Using Deep and Active Learning Classifiers to Identify Congressional Delegation to Administrative Agencies*

Joshua Y. Lerner[1] and Gregory Spell[2]

[1]Department of Political Science, Duke University
[2]Department of Electrical and Computer Engineering, Duke University

July 29, 2020

Abstract

Congressional oversight of the federal bureaucracy remains key to understanding implementation of major laws. Essential to this are theories of how and why Congress delegates powers to administrative agencies, which posit a complex relationship between individual members of Congress and the agencies they oversee. This field of study is rich in theory but has been lacking in systematic large-scale empirics. However, given advances in the use of text-as-data methods, we believe we can begin to test the conditions both institutional and partisan under which Congress delegates authority to administrative agencies. Using a convolutional neural network on the text of bills, we classify bill sections by their role in delegating authority to administrative agencies. We introduce an active learning approach to text classification, applying an iteratively improving coding scheme that enhances existing supervised learning approaches. This method will allow us to study the statutory scope of administrative agencies systematically and provides a first-of-its-kind dataset to study how administrative law develops. We will also be able to examine the conditions under which these agencies have their authority amended and how changes in legislative institutions impact patterns of delegation in Congress.

*Prepared for the American Political Science Association's Annual Meeting in Boston, MA, August 29th-September 2nd 2018

Introduction

"Ideology", "power", "democracy", "authority," and "agency", among others, are all examples of latent concepts that are critical to theories of politics and society but are inherently unobservable. To someone well versed in the literature on these issues, however, identifying particular cases may not be intrinsically difficult: measuring latent constructs directly might be a challenge, but making simple classifications or comparisons would not require a Herculean effort. However, until recently, utilizing expertise that is intrinsic to the deep study of politics (or any field with such latent concepts) has required either limiting of the scope of analysis, such that using this expertise can be done directly, or operationalizing this expertise in such a way that allows for broad generalizations but ultimately requires major conceptual compromises. For example, reducing "power" to centrality in a co-sponsorship network is a dramatic shift in measuring the underlying latent concept and requires a far-reaching metaphor for how the data and method relates to the concept.

What if the expert evaluation of these underlying concepts could be scaled up and applied to broader swaths of data? What if researchers no longer have to rely on complex metaphorical relationships between abstract concepts and real data, but can rely on the validity of expert coders *and* not require a massive data collection operation? Fortunately, computer scientists and statisticians have invested heavily in developing machine learning algorithms that can predict qualitative labels from readily-available quantifiable features, such as document word counts. After being trained on a set of hand-labeled ads, the learning algorithm can generate predicted labels for a much larger data set and can reduce greatly the up-front costs of setting up a human-run labeling project. However, most machine learning operations separate the hand-coding from the evaluation process and treat it merely as a single attempt at attaining accuracy. Complex concepts will be hard to translate into a simple coding scheme, and thus machine coding done this way will generally fail. In this paper, we

argue that researchers can effectively combine their expertise through an interactive machine-learning framework that will, in essence, learn on the go. More commonly referred to as an "active learning" approach to machine learning, we argue that this framework best combines the portability and flexibility of machine learning with the expert evaluation usually used in more modestly-defined approaches, such as case studies.

To demonstrate the utility of this framework, we tackle a canonical problem in political science, on how and when Congress delegates authority to administrative agencies. We generate new data on legislative delegation, and establish how our approach enables the development of new and exciting datasets. We address the delegation question for three reasons. First, the federal bureaucracy is essential to both economic growth and the maintenance of an orderly society; they facilitate the implementation of laws written by Congress and represent the most direct way any average citizen will interact with the federal government.

Second, theories of delegation abound in political science, economics, and public administration, but there have been few empirical tests of these theories. There are numerous competing, highly comprehensive theories about how, why, and when Congress delegates authority, but up until now, most studies either focused only on a single bit of policy (Balla 1998; Huber and Shippan 2002), a handful of "significant" bills (Epstein and O'Halloran 1999), or forewent empirical analysis altogether and focused on model building (see Gailmard and Patty 2012 for a comprehensive overview of such models). Having a measure that is both broad and comparable between domains should give us an opportunity to evaluate the efficacy of these theories and should allow us to generate new research into the relationship between Congress and the bureaucracy.

The final reason we chose this domain is that it is a question that has an easily identifiable textual component but requires enough expertise to make large-scale coding more complex. Identifying agency delegation by Congress is exactly the kind of problem where explicitly

scaling up the hand coding is both impractical and undesirable.

This paper proceeds as follows: first, we discuss the history of automated machine labeling of texts in political science. Second, we discuss our approach—active learning —and why it improves upon existing approaches (and we provide a baseline for comparison). Third, we address the problem we will be using our method to study—the delegation of authority from Congress to administrative agencies. And finally, we will discuss the results of our active learning model, how it improves upon existing models, and what can be done with our newly labeled dataset.

Automated Machine Labeling for Texts in Political Science

We use our language to describe all of our concepts, including political and legal ones. It is not surprising, then, that the study of political language has expanded tremendously over the last decade. With the increasing capabilities of text-as-data approaches, and with the proliferation of easy to use methods in both *R* and *Python*, research into text-as-data methods have grown exponentially. Grimmer and Stewart (2013) first, and Wilkerson and Casas (2017) later, discuss numerous applications of these methods throughout political science and demonstrate that we are just scratching the surface when it comes to long-term applicability.

