Although students are fundamentally impacted by policies made at the school and district level, education researchers have traditionally faced severe data limitations in studying local policy variation. Administrative datasets capture only a fraction of the policy decisions districts make, 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 implementation 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 and 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 NLP techniques. Using 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, the allocation of resources, and the nature of educational programs (Cohen & Spillane, 1992). Even with national policies such as No Child Left Behind and the Every Student Succeeds Act, federal legislation incentivizes states to enact reform, states choose how to respond, and then states commonly pass implementation details to districts (Berman & McLaughlin, 1977; Coburn, Hill, & Spillane, 2016; Cohen & Spillane, 1992; Wong, Wing, Martin, & Krishnamachari, 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, there may be limited information available in administrative datasets that capture the set of decisions districts make in response 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 which 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 locating 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 that are 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, the researcher 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 in a single point in time, the gather-narrow-extract pipeline is useful for maximizing generalizability (i.e., by including the full universe of potential data sources) and replicability (i.e., 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 policy implementation of school entities, as well as for constructing quasi-experimental evaluations of policies.

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 gather-narrow-extract framework for the collection of district-level policy data in order to demonstrate the set of decisions researchers face when collecting policy data using automated techniques. Finally, the paper concludes with a discussion of the strengths and limitations of using 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 which 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 discusses 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, students manuals are not housed at a single known location, instead they 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 which 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 1s and 0s as appropriate. All told, these challenge 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 addressed using a web-crawler which imitates the actions of a researcher following paths of hyperlinks to systematically search a set of websites.
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 like 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 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, the researcher 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 to which category 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 text 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 which discuss the 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 the local policy of each school. 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 which 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 from them structured policy data: she *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 classifiers trained 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 choose to hand-code the documents according to different policies. Regardless, the feasibility of this framework for collecting implementation data rests 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 any of a variety of statistical techniques. Then, the researcher can use the predicted values in subsequent tasks including descriptive or causal analysis (Gentzkow, Kelly, & Taddy, 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 researchers may leverage this in text classification.

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, and 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.

---

1 For a more comprehensive coverage of NLP techniques, readers may turn to Grimmer and Stuart’s (2013) summary of NLP in political science and Gentzkow, Kelly and Taddy’s (2017) summary paper of NLP techniques in economics.
that characterize the text - features ranging in complexity from length and word frequencies to word order and patterns.

**Length.** The simplest feature describing a text is generally its length - 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, Laliberté, and Oreopoulos found, for example, that students who write longer responses to survey questions about goals tend to have a higher college GPA than their peers (Beattie, Laliberté, & Oreopoulos, 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 was applied by Bettinger and team (2015) in their analysis of the effect of online course interactions on students’ academic success. The researchers created a dictionary of student names and looped through every post in the forum to identify whether the post referred to another student by name. The occurrence of a peer’s name was 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 to identify policies (for example, searching student manuals for the term *uniform*), though such choices should be tested for validity.

**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, \( W_1 \ldots W_N \), 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
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\(^2\).

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, Raghavan, & Schütze, 2008). Lemmatization
also removes inflectional endings, returns the root form of a word, known as the lemma
(Manning et al., 2008). 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-perform and under-perform expectations (based on high-school GPA), Beattie,

\(^2\) For example, Python’s Natural Language Toolkit (NLTK) maintains a list of 179 stop words (Bird et al., 2018).
Laliberté and Oreopoulos (Beattie et al., 2018) found 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-performers 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.

**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:

\[ tfidf_{t,d} = tf_{t,d} \cdot log \frac{N}{df_t}, \]

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 when the term occurs fewer times in a document or in many documents, and lowest when it occurs a small number of times in almost
every document. Tf-idf is often used to generate distinct topics within a corpus of text, a task referred to as **topic modelling**.

**Context-dependent features.** All of the previous features do not make any effort to consider the context of a token’s occurrence. They assume documents can be represented as a **bag of words** where word order does not provide information on a document’s content. If this assumption is unpalatable, a researcher can choose to retain some word order using **bigrams** (token pairs) or **trigrams** (token triples), allowing the researcher to distinguish the bigram *school uniform* from the unigram *uniform*, a word which may mean invariable rather than an item of clothing.

