applied it on the random sample of 85 labeled plain text documents set aside for validation. Of these, the classifier correctly identified 96% of the true DIPs (a measure of recall). Of the documents classified as DIPs, all were true positives (a measure of precision).

The output of a CNN is a probability of category membership for each input, and so for each district, I kept only the document with the highest probability of DIP membership—and did so only if the document was positively classified as a DIP (documents with greater than 0.5 probability of DIP classification). Because the model resulted in perfect precision (at least in the 85 document validation sample), I could feel confident that I was unlikely to extract laws from

[6] I used the open-source Python library spaCy (Honnibal, 2017), which includes pre-trained word embeddings and pre-trained convolutional filters.

non-DIPs. Because the model had a high recall rate, I could also feel confident that if a district was missing a DIP, then none of the documents scraped from the district website were likely a DIP. I therefore manually searched for the DIP for every district without a document and added these DIPs to my dataset of DIP text (step 9 in Figure 2).

Step 3: Extract District Policies Using NLP Techniques

After the corpus has been narrowed to the policy documents of interest, the goal is to process the document in order pick up on policy nuance. At this stage, researchers need some method of extracting implementation details from policy documents so that they may define the school district's status with respect to one or more policies. Here, the research question of interest is, *Which regulations does the district exempt, according to the DIP?*

The first step of policy extraction is to examine the policy documents to determine patterns (see Figure 2). In DIPs, Texas statutes are always represented by two to three numerals referencing the education code chapter, followed by a period and two or more numerals referencing the specific statute. This statute-like pattern is represented easily by a regular expression[7], which then acts as a dictionary of search terms (step 3a in Figure 2). I examined whether this regular expression successfully identified every exemption in a small set of ten documents set aside for validation and tuned the regular expression until this was the case. During the tuning phase, I edited the expression to allow an open parenthesis or the section symbol § to precede the regulation (step 6a in Figure 2). To classify the full set of documents, I created an algorithm to loop through every DIP, extracting each instance of the regular expression. If a DIP contained a statute-like pattern, I coded the district as having exempted itself

[7] I used the following regular expression in my Python code: \d{2,3}.\d{2,}. Researchers should note that though the syntax of regular expressions is constant, their specification can depend on programming language and/or software implementation.

from the statute mentioned. The output of this routine was a district-by-rule dataset with an indicator for whether each district exempted each rule.

To validate the full pipeline, I chose 30 Districts of Innovation at random, manually searched for each DIP, and noted each exempted law in the documents. This hand-coded dataset served as a final test set and included 243 exempted regulations. I then compared the test set to the laws extracted using automated techniques in the gather-narrow-extract framework. Of the 243 true exemptions, my automated framework correctly identified 239 true positives and misclassified 4 false negatives. The framework additionally extracted 10 false positives, resulting in a precision rate of 96%, a recall rate of 98% and an F_1-measure of 97%.

Results

<Insert Table 1 here.>

From my district-by-rule dataset, I was able to document the frequency of each district regulatory exemption in the state of Texas. Of the states' 1022 non-charter public school district, 824 have claimed district of innovation status (81%). Table 1 displays the top ten most commonly exempted regulations claimed by these districts. The most commonly exempted regulations concern school schedules – nearly all Districts of Innovation have exempted the statute requiring that they not begin instruction before the fourth Monday in August (98% of Districts of Innovation). Additionally, many district exempt the statute requiring that districts operate for at least 75,600 minutes (44%), and the requirement that the instruction year does not end before May 15th (28%). Districts also frequently exempt the requirement that elementary school class sizes are no larger than 22 students (45%) and that schools inform parents when class sizes exceed 22 students (37%). Finally, districts commonly exempt regulations surrounding hiring and employee contracts, including the requirement that teachers be certified

(88%) and tenured after a few years of service (52%), as well as the requirement that teacher contracts be for a minimum of 187 days (36%).

The results of this applied example demonstrate the power of the gather-narrow-extract technique – it makes descriptive analyses possible at scale and can bring nuance to quasi-experimental analyses. By extracting policy data from publicly posted documents, researchers can explore heterogenous impacts by implementation strategies. Without this district-level dataset, we would not know how districts are using regulatory flexibility, only that they have claimed District of Innovation status. Now, future analyses of the impact of District of Innovation status can differentiate the impact of the near ubiquitous school start date exemption from exemptions we anticipate would have a more substantial impact on student achievement, like class size and minutes of operation. NLP techniques allow researchers to enter the black box of policy implementation, allowing for more nuanced evaluation of policies across diverse populations.