Classification is, not surprisingly, one of the most popular objectives in of all text-as-data work. Unsupervised machine learning methods (e.g., K-means, principal components analysis (PCA), latent semantic analysis, latent Dirichlet allocation) compare the similarity of documents based on co-occurring words, often with weights given for relative infrequency. Despite their name, unsupervised methods require extensive input from the user, who must (among other things) specify the number of clusters or topics before running the model and

interpret their meaning. In one of the earliest applications by political scientists, Quinn et al. (2010) used a simple topic model, a form of unsupervised learning, to classify Senate speeches by policy topic. Quinn et al. (2010) then validated their results by showing that their topics were similar to those developed using more time-consuming methods. Grimmer and King (2011) demonstrate how unsupervised methods can lead to new discoveries. They find that congressional press releases cluster in ways that match Mayhews (1974) typology of constituent advertising, position taking, and credit claiming, but they also observe an additional cluster they label "partisan taunting" (see also Grimmer 2013). Roberts et al. (2014) show how incorporating non-textual information about documents into topic models can aid in the interpretation of open-ended survey responses.

Whereas unsupervised methods are frequently used for discovery, supervised learning methods primarily serve as a labor-saving device for classification. With supervised learning, the learning algorithm is presented with a set of inputs along with their desired outputs (also called labels). The goal in supervised learning applications is to discover some underlying logic that enables the program to map the input to output. For example, Collingwood and Wilkerson (2011) use supervised methods to apply the widely used Policy Agendas topic-coding scheme to new research domains, and discuss the successes and failures of the extension. Minhas, Ulfelder, and Ward (2015) use the text of Freedom House and State Department memos to generate annual measures of national political regime types. The fact that supervised methods often require thousands of training examples make them a non-starter for many researchers and projects. However, there are often creative ways to reduce the effort required. Examining 250,000 Enron emails, Drutman and Hopkins (2013) use simple identification techniques first to exclude the 99% that were not political. Crowd-sourcing is also frequently used to build training sets in computer science. When a project does not require individual document labels, *ReadMe* is a supervised method that reliably predicts class proportions using a much smaller number of training examples (Hopkins and

King 2010). King et al. (2013) use *ReadMe* to classify millions of social media posts by topic in a study of government censorship in China.

There is a vast literature in political science of using text-as-data methods to learn about important, otherwise unobservable concepts, using classification or clustering techniques. Our approach can be thought of as a way to address two primary concerns: accuracy/reliability of classifications and an overall reduction in the cost of inputs made in supervised learning environments. We argue that our approach allows for supervisory interaction with outputs from the model, gives a role for expert driven coding, and allows for better machine-human synergy. It also makes model evaluation part of the hand coding process instead of a separate procedure; borrowing information to improve hand-coding for better investigator-coordinated supervised learning. Given the traditional ways in which supervised methods have been used in political science, we view our approach as a natural extension and something that can further reduce computational costs to ambitious projects.

An Active Learning Convolutional Neural Network

Active learning (sometimes called "query learning" in computer science) is a subfield of machine learning. The fundamental idea behind active learning as an approach is that a machine learning algorithm can achieve higher accuracy with fewer training labels if it is allowed to choose the data from which it learns. If the learning algorithm is allowed to choose the data from which it learns—to be "curious"—it will perform better with less training. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by a human annotator. Active learning is well-motivated in many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain (Settles 2012).

Why is this a desirable property? Consider that, for any supervised learning situation,

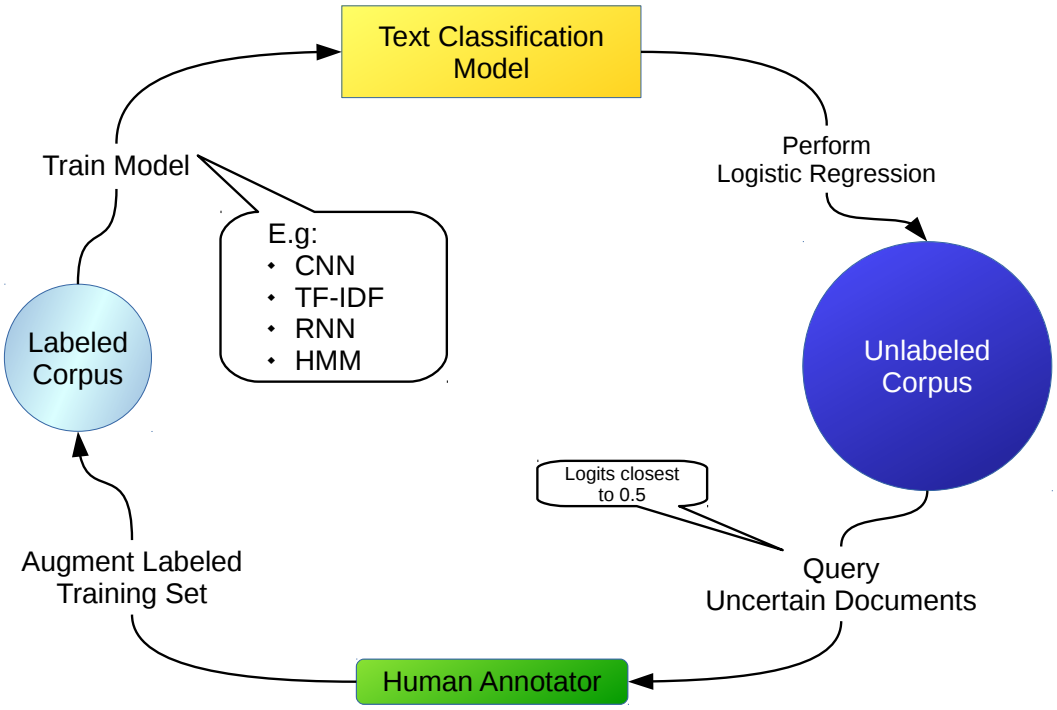
the model must often be trained on an extensive selection of labeled observations. Sometimes these labels come at little or no cost: identifying "spam" in unwanted emails, online rating systems for movies or restaurants, or using Policy Agenda labels on bills that are not yet classified (like in Collinswood and Wilkerson 2012). In these cases, labels cost virtually nothing, but for most other supervised learning tasks where data creation is the goal, increasing the total number of labeled instances can be challenging, time-consuming, expensive to obtain, or all of the above.