Finally, to retain some of the semantic meaning of a token, 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, Chen, Corrado, & Dean, 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 and team (2018) use these more nuanced features to code the authenticity of teacher’s question-asking behavior from transcribed recordings (Kelly, Olney, Donnelly, Nystrand, & D’Mello, 2018; Olney, Samei, Donnelly, & D’mello, 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 text, their relationship to a document’s type or topic is determined by a classifier. 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. Bettinger’s (2015) dictionary of student names 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) were classified as indicative of peer interactions while those without a dictionary occurrence were 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 et al. (2013) argue that a key principal of text analysis is “validate, validate, validate” (p. 5) and that Bettinger and team (2015) took the time verify that student names are indicative of response forum posts.
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 are used to 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 text for validation. In the testing phase, the model’s predictive capability is tested on previously-unseen data, and a researcher can optimize a classifier by iteratively comparing different specifications on their test 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 including word frequencies – the number of words likely far outnumbers the number of documents, drastically reducing statistical power. In order to effectively use logistic regression, a researcher will either need to 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, Tibshirani, & Friedman, 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).

One of the newest and most complex classifiers is **convolutional neural networks** (CNN). Unlike the previously mentioned models, CNN’s are capable of taking account of a token’s location in the 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, Divvala, Girshick, & Farhadi, 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 both provides high performance on classification tasks and removes 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, CNN’s 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).

**Choosing Among Classifiers**

A researcher’s choice of classifier will be heavily dependent on the task at hand. Dictionary-based methods require prior information of the function mapping features and classes and is therefore most appropriate when prior information on the 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 willing and able to create a set of labelled training documents. Penalized linear models and SVM offer efficient methods of classifying texts using word frequencies, document-term matrices, and tf-idf weighting. If the researcher does not need
to be able to interpret the classification function, then they can turn to CNNs as an easy-to-implement, high-performing, and context-sensitive approach to classification. Appendix A provides an example comparison of classifier performance applied to district documents.

**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*, which grants public schools the ability to exempt the majority of the state educations, including regulations 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 potential to dramatically change the day-to-day operations of many schools in the state.

The Texas Districts of Innovation law provides an ideal context to demonstrate 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 amount of time such a task would take inhibits a timely description of important events, and, because districts may declare District of Innovation status and amend their documents at any time, the dataset may already be out-of-date by the time a researcher completes hand-coding exempted laws. Second, DIPs are an exemplar of a number of 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 a 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 a researcher faces as they apply gather-narrow-extract for data collection, and Appendix B contains the list of open-source Python modules I used to implement each step.

<Insert Figures 1, 2 here.>

**Step 1: Gather Potentially Relevant Documents using a Web-crawler**

Like many states, Texas Education Agency maintains a list of school district websites and, helpfully, the state also maintains a list of URLs for Districts of Innovation. These links rarely led directly to a DIP, but rather 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 build a web-crawler to search each district website for links to documents. While developing the web-crawler, I iteratively tested it on a small number of links, 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.

The web-crawler followed a loop in which it (1) visited each URL in the list of school district websites (these starting links are termed seed links); (2) copied every URL that linked to a static document (indicated by the extensions .PDF, .doc, or .docx, and the strings,
“drive.google.com” and “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, termed a seed link\(^3\). From this list of URLs, I used Apache Tika to scrape the raw text from each HTML page, PDF, Word Document, and Google Doc (C. A. Mattmann & Zitting, 2011). The final result of the web-crawler was a dataset of district names, URLs which may contain DIPs, and extracted plain text. Figure 3 displays a snippet of this dataset.

<Insert Figure 3 here.>

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

I chose to implement a CNN text classifier in order to identify DIPs\(^4\), a technique I selected due to both its high-performance on text classification tasks and minimum required pre-processing. Though CNNs are more complex than simpler classification methods like penalized regression models, they are as easy to implement as their simpler counterparts. The tradeoff in classifier choice is not between performance and implementation difficulty, but rather performance and interpretability, as the highest-performing classifiers do not provide a function

---

\(^3\) A number of software libraries are available to simplify the process of building a web-crawler, in particular modules for making HTTP requests, parsing HTML, and extracting text from PDFs and other media. Appendix A provides the libraries I used.

\(^4\) Appendix B displays a comparison of classifiers applied to identifying relevant documents. The graph demonstrates the high-performance of the CNN classifier, even with a training dataset size of less than 100 documents.
between text features and class membership probability. This is not necessarily a disadvantage when applied to the document collection process - I did not care to know the text features of a DIP, but strictly that my classifier was able to identify them.