Discussion

As a data collection framework, gather-narrow-extract brings many potential advantages. Web-scraping and NLP drastically reduce the time from research question to data-in-hand, increasing the speed at which researchers can produce answers to timely policy questions. Without NLP, many research questions may be left unanswered because of the resource-intensive nature of manually collecting and hand-coding hundreds or thousands of text documents. Second, gather-narrow-extract increases the replicability of research. Either the original researcher or colleagues can simply rerun the original scripts to update or confirm analyses, or to impose new rules on the same set of documents. Third, when text classification is combined with web-scraping, data collection and analysis can be scaled to entire populations of interest, increasing external validity and statistical power with minimal resources.

However, web-scraping and NLP are not one-size-fits-all solutions to studying local policy variations. The gather-narrow-extract framework and its resulting data have a number of disadvantages. First, there is a startup expense to any automation effort that is only worth paying beyond some level of repetitive action. Employing automated techniques may not be worth this expense in order to parse a few documents (excepting a scenario in which content is changing frequently and the researcher hopes to study changes over time).

Second, documents collected and processed through web-scraping and text classification are observational by definition; as such, they share many of the challenges of other types of observational data. Valid inferences about causality cannot be made with correlational designs. Researchers may match school districts on characteristics found in administrative data in a non-equivalent control group design, however, or they may collect multiple years of data surrounding a policy change in an interrupted (or comparative interrupted) time series design. Though unmeasured confounding factors threaten both of these designs (Shadish et al., 2002), they can yield results comparable to experiments under certain conditions (St. Clair et al., 2016; Wong et al., 2017) .

Third, as with all research, the inferences drawn from scraped documents are limited by the specific sample observed, potentially limiting external validity. A researcher may unknowingly fail to collect documents for some portion of the population, whether because a district did not make a document available on their website or because the document was stored in a location or format that was inaccessible to the researcher’s web crawler. These situations are threats to external validity when interpreting results if a document’s format and location are correlated with other constructs of interest. When faced with a truncated sample of scrapable data, Landers et al. (2016) recommend that the researcher develop and test a data source theory regarding the origin of the online data and the types of policymakers that choose to make data public online.

Fourth, considerations of how constructs are operationalized are critical when a text classifier is used to identify educational policies. A text classifier will only result in a dataset with construct validity if it is trained on a dataset with construct validity. For instance, if researchers plan to use text classification to determine whether a district implements performance-based pay for teachers, they should carefully consider what qualifies as performance-based pay, as well as whether and how this will be identified in any documents collected. Researchers should also think carefully about whether a policy is made at the school- or district-level; will the researcher scrape school or district websites? These decisions should be recorded in the manuscript so subsequent readers can determine whether they agree with the operationalization (Shadish et al., 2002).

Finally, researchers should take care to consider the legal and ethical implications of using web data without the permission of its creators. Federal copyright law generally requires owner consent to repurpose copyright content, but the fair-use doctrine makes an exception for researchers among other protected groups including teachers, reporters, and artists (Title 17 U.S. Code § 107). Still, because of ambiguity in case law and inconsistency across jurisdictions, I agree with the recommendation of Landers et al. (2016) that researchers only scrape publicly available, unencrypted data from websites that do not use specific code in their web pages to discourage automated web crawlers and scrapers. Policy data from such websites may still be extracted manually with few ethical concerns because this information does not concern individual student data, is publicly available, and is commonly protected by state Public

Information Acts (like Texas Government Code 552).

Conclusion

In an era where rich information on educational policies and practices is readily available on the Internet, education researchers face both challenges and opportunities in leveraging data for innovative analyses. Schools and districts frequently maintain policy documents designed to provide information to non-research stakeholders in an easy-to-understand format. Traditionally, researchers have faced obstacles in using this information due to the resource-intensive nature of hand-coding documents. To ignore local policy documents, however, is a missed opportunity—these data are both rich and immediately relevant. The web-scraping and text classification methods described here allow researchers to leverage policy documents without burdening districts and states to reformat their data for analysis. The gather-narrow-extract framework provides researchers a template for how they may extract structured information from student and staff manuals, academic plans, school improvement plans, meeting minutes, and any other number of text documents located on the Internet in an automated fashion.