Complex tasks can also often require more information than traditional machine learning approaches allow for. Researchers may also need to have models learn-on-the-fly not just in one wave. A potential solution to this problem would be active learning. A motivating example from our research in identifying delegation to administrative agencies is that the general framework for delegating authority is relatively straightforward. Some agency is given a task (often told it "shall" or "must" do something) within a single sentence. A standard learner would probably identify the agency or program named and the key verbs specifying the task and be able to use that combined information to identify most instances of delegation easily. The problem is there are hundreds of currently active agencies and programs, some of which have unusual names (the "Corporation for National and Community Service" is an example of such an agency), that would reduce the likelihood of delegation being properly identified by the algorithm.[1] We ran into this problem early on when discussions of the "Attorney General" were always mislabeled; because, up until that point, there were no observations in our training data where a cabinet level secretary was named as anything but "secretary." Because the specific words "Attorney General" have few analogous positions in other departments, it would have to be specifically hand coded for the model to learn what that is.[2] Now, in general, this is not a complicated fix: hand label some of these aberrant

[1] Unless that agency also showed up in the training data which, given that there are hundreds of agencies, is likely to miss many.

[2] As opposed most other cabinet level positions. Learning from the title "Secretary" probably matters

Figure 1: Visualization of Active Learning Models



observations, and it should solve the problem. However, what we could not know *a priori* was what exact issues were going to appear: moving to an interactive labeling and machine-learning framework alleviated these concerns because the machines would be able to tell us exactly where it was having difficulties making these classifications. Learning on the go was the preferred option because, though our classification scheme is simple, there are enough moving parts that it is hard to know precisely what problems would have arisen before we started the coding.

Here is how an active learning approach differs from a traditional machine learning problem, detailed in Figure 1. We take our standard machine learning data division, where we split our already classified data into a training and test set. We take a supervised classification model (for our project we used a convolutional neural network (CNN) detailed later on,

more to classification than whatever specific agency "Secretary" is modifying

and a support vector machine (SVM), but any classification method would work well here) and train it on the training data and test it on the test data. Once the model performs well enough, we move it to an unlabeled selection. So far this is a supervised learning procedure.

In a typical supervised learning environment, this is where the process would end. It would classify the unlabeled set, and the researchers would then evaluate whether or not the model performed well enough on its own. If it did not do a good enough job, the usual response would be to label more documents. In our case, we assume the model performs sub-optimally and, instead of evaluating the labels at this stage, we only ask it to give us the documents it was uncertain in labeling. Once we have those uncertainly labeled documents (and there are many different ways of evaluating which documents the learner is uncertain about), we then label them ourselves. We take these now labeled documents and include them in the original training set, and the process starts again. Only after running this a few times (which depends on out-of-sample model performance and how many documents you are willing to label) do you have enough power to take the new labels and use them as complete classifications.

There are many different approaches to identifying the uncertain observations set aside for human coding in active learning. The most commonly used, and most straightforward to implement, is what is called "uncertainty sampling"– query the instance with least confident prediction, the smallest margin between most probable classes, or the greatest entropy. For binary classification, these reduce to the same strategy: which is to identify observations closest to the prediction threshold (a Logit of 0.5). The other commonly used querying strategies are ensemble approaches (where you identify observations where there is disagreement on classification between multiple different methods) and expected model change approaches (where you select the instance that would impact the most significant change to the current model if we knew its label) (Settles 2012). We went with uncertainty sampling because it is the most widely used, is the most computationally straightforward, and remains very flexible

in making comparisons between various methods (Tong and Koller 2001; Settles 2012).

For this paper, we use a convolutional neural network (CNN) as our primary classifier and, as a point of comparison, a support vector machine (SVM), but one could easily replace the training model with any supervised learner, whether that is a LASSO, Random Forest, Bayesian Additive Regression Tree, or whichever model the researcher prefers. The key to active learning is querying documents in the unlabeled set that the model has difficulty in classifying. For our cases, we used the estimated probabilities closest to 0.5, which is the observations that were the hardest to classify as to whether Congress had delegated authority or not.

Convolutional Neural Network Model for Text Classification

We now turn to the specific type of supervised learning model we will use as the basis for the active learning module discussed above: a convolutional neural network model for text classification. A neural network is a type of machine learning algorithm designed solve problems in a similar way to the human brain. Normal machine learning operations follow a static approach to learning, where the algorithm follows a set of instructions in order to solve a problem. In contrast, neural networks learn by *example* rather than being programmed to perform a specific task. Technically, they are composed of a large number of highly interconnected processing elements (nodes) that work together to solve a specific problem, similar to how the human brain works (Taylor and Koning 2017). For our model, we follow the example of Kim (2014) who showed how to define a simple convolutional neural network model designed for text classification.

For the fitting of most supervised models, documents must be reduced to some word-frequency representation. Traditionally, the workflow would be to engage in conventional preprocessing: removing capitalization, punctuation, numbers, symbols and so-called "stop words", which are words that convey little substantive meaning but serve to facilitate lan-

guage transmission (think conjunctions, prepositions, and pronouns for example). The second step would be to discard the order in which words occur singularly (or in pairs or triplets) in documents (Grimmer and Stewart 2013). This procedure takes documents and converts them to a "bag-of-words", where the order would not inform the analyses. Other considerations can be made, but this is the general first set of steps in turning text into something analyzable.[3] These approaches generally use tf-idf (term-frequency inverse-document-frequency) weights, which are represented as indices in a vocabulary. This representation, however, severely limits the modeling power of the learned features of the documents because the similarity between words is not modeled. An improvement on this would be to create a distributed representations of words, in which each word in a vocabulary is represented as a real-valued vector that captures both semantic and syntactic information (Mikolov et al. 2013).