As a supervised learning method, CNNs learn the function between input text and output classification from a set of labeled training data - here, text labeled as true if the document is a DIP or false if it is irrelevant. I began by labelling a random sample of 225 plain text documents and split this dataset into training and testing data. Training data were held in a two-column dataset where the first column contained plain text and the second column contained the true/false label. From the training text, the CNN created a function predicting DIP status using a set of pre-trained word embeddings and convolutional filters. The output of a CNN is a probability of category membership for each input and so, for each district, I classified the document with the highest probability of DIP membership as a positive identification and negatively-classified all others.

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 88% of true positives (a measure of recall), and of the documents identified as DIPs, 99% were true positives (a measure of precision). These are respectable performance rates which could be improved by increasing the size of the training data (see Appendix B). However, when false positives can be identified and manually collected, the automated collection of documents can be most efficient when combined with manual collection of edge cases – strange scenarios that only apply to a few documents.

---

5 In order to store text and labels in a flat file, I used a special character not seen in the set of documents as a delimiter - the Icelandic thorn: Þ.
I was able to identify false positives when positively-classified documents did not contain any mention of regulations. For example, if a school calendar were inappropriately classified as a DIP, I would not be able to find any regulatory exemptions for that District of Innovation and conclude the document is not likely to be DIP. Since the classification model effectively ranked all collected documents for a district by their likelihood of being a DIP, this situation was easily addressed by manually checking remaining district documents in descending order of DIP likelihood.

**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 DIP indicate the district is exempting?*

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. You can see this demonstrated in the example DIP displayed in Figure 4. This statute-like pattern is represented easily by a regular expression\(^6\), which then acts as a dictionary of search terms. 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 from the statute mentioned. As in all classification problems, my classification rule can and should be tested using documents set aside for validation. In a reserved set of 25

---

\(^6\) 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. For more information on regular expressions, I recommend *Mastering Regular Expressions* (Friedl, 2002).
DIPs, each occurrence of a top-ten most frequent dictionary term was a true exemption\textsuperscript{7}.

If any document did not contain a statute-like pattern, I checked that this document was a true DIP and, if it was not, I manually searched for the correct document (among documents from its district collected by the web-crawler but discarded by the classifier), added it to the dataset, and repeated the data extraction process. In this way, I was able to capture the exempted laws of all Districts of Innovation with significantly less effort than would be required by manually identifying and encoding each document. The output of this routine was a district-by-rule dataset with an indicator for whether each district exempted each rule.

<Insert Figure 4 here.>

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. Table 1 displays the top ten most commonly exempted regulations. These exemptions fall into four categories: class schedules, class sizes, certification requirements, and employee contracts. The most commonly exempted regulations concern school schedules - districts commonly exempt the statute requiring that they not begin instruction before the fourth Monday in August (92% of Districts of Innovation), the statute requiring that schools operate for at least 75,600 minutes (40%), and the requirement that the

\textsuperscript{7} Outside of the most frequent statute-like patterns, this was not universally the case. Some documents included a series of numbers that did not refer to any kind of statute, and some documents referred to a statute they were not planning to exempt. Because a search term’s context can be important for precise classification, I applied a CNN to sentences containing potential references to regulation exemptions. Tested on a random sample of 12 documents labeled with their exemptions (71 potential statute references in total), pattern-matching identified true exempted laws 100% of the time, but 14% of matched patterns were false positives. When employing a classifier on these patterns, the routine identified 92% of true exemptions and reduced the rate of false positives to 8% on the test set. However, among the ten most-commonly exempted regulations, test set precision and recall were each perfect without a classifier (100%), so I deferred to using regular expressions to extract exemptions from documents when summarizing those exemptions.
school 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 (43%) and that schools inform parents when class sizes exceed 22 students (36%). Finally, district commonly exempt regulations surrounding hiring and employee contracts, including the requirement that teachers be certified (78%) and tenured after a few years of service (40%) and the requirement that teacher contracts be for a minimum of 187 days (34%).

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. Without this school-level dataset, we would not know how districts are using regulatory flexibility, only that they have claimed District of Innovation status. NLP techniques give researchers the ability to get inside the black box of policy implementation, allowing for a 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 a one-size-fits-all solution 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 where content is changing frequently, and the researcher is interested in studying changes over time). Figures 5 and 6 present the criteria a researcher should consider in deciding whether to automate document collection and/or processing. Criteria are presented on a continuum because, broadly speaking, these decisions involve balancing the complexity of a task with the amount of data that needs to be collected.