As more districts use their websites to convey information to staff, students, and parents, researchers can make use of this information to describe changes in local policies quickly, accurately, and cost-effectively. In the past, when policymakers have introduced policies that are anticipated to have a meaningful impact on students, evaluative research has lagged behind, assessing the effect of the policy long after it has passed—and sometimes even after the policy has been revised. Policymakers need strong and timely evidence to aid in decision-making, and the gather-narrow-extract approach provides a method for assisting researchers in meeting this need.

Bibliography

17 U.S. Code § 107. Limitations on exclusive rights: Fair use (1992).

Beattie, G., Laliberté, J. P., & Oreopoulos, P. (2018). Thrivers and divers : Using non-academic measures to predict college success and failure. *Economics of Education Review*, *62*(November 2017), 170–182. https://doi.org/10.1016/j.econedurev.2017.09.008

Berman, P., & McLaughlin, M. W. (1977). Factors Affecting Implementation and Continuation. In *Federal Programs Supporting Educational Change* (p. 238). Santa Monica.

Bettinger, E., Liu, J., & Loeb, S. (2016). Connections Matter: How Interactive Peers Affect Students in Online College Courses. *Journal of Policy Analysis and Management*, *35*(4), 932–954. https://doi.org/10.1002/pam.21932

Bird, S., Loper, E., & Klein, E. (2018). Natural Language Toolkit — NLTK 3.3 documentation. Retrieved July 27, 2018, from https://www.nltk.org/index.html

Coburn, C. E., Hill, H. C., & Spillane, J. P. (2016). Alignment and Accountability in Policy Design and Implementation. *Educational Researcher*, *45*(4), 243–251. https://doi.org/10.3102/0013189x16651080

Cohen, D. K., & Spillane, J. P. (1992). Policy and Practice : The Relations between Governance and Instruction. *Review of Research in Education*, *18*, 3–49.

Dobbie, W., & Fryer, R. G. (2013). Getting Beneath the Veil of Effective Schools: Evidence From New York City. *American Economic Journal: Applied Economics*, *5*(4), 28–60. https://doi.org/10.1257/app.5.4.28

Erickson, B. J., Korfiatis, P., Kline, T. L., Akkus, Z., Philbrick, K., & Weston, A. D. (2018). Deep Learning in Radiology: Does One Size Fit All? *Journal of the American College of Radiology*, *15*(3), 521–526. https://doi.org/10.1016/j.jacr.2017.12.027

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, 1–32. Retrieved from http://annabellelukin.edublogs.org/files/2013/08/Firth-JR-1962-A-Synopsis-of-Linguistic-Theory-wfihi5.pdf

Friedl, J. E. F. (2002). *Mastering Regular Expressions* (Second). Sebastopol: O'Reilly & Associates.

Gentzkow, M., Kelly, B. T., & Taddy, M. (2017). *Text as Data* (NBER Working Paper Series No. 23276). Retrieved from https://www.nber.org/papers/w23276

Grimmer, J., & Stewart, B. M. (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, *21*(03), 1–31. https://doi.org/10.1093/pan/mps028

Haskins, R., & Baron, J. (2011). *Building the Connection between Policy and Evidence*. Retrieved from http://coalition4evidence.org/wp-content/uploads/2011/09/Haskins-Baron-paper-on-fed-evid-based-initiatives-2011.pdf

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. *Spring Series in Statistics* (Second). Retrieved from http://www.springerlink.com/index/10.1007/b94608

Honnibal, M. (2017). spaCy · Industrial-strength Natural Language Processing in Python. Retrieved July 19, 2018, from https://spacy.io/

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In *ECML'98 Proceedings of the 10th European Conference on Machine Learning* (pp. 137–142). Chemnitz. https://doi.org/10.1007/BFb0026683

Jurafsky, D., Martin, J. H., & Computational, P. (2018). *Speech and Language Processing: An Introduction to Natural Language and Speech Recognition.* (Third Edit). Retrieved from https://web.stanford.edu/~jurafsky/slp3/