Whereas the bag-of-words assumption underpinned the use of tf-idf features, the a distributional approach to language – which suggests that words with similar meanings often occur in the same contexts – motivates word embeddings, an approach essential to using "deep learning" methods of textual analysis (Kim 2014). Neural language models learn word embeddings through training which involves predicting a target word given the context (surrounding words) of that word. In fact, the Skip-gram model–the model most commonly associated with deep learning applications– does the reverse: the model accepts a word as input and seeks to predict the surrounding words, an approach taken for computationally efficient learning. The Skip-gram model comprises the basis of Google's *word2vec* software, which we used to learn the cardinal embeddings to fit our CNN.

Word embeddings are the result of methods designed to learn the features of words by their relative concurrence with other words. As a series of techniques, word embeddings

[3]See Denny and Spirling 2018 for a discussion of the various decisions that are often made at this stage and how they effect outcomes.

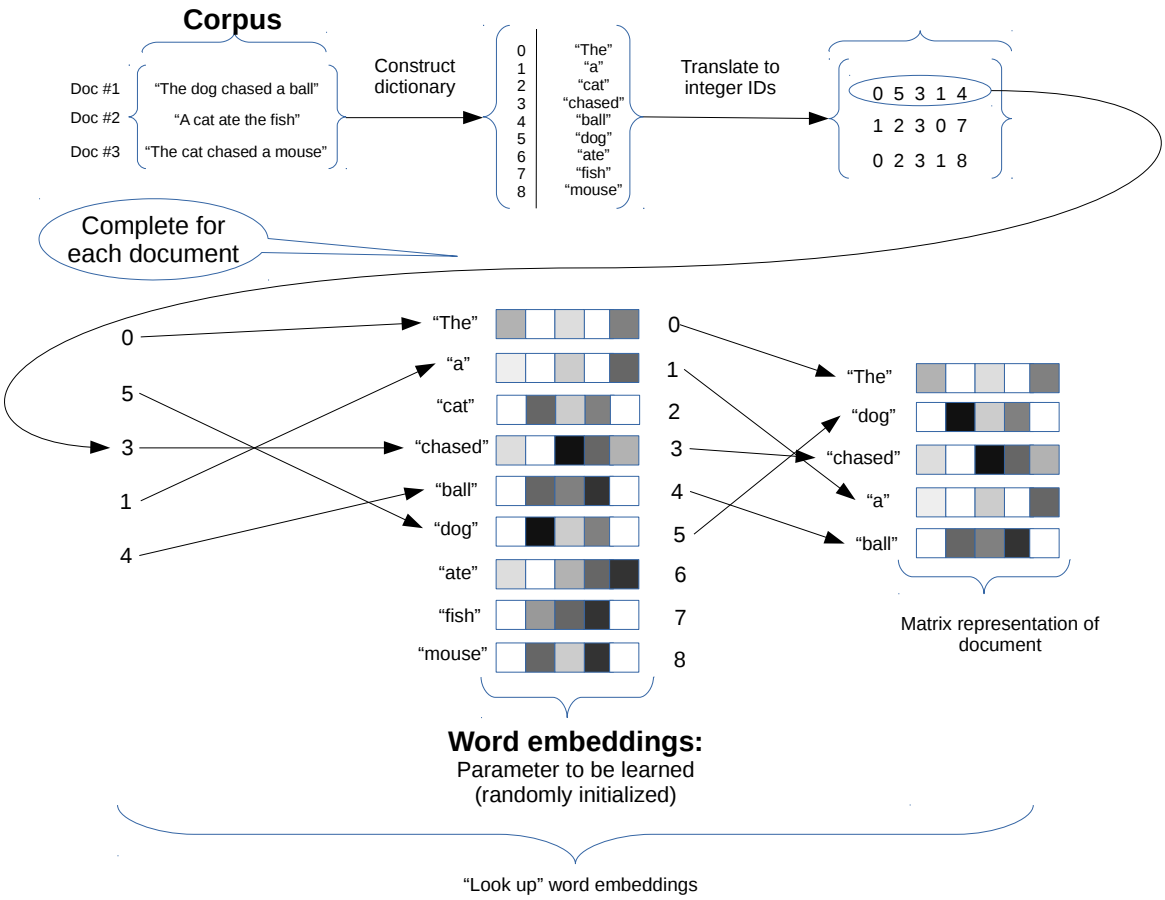


Figure 2: Visualization of Document Representation in CNN Model

represent a move towards robust unsupervised methods that aim to discover the actual architecture of word patterns in text. Most modern word-embeddings schemes build off of *word2vec*, a model developed by Mikolov et al. (2013) for Google, which took existing word embeddings models and created a computationally efficient approach to reducing the complexity of texts in meaningful and tractable ways. All word embeddings approaches take words from a vocabulary as their input and embeds them as numerical vectors into a lower dimensional space.[4] This process, in turn, generates vector values for each word in relationship to all the other words. These weights are designed to encode general semantic relationships between words, and to create a mapping of word cooccurrence and frequency.[5] In the word embeddings framework, each word is described as a vector in n-dimensional space that represents the whole vocabulary space. Since each word occupies its own dimension, the algorithm can place any word in the exact space it occupies alongside all other words, creating a highly-multidimensional vectorized representation of a corpus vocabulary. When thinking about word embeddings, it is oft wise to think of the saying that "a word is characterized by the company it keeps" (Firth 1968).

As mentioned previously, the Skip-gram model is trained by predicting the words surrounding a given word. Suppose that by "surrounding" words, we mean words within c words of a center word w_t. For a sequence of words $w_1, w_2, \dots w_T$, the Skip-gram model seeks to maximize the average log-probability of words within a context window c of each word w_t:

$$\text{maximize } \mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} \mid w_t) \tag{1}$$

We will refer to the quantity $\mathcal{L}$ as the Skip-gram objective. The probabilities $p(w_{t+j} \mid w_t)$

[4]Which it then fine-tunes through back-propagation. Increased efficiency of back-propagation in recent years has allowed for dramatic increase in utility of these models.