Second, documents collected and processed through web-scraping and text classification are observational data 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, however, match school districts on characteristics found in administrative data in a non-equivalent control group design, 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, Cook, & Cambell, 2002), they can yield results comparable to experiments under certain conditions (St.Clair, Hallberg, & Cook, 2016; Wong, Valentine, & Miller-Bains, 2017).

Third, as with all research, the inferences drawn from scraped documents are limited by the specific sample that is 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 the 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 is correlated with other constructs of interest. When faced with a truncated sample of scrapable data, Landers and team (2016) recommend that the researcher develop and test a data source theory regarding the origin of the online data and the types of policy-makers 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 school 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. 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 Landers and team’s recommendation 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 (2016). 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, too, can make use of this information to describe changes in local policies quickly, accurately, and cost-effectively. In the past, when policymakers have introduced a policy that is anticipated to have a meaningful impact on students, evaluative research has lagged behind, assessing the effect of the policy long after the policy has passed—and sometimes, after the policy has been revised. Policymakers need strong and timely evidence for decision-making, and the gather-narrow-extract approach provides a method for assisting researchers in meeting this
need.
Bibliography


http://docs.python-requests.org/en/master/


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


https://doi.org/10.3102/0013189X17743230
Figure 1: Steps for automated document collection.

1. Collect 5 to 10 documents for informal testing.
2. Collect list of seed URLs. Build and apply web crawler.
3. Have you captured the links to all of your test documents?
   - Yes: Label random sample of documents as relevant or irrelevant. Split into training and validation sets.
   - No: Try:
     - Revise code.
     - Alter list of seed websites.
   - Try:
     - Increase the size of your training sets.
     - Increase the complexity of features or the classifier.
4. Train and apply classifier.
5. Is classifier performance satisfactory?
   - Yes: Repeat as necessary to update data.
   - No: Try:

Figure 2: Steps for automated data collection from text.

1. Examine documents to determine patterns.
2. Transform documents into plain text.
3. Is there more irrelevant than relevant text within each document?
   - Yes: Label random sample of documents. Split into training and validation.
   - No: Train and apply classifier.
4. Is classifier performance satisfactory?
   - Yes: Repeat as necessary to update data.
   - No: Try:
     - Increase the size of your training and validation test set.
     - Increase the complexity of features or the classifier.
5. Consider:
   - Break document into units (whether pages, paragraphs, sentences, or phrases) and train a classifier for relevance.
   - Use key words or regular expressions to point your classifier to the most relevant spans of text.
Figure 3. Snippet of Data after Web-crawling

<table>
<thead>
<tr>
<th>District</th>
<th>Link</th>
<th>Text Snippet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbott ISD</td>
<td><a href="https://1.cdn.edl.io/GBV7M8bdP47WCfN702gzyy375OT1XxiuNrmDB6rbG117TVdy.pdf">https://1.cdn.edl.io/GBV7M8bdP47WCfN702gzyy375OT1XxiuNrmDB6rbG117TVdy.pdf</a></td>
<td>2015-2016 Texas Minimum State Vaccine Requirements for Students Grades K-12 This chart summarizes the vaccine requirements incorporated in the Texas Administrative Code (TAC)</td>
</tr>
<tr>
<td>Abbott ISD</td>
<td><a href="https://1.cdn.edl.io/8rtuLbNL4QfStkYrXK0eUh9uDzTDoddvoVTPiVnuBJSqix2.pdf">https://1.cdn.edl.io/8rtuLbNL4QfStkYrXK0eUh9uDzTDoddvoVTPiVnuBJSqix2.pdf</a></td>
<td>2010-2011 Texas Minimum State Vaccine Requirements for Students Requisitos de vacunas mínimos estatales de Texas de 2015-2016 para estudiantes de kind</td>
</tr>
</tbody>
</table>

Figure 4. Extract from Sample District of Innovation Plan (Hamilton ISD, 2017)

**First Day of Instruction - Board Goal 4 Student’s Instructional Needs Exemption From: TEC §25.0811, EB(LEGAL), EB(LOCAL)**