Kelly, S., Olney, A. M., Donnelly, P., Nystrand, M., & D'Mello, S. K. (2018). Automatically Measuring Question Authenticity in Real-World Classrooms. *Educational Researcher*, *47*(7), 451–464. https://doi.org/10.3102/0013189x18785613

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1746–1751). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181

Landers, R. N., Brusso, R. C., Cavanaugh, K. J., & Collmus, A. B. (2016). A primer on theory-driven web scraping: Automatic extraction of big data from the Internet for use in psychological research. *Psychological Methods*, *21*(4), 475–492. https://doi.org/10.1037/met0000081

Loeb, S., & McEwan, P. J. (2006). An Economic Approach to Education Policy Implementation. In M. Honig (Ed.), *New Directions in Education Policy Implementation* (pp. 169–186). SUNY Press. Retrieved from https://cepa.stanford.edu/sites/default/files/LOEBandMCEWAN.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval. Cambridge University Press*. Cambridge: Cambridge University Press. https://doi.org/10.1109/LPT.2009.2020494

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge: MIT Press. Retrieved from https://dl.acm.org/citation.cfm?id=311445

Mattmann, C. A., & Zitting, J. L. (2011). *Tika in Action*. Manning Publications.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. Retrieved from http://ronan.collobert.com/senna/

Olney, A. M., Samei, B., Donnelly, P. J., & D ’mello, S. K. (2017). Assessing the Dialogic Properties of Classroom Discourse: Proportion Models for Imbalanced Classes. *Proceedings of EDM 2017*, 162–167. Retrieved from http://educationaldatamining.org/EDM2017/proc_files/papers/paper_26.pdf

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2016.91

Reitz, K. (2018). Requests 2.19.1 documentation. Retrieved July 19, 2018, from http://docs.python-requests.org/en/master/

Richardson, L. (2017). Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation. Retrieved July 19, 2018, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Shadish, W. R., Cook, T. D., & Cambell, D. T. (2002). *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. *Experimental and Quasi-Experimental Design for Generalized Causual Inference*. https://doi.org/10.1198/jasa.2005.s22

St.Clair, T., Hallberg, K., & Cook, T. D. (2016). The Validity and Precision of the Comparative Interrupted Time-Series Design. *Journal of Educational and Behavioral Statistics*, *41*(3), 269–299. https://doi.org/10.3102/1076998616636854

Sun, M., Liu, J., Zhu, J., & LeClair, Z. (2019). *Using a Text-as-Data Approach to Understand Reform Processes: A Deep Exploration of School Improvement Strategies* (EdWorkingPape). Retrieved from http://edworkingpapers.com/ai19-68

Texas Education Code (2018). Retrieved from https://statutes.capitol.texas.gov/Docs/SDocs/EDUCATIONCODE.pdf

Wong, V. C., Valentine, J., & Miller-Bains, K. (2017). Empirical Performance of Covariates in

Education Observational Studies. *Journal of Research on Educational Effectiveness*, *10*(1), 207–236. https://doi.org/10.1080/19345747.2016.1164781

Wong, V. C., Wing, C., Martin, D., & Krishnamachari, A. (2018). Did States Use Implementation Discretion to Reduce the Stringency of NCLB? Evidence From a Database of State Regulations. *Educational Researcher*, *47*(1), 9–33. https://doi.org/10.3102/0013189X17743230