[5]The simplest version of these weights would be just to measure the frequency of each word, or pairs of words, or each word triplet in a corpus.

of a context word w_{t+j} given a word w_t are defined through the softmax function, a multi-dimensional extension of the logistic function:

$$p(w_O \mid w_I) = \frac{\exp\left({v'_{w_O}}^{\intercal} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\intercal} v_{w_I}\right)} \tag{2}$$

where W is the number of words in the vocabulary of the corpus. Note that a word w has a different vector representation, v_w, when it is an "input" word than v'_w when it is an "output" (context) word (meaning words going into the model— the"input' words— and words being predicted by the m). The word embeddings are learned by finding the vectors which maximize $\mathcal{L}$, a process accomplished by gradient ascent. Training such a model directly requires computation of the gradient $\nabla \log\, p(w_O \mid w_I)$ which is proportional to the vocabulary size (W). Obtaining descriptive vectors, however, requires large training data which typically has a large vocabulary, thus rendering this computationally prohibitive. Given this limitation, most approaches use pre-trained word embeddings provided with *word2vec* that learn their relations from the *Wikipedia* article corpus. Given that we have a large enough corpus ourselves (the text of congressional bills) that would encode relations that are particular to our specified task, we are in the unique position of gaining a considerable amount from training our own embeddings instead of relying on existing embeddings.

Mnih and Kavukcuoglu (2013) determine a scalable approach to learning word embeddings by training using noise-contrastive estimation (NCE), which reduces the estimation of an unknown density to a problem of probabilistic binary classification: a logistic regression classifier is used to discriminate between samples from the data distribution and samples drawn from a noise distribution. This technique reduces computation because it allows fitting a model that is not explicitly normalized – the normalization constant is the denominator in equation (2), making the training time independent of the vocabulary size (W).

Using the software *word2vec* provided by Google, word embeddings were learned from the

Figure 3: T-SNE graph (without axes) showing Word Embeddings Learned on 110th Congressional Bills Corpus reduced to two dimensions. Distance between words corresponds to proximity found in the corpus.

congressional bills corpus using the Skip-gram model described above. The model was implemented in *Python* using the newly developed *TensorFlow* module, which has been designed specifically around easy development of deep learning systems (Abadi et al. 2016). *TensorFlow* refers to a deep-learning architecture created by Google and used to design, build, and train deep learning models. At its simplest, *TensorFlow* can be used do to numerical computations with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors (in our case, it is the learned features of the words from the word embeddings), that are communicated between these edges. A tensor is the mathematical representation of a physical entity that may be characterized by magnitude and multiple directions; this unique vector representation of data is the building block off of which the deep learning modules in *TensorFlow* are used.

Additionally, in *TensorFlow*, a t-distributed stochastic neighbor embedding (t-sne) graph is available to visualize the reduced vector space of the learned word embeddings (Abadi et al. 2016). Figure 3 shows this t-sne plot from which a subset of the words may be seen projected onto a two-dimensional space. It is revealing that some words of interest appear reasonably "close" together in the center-right to bottom-right sectors of the word cluster: "Education," "Treasury," "Defense," "Energy," "Veterans" – all of which are cabinet level departments. Additionally, in the bottom-left sector of the cluster, words such as "regulations," "program," "projects," and "activities" appear near each other, all of which are words that may be associated with agencies performing some task that has been delegated to them. These observations offer evidence that learning word embeddings should improve underlying classification accuracy.

After acquiring vector representations of the words comprising the legislative corpus, the use of a convolutional neural network for text classification becomes relatively straightforward, following a simplified approach to that in Kim (2014). Figure 4 introduces the simplest