*A district may not begin instruction for students for a school year before the fourth Monday in August. A district may not receive a waiver of this requirement.*

**Proposal:**

This flexibility of a start date allows the district to determine locally, on an annual basis, what best meets the needs of the students and local community. This flexibility of the start date also offers the following opportunities:

- a) This will allow the first and second semesters to be somewhat equal in the number of days of instruction.
- b) Students participating in dual enrollment opportunities will work with balanced semesters, which align with our local colleges.
- c) An early start date permits students an additional week of instruction prior to state assessments in December.
- d) This will allow the district time to provide proper remediation to students in summer school.
- e) Students will be afforded opportunities to enroll in summer college sessions with finalized official transcripts and staff will be able to attend summer school classes as well.
- f) This would allow an option to start school with a shorter week, easing the transition back to school for all students.
<table>
<thead>
<tr>
<th>Statute</th>
<th>Proportion of Districts of Innovation Exempting</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEC 25.0811 – First Day of Instruction</td>
<td>0.90</td>
</tr>
<tr>
<td>TEC 21.003 – Certification Required</td>
<td>0.78</td>
</tr>
<tr>
<td>TEC 25.112 – Class Size</td>
<td>0.43</td>
</tr>
<tr>
<td>TEC 25.081 – Operations of Schools</td>
<td>0.40</td>
</tr>
<tr>
<td>TEC 21.102 – Probationary Contract</td>
<td>0.39</td>
</tr>
<tr>
<td>TEC 25.113 – Notice of Class Size</td>
<td>0.36</td>
</tr>
<tr>
<td>TEC 21.401 - Minimum Service Required</td>
<td>0.34</td>
</tr>
<tr>
<td>TEC 21.057 – Parental Notification (of uncertified teacher)</td>
<td>0.30</td>
</tr>
<tr>
<td>TEC 25.0812 – Last Day of School</td>
<td>0.28</td>
</tr>
<tr>
<td>TEC 25.092 - Minimum Attendance for Class Credit or Final Grade</td>
<td>0.26</td>
</tr>
</tbody>
</table>

*Notes: Statistics are as of March, 2019.*
Figure 5. Criteria for consideration when deciding whether to automate document collection.

- Strong seed links are difficult to locate
- Documents are few in number
- Text content will not change

- List of strong seed links is easily accessible
- There is a large number of documents
- Text content evolves frequently
Figure 6. Criteria for consideration when deciding whether to automate data extraction from text.

- There is insight to be gleaned by reading documents closely
- There is no clear pattern to how information is stored in the text
- Documents are few in number
- Text content will not change

- Encoding would be tedious and therefore error-prone
- There is a clear pattern to how information is stored in the text
- There is a large number of documents
- Text content evolves frequently
Appendix A. A Comparison of Classifier Performance Used to Identify District of Innovation Plans

Notes: Using the dictionary-based classifier, scraped district documents were classified as a District Innovation Plan if the document contained the phrase “District of Innovation” and an occurrence of the statute-like regular expression - \d{2,3}.\d{2,}. The penalized linear model used a LASSO penalty with an alpha tuned to 1 (chosen through comparing the performance of alpha terms from .01 to 1) and the SVM hyperplane was chosen using a linear kernel. Both the penalized linear model and the SVM model are applied to TF-IDF features and rely on the Python module NLTK for feature extraction and Scikit-learn for model application. The CNN model uses pre-trained word embeddings from spaCy. The y-axis displays the model F-measure (the harmonic mean of precision and recall) of each classifier tested on a random sample of 85 labeled scraped documents.
Appendix B: Recommended Python Modules

<table>
<thead>
<tr>
<th>Programming Task</th>
<th>Module Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit HTTP Requests</td>
<td>Requests (Reitz, 2018)</td>
</tr>
<tr>
<td>Parse HTML</td>
<td>BeautifulSoup (Richardson, 2017)</td>
</tr>
<tr>
<td>Extract Text from PDFs and other Media</td>
<td>Tika-Python (C. Mattmann, 2018)</td>
</tr>
<tr>
<td>Feature Extraction and Text Classification</td>
<td>spaCy (Honnibal, 2017)</td>
</tr>
<tr>
<td></td>
<td>NLTK (Bird et al., 2018)</td>
</tr>
<tr>
<td></td>
<td>Scikit-learn (Pedregosa Fabian et al., 2011)</td>
</tr>
</tbody>
</table>