Figure 1: Gather and Narrow Policy Documents

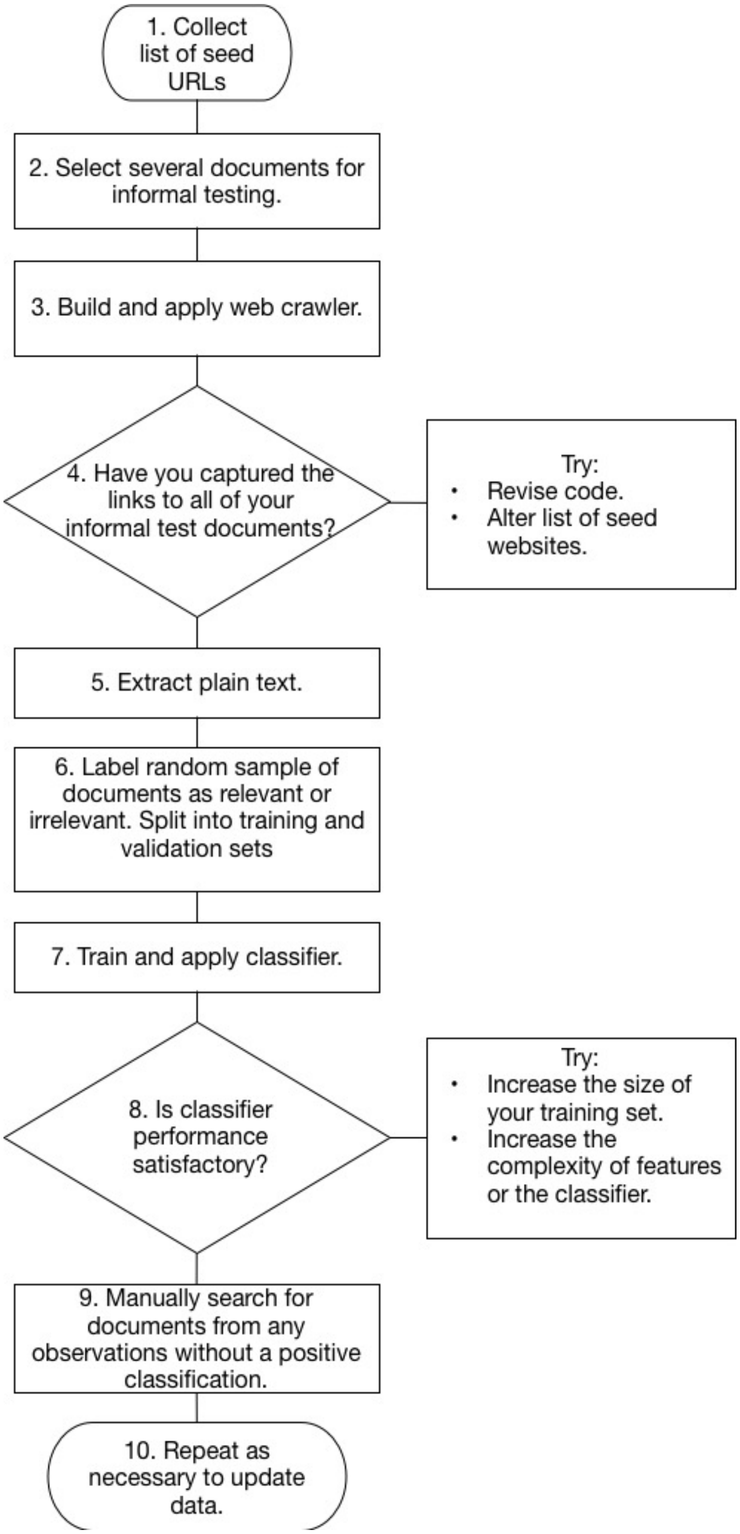

1. Collect list of seed URLs
2. Select several documents for informal testing.
3. Build and apply web crawler.
4. Have you captured the links to all of your informal test documents?
Try:
• Revise code.
• Alter list of seed websites.
5. Extract plain text.
6. Label random sample of documents as relevant or irrelevant. Split into training and validation sets
7. Train and apply classifier.
8. Is classifier performance satisfactory?
Try:
• Increase the size of your training set.
• Increase the complexity of features or the classifier.
9. Manually search for documents from any observations without a positive classification.
10. Repeat as necessary to update data.