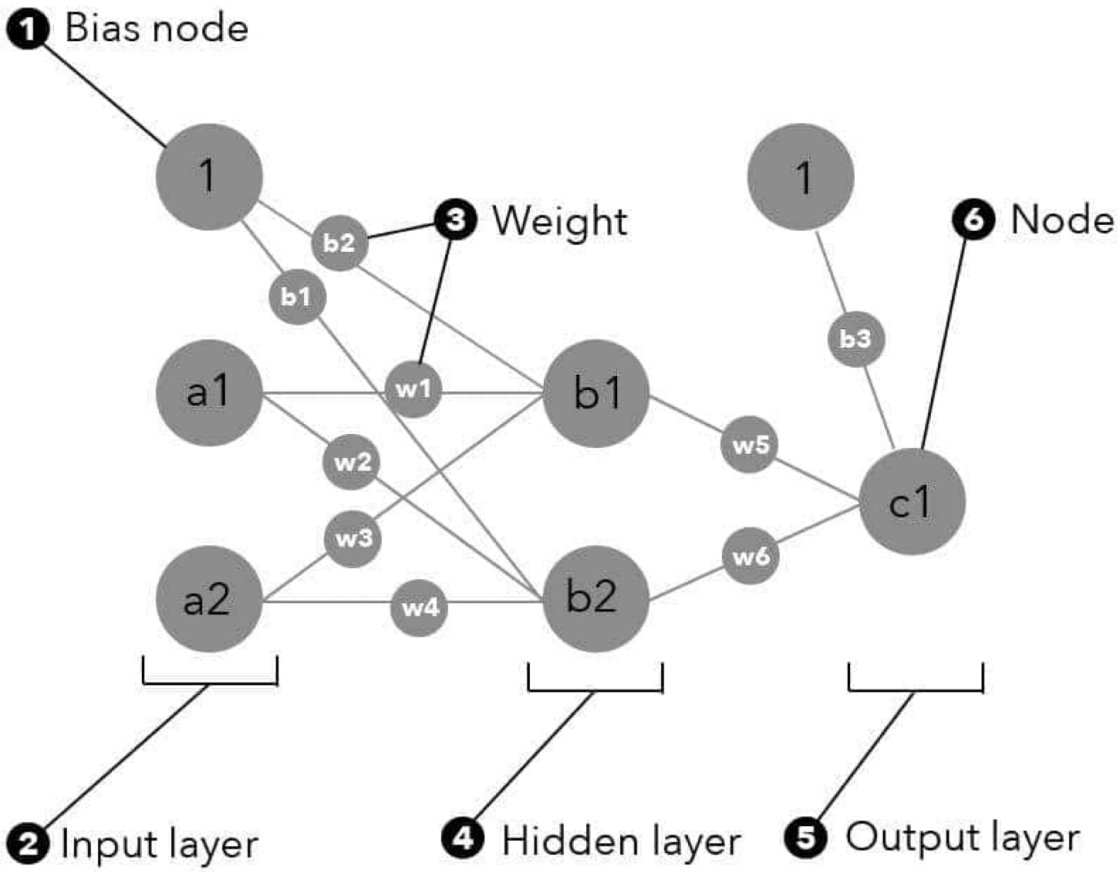


Figure 4: Illustration of Simple Neural Network from Taylor and Koning (2017)

form of a neural network. Essentially, neural networks have three basic parts (defined as "nodes) an input layer, a hidden layer, and an output layer. Inputs are simply what raw data goes into the network to be analyzed, the hidden layers are the tangle of nodes, weights, feature spaces, and regions, that reduce the underlying data to comprehensible units, and the output layer is the reading through of the data in the hidden layer into the classification task that the model is built for.

Suppose that the i-th word in a bill section is represented by a k-dimensional vector $\mathbf{v}_i \in \mathbb{R}^k$. Each bill section is padded to be the same length n, and a single section is then represented as the concatenation (stacking) of the word vectors that comprise it:

$$\mathbf{v}_{1:n} = \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \cdots \oplus \mathbf{v}_n \tag{3}$$

A convolution operation occurs over a window of h words, and involves the application of a filter $\mathbf{w} \in \mathbb{R}^{hk}$ to the window of text, the addition of a bias term $b \in \mathbb{R}$, and a non-linear transform f. For each filter, convolve over a window of size h:

$$c_i = f(\mathbf{w} \cdot \mathbf{v}_{i:i+h-1} + b) \tag{4}$$

The $\cdot$ operation in equation (4) is the dot-product – the sum over element-wise multiplications. For our purposes, the non-linear activation function chosen is the *rectified linear unit* (reLU), which is defined as: $f(x) = \max(0, x)$. We obtain from equation (4) a scalar feature c_i, and by convolving over each possible window of words in the bill section, we obtain a feature map. Each bill section is represented as the concatenation of vector representations of its comprising words. We then perform max-pooling (a dimension reduction technique) over features extracted for all possible windows in bill section: $\hat{c} = \max \mathbf{c}$, where $\mathbf{c} = \left[c_1, c_2, \ldots, c_{n-h+1}\right]$ This gives us a fully connected softmax layer on top of all filters of all window lengths for classification. From this, simple logistic regression gives us the

classification estimates.

Figure 5 gives us an example of how this process works (from Zhang, Lease, and Wallace 2017). The raw text is read in at the "sequence matrix" stage, where it is decomposed in its dimensions (word embeddings, whose dimensions are reduced to d=5). With this decomposed text, the layers are called upon to find underlying associations within the feature space (the reduced space of the embeddings), the convolution step, that turns it into an arbitrarily defined number of regions and filters. The next step takes the regions and reduces them to simple feature space that summarizes the overall region. This max pooling and space reduction iteratively works through different combinations of dimension reductions until it maximizes the predictions.

As the model works from the right part of Figure 5 through the left, we are reducing and expanding the space repeatedly trying to minimize the classification error in the final step (which is where the logistic regression occurs), which reduces our task to the underlying classification scheme discussed before. The process going through the model iteratively between the hyper-parameter settings is the process of back propagation, and is the process through which the optimal values for these parameters are selected. Figure 5 goes through this process with one set of hyper-parameters chosen (3 regions of 2,3,4; 2 filters, 2 feature-space reductions, 1 step iterative max-pooling; and reduction to 6 univariate vector concatenations). The actual model iterates through a sequence of all (or most depending on what specific model you're running) and chooses the hyper-parameter values that minimize classification errors. Figure 6 illustrates how back and forward propagation work within the model: showing the iterative process through which the model evaluates its performance and selects the values for the hyper-parameters that best capture the outcome of interest.