Figure 2: Extract Policy Data

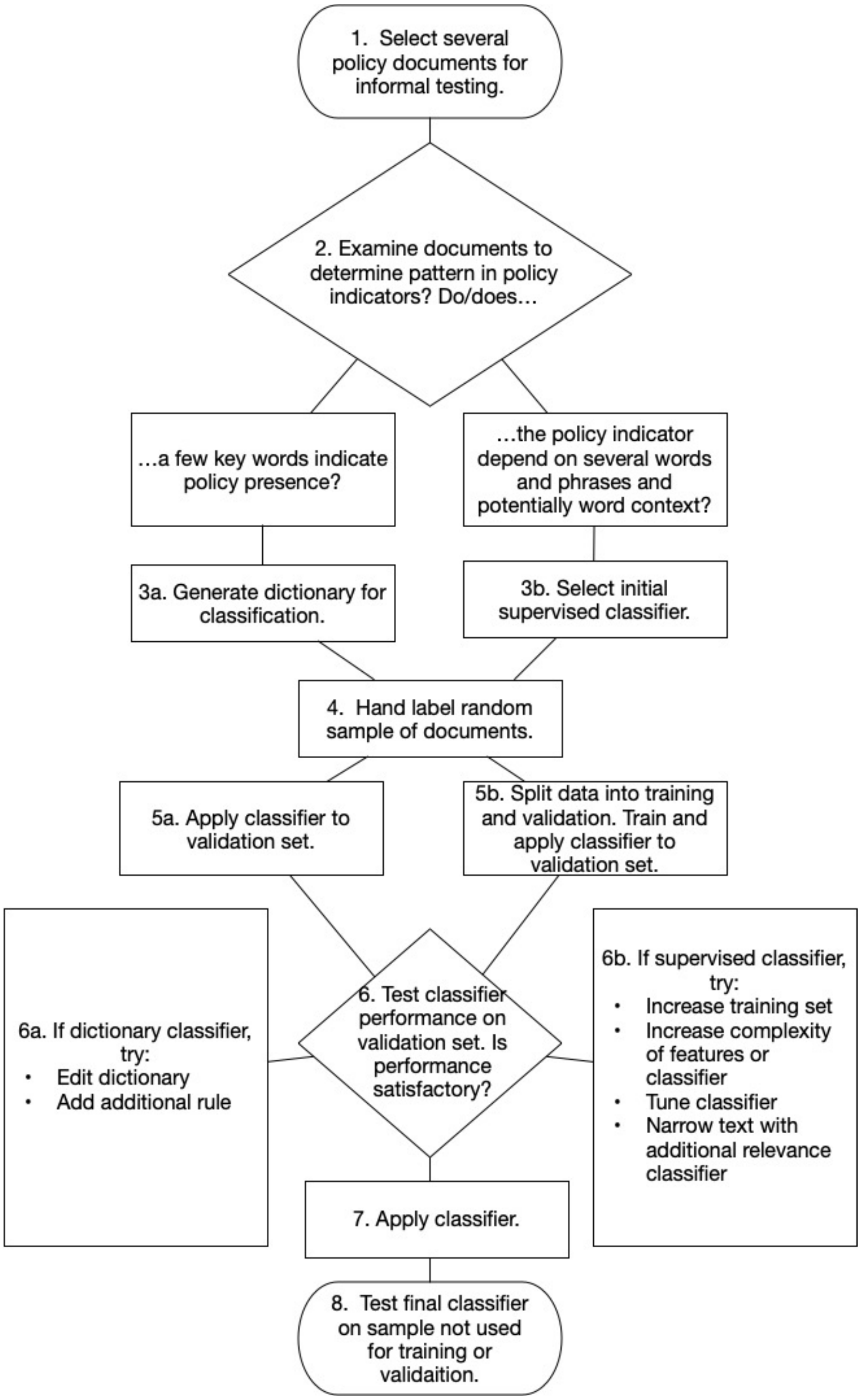

1. Select several policy documents for informal testing.
2. Examine documents to determine pattern in policy indicators? Do/does…
…a few key words indicate policy presence?
…the policy indicator depend on several words and phrases and potentially word context?
3a. Generate dictionary for classification.
3b. Select initial supervised classifier.
4. Hand label random sample of documents.
5a. Apply classifier to validation set.
5b. Split data into training and validation. Train and apply classifier to validation set.
6a. If dictionary classifier, try:
• Edit dictionary
• Add additional rule
6. Test classifier performance on validation set. Is performance satisfactory?
6b. If supervised classifier, try:
• Increase training set
• Increase complexity of features or classifier
• Tune classifier
• Narrow text with additional relevance classifier
7. Apply classifier.
8. Test final classifier on sample not used for training or validaition.