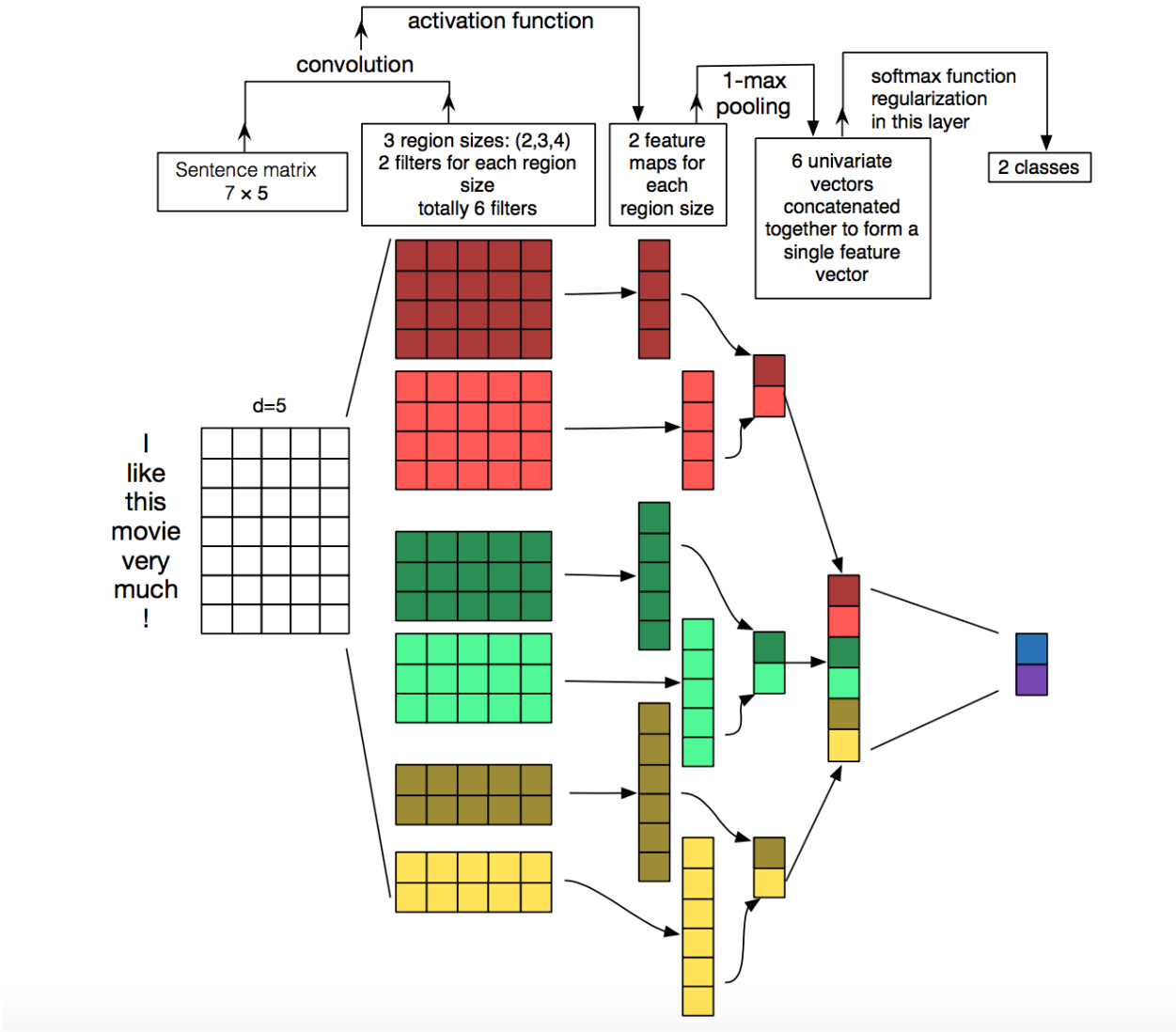


Figure 5: Illustration of Textual CNN from Zhang, Lease, and Wallace (2017)

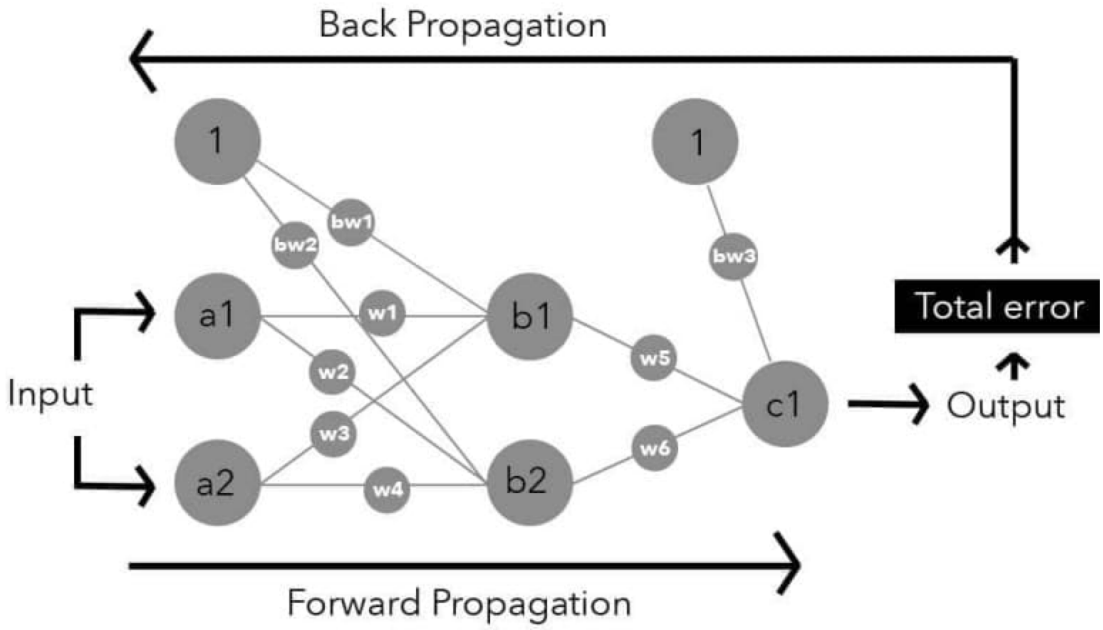


Figure 6: Illustration of Back Propagation from Taylor and Koning (2017)

Congressional Delegation to Administrative Agencies

Delegation is necessary for the operation of the United States government. Political scientists generally view congressional delegation as a trade-off between efficiency and accountability (Niskanen 1971; McCubbins and Schwartz 1984; Kiewiet and McCubbins 1991 among many others). Hypothetically, congressional delegation can have enormous productivity gains for both individual members of Congress and administrative agencies.However, in the course of designing delegation – the principle, Congress, faces numerous internal coordination problems. Further, Congress must account for how delegation can create opportunities for Agencies to act against its interests the problem of agency loss. For these reasons—the institutional, partisan, and policy-making nature—congressional delegation been a focal point for the study of political institutions.

Often, researchers have formalized the intuition of delegation through versions of the "ally principle" (Epstein and O'Halloran 1999; Huber and Shipan 2002; Bendor and Meirowitz 2004; Franchino 2007; Gailmard and Patty 2012; Farhang and Yaver 2016), which argues that when the executives interests are aligned with those of the legislature, legislators are more willing to pass legislation that delegates significant authority to agencies. By contrast, when legislative and executive policy interests diverge, legislators tend to favor institutional structures which provide greater oversight opportunities.