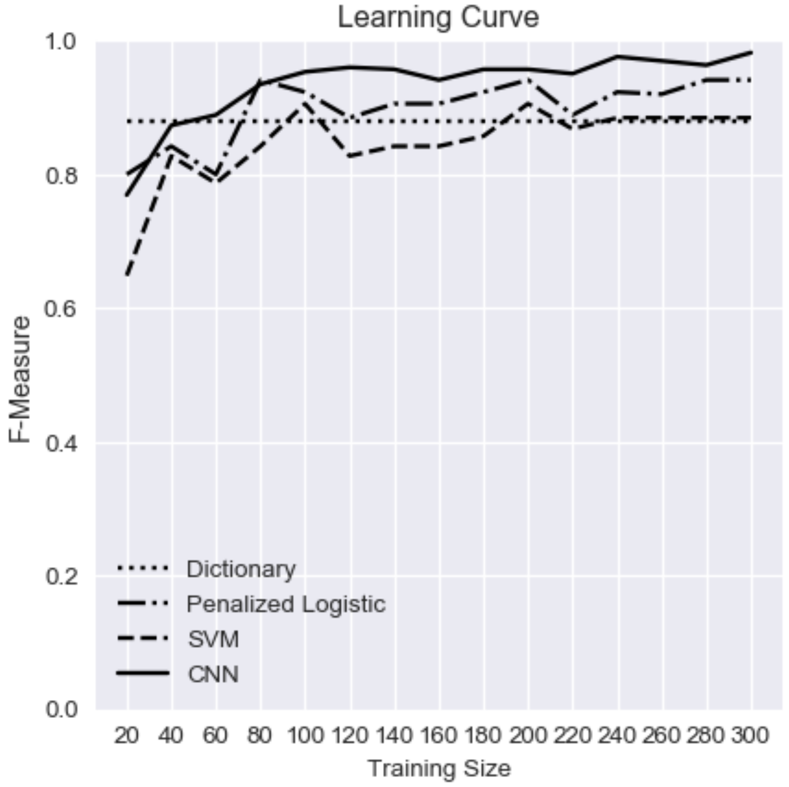
Table 1: Top Ten Commonly Exempted Regulations

Statute	Proportion of Districts of Innovation Exempting
TEC 25.0811 – First Day of Instruction	0.98
TEC 21.003 – Certification Required	0.88
TEC 25.102 – Maximum Probationary Contract Length	0.52
TEC 25.112 – Elementary Class Size Maximum	0.45
TEC 25.081 – Minimum Minutes of Operation	0.44
TEC 25.113 – Notice of Class Size	0.37
TEC 21.401 - Minimum Service Required	0.36
TEC 21.057 – Parental Notification (of uncertified teachers)	0.33
TEC 21.053 – Presentation of Teacher Certificates	0.30
TEC 25.0812 – Earliest Last Day of Instruction	0.28

Notes: Statistics are as of March 2019. At that time, 824 districts had claimed District of Innovation status (81% of 1022 non-charter school districts). Of these, I was able to locate 814 Innovation plans; ten Districts of Innovation are not included in these statistics.

Appendix A. A Comparison of Classifier Performance Used to Identify District of Innovation Plans

Figure A.1



Notes: Figure A.1 displays the F_1-measure (the harmonic mean of precision and recall, equally weighted) of four classifiers – a dictionary classifier, LASSO regression, SVM, and CNN – tested on a random sample of 85 labelled documents and trained on a random sample ranging from 20 to 300 documents. Using the dictionary-based classifier, scraped district documents were classified as a DIP if the document contained the phrase "District of Innovation" and an occurrence of the statute-like regular expression - `\d{2,3}.\d{2,}.` Dictionary classifiers are not trained, but determined by the researcher. For this reason, the F_1-measure of the dictionary classifier is represented in Figure A.1 with a solid line; performance is unrelated to the training sample used by the other classifiers. The penalized logistic regression used a LASSO penalty with an alpha tuned to 6 (chosen through comparing the performance of alpha terms from .01 to 10) and the SVM hyperplane was chosen using a linear kernel. Both the penalized linear model and the SVM model were applied to TF-IDF features and relied on the Python module NLTK for feature extraction and Scikit-learn for model application. The CNN model uses pre-trained word embeddings from spaCy. Note that the dictionary classifier outperformed all three statistical classifiers when those classifiers were trained on a small sample of 40 documents or fewer. Once the training size was increased to 80 documents, the LASSO and CNN classifiers consistently out-performed the simpler dictionary method. The performance of SVM remained below that of the dictionary classifier. These performance rates indicate that statistical classifiers are not always the best choice; a well-tuned dictionary classifier may be better at identifying relevant documents than some out-of-the box statistical classifiers, particularly when the training dataset is small.