Beyond institutional factors, some have posited that the design of agency authority is affected by characteristics of the issues and policy areas addressed (e.g., McCubbins 1985; McCubbins and Page 1986; Epstein and O'Halloran 1999; Bendor and Meirowitz 2004). For example, many expect lawmakers to design complex decision-making infrastructures when addressing issues that are salient to them or for their constituents. While the role of issue and policy areas have been discussed extensively in the theoretical literature on the formal allocation of authority to agencies – these concepts are largely ignored in empirically-oriented

scholarship. This limitation results from a simple measurement problem, even for motivated and well-resourced academics: reading and interpreting legal texts is labor-intensive. Thus, most empirical work on the allocation of authority has been restricted to single policy areas or to small sets of significant" legislation.

Given the generalizability limitations of previous empirical studies on delegation, our work created a generalized dataset measuring delegation to administrative agencies from Congress should fulfill an essential role in the literature, and provide fertile ground for the testing of new and old theories.

Beyond the viability of this as an avenue of research, the remaining questions are, then why would we expect this to work best within an active learning framework? As discussed earlier, active learning is most often utilized in machine learning settings where getting additional training labels is costly, the task is both computationally straightforward yet might be conceptually complex—active learners appear to work best in cases of binary classification, for example–and setting up a researcher-to-classification pipeline is simple. In this case, we can see precisely why hand coding delegation in bills would fulfill these criteria. First of all, it is a task that requires both training and familiarity with how Congress writes bills: the use of statutory language is always done deliberately and with the mind that, the courts, the Executive Branch, and future Congress will have to interpret specifically what was written. Furthermore, it is in Congress's best interest to standardize the writing of bills such that there is little flexibility in precisely what is being proposed or not. So how Congress interacts with administrative agencies through bills is already mostly predetermined.

Secondly, this task, although relying on the uniformity of statutory language, still requires a fair amount of careful reading, and is not something that can be done too quickly without sacrificing accuracy. This, combined with the simple classification scheme where Congress either delegates authority to an agency or not, gives us a situation where any given classified section is not difficult to come by, but it is labor intensive to scale up the coding. Thus,

the need for active learning: if we are to get reasonable accuracy in this classification task without hand-coding too many sections, targeting which sections are the most profitable is an efficient way to use scarce resources.

We believe that this exact issue is not unique to coding agency delegation in Congress, but would apply to a vast number of classification tasks measuring latent concepts. This approach, we argue, reduces the need for additional computational bottlenecks for such tasks and could make classifying abstract features in large text documents much more tractable.

Data

We utilize data for all versions of all bills (both successful and unsuccessful) for the 110th and 111th Congresses. We break each version of each bill into titles and analyze them at the bill title level. We do this for three reasons: first, because each title in each bill will deal with a particular agency or activity, but would contain an entire delegatory phrase, keeping the task more straightforward. Second, any given bill could delegate authority to multiple agencies in multiple titles, so to avoid missing any additional delegations we wanted to reduce it to units that are about a single delegation. The final reason we chose bill titles as opposed to sentences, is that sentences quickly expanded our sample size (which was already very large), and would slow down computations dramatically, without any obvious mechanism for improving the classifications.

While it might seem advantageous to go even more fine-grained than the bill section, there are real limitations drawn from how bills are written. For example, the most common unit of analysis in natural language processing tasks is the sentence. Construction of any given sentence within a bill is, however, mostly contingent on the remaining information presented at the title level. Furthermore, in titles where authority is delegated to an agency, interspersed with regulations and other tasks given to an agency, sentences end up being less

precise than looking at the whole section, which provides much-needed context. Studying bills at the title level will ultimately let us make better inferences about the agencies or programs wo which Congress has delegated. To accomplish this, we hand-coded around 2700 bill titles (selected randomly by bill) out of 1.5 million, with around 33% of sections containing a delegation of authority.

Delegation Coding

Essential to our project is a consistent definition of delegation to administrative agencies. An act of delegation is a mandate or permission for a federal agency or program (including the President) to exercise public authority in some way (see McCubbins, Noll, and Weingast 1987; Kiewiet and McCubbins 1991; Epstein and O'Halloran 1994; Huber and Shipan 2002; and Gailmard and Patty 2012 for a discussion of this point). For our task we stated that allocating money for federal agencies to spend, instructing agencies to promulgate rules, granting agencies the ability to exempt themselves from preexisting rules, requiring agencies to compile reports or commission pilot studies, and charging agencies with the enforcement of specific policies are all examples of acts of delegation.

For the hand-coding, we gave straightforward instructions as to how we identify delegation. First, is Congress acting upon an administrative agency? This will include all references to "The Secretary of...", "The Administrator", "The Commission[er]", "Head of the ... agency", "The Administration of ...", "Office of ... ," "Attorney (or Surgeon) General", and "Corporation," among others. We operated with a list of the over 1000 administrative agencies and worked to match each instance of delegation to one of those agencies. Most often, if Congress is referring to a governmental entity (except for organizations already within Congress, which they make obvious), it is an administrative agency. We make an exception for delegating to the courts or states and local governments because those are separate, and therefore the rules governing them are different.

Second, what is the title asking the agency to do? In general, Congress delegates authority by asking an agency to perform a specific a task, collect information, write new regulations, hire people, write a report to Congress on their activities, delegate authority to sub-agencies or outside of government, and make or distribute an award, among many other things. These are the big picture tasks as broadly defined. A bill is not delegating authority if only appropriates money, if they are referencing actions already taken, or if Congress is writing new rules or regulations strictly. Keeping these actions separate allows us to track statutorily derived authority for the agencies, not merely what funds they have been allotted.

Here are some examples of bill titles that were selected by the active learner as uncertain (and how we ended up coding it). Titles the model selected as ambiguous (logit close to 0.5)

- `Section 2402. energy conservation projects . using amounts appropriated pursuant to the authorization of appropriations in section 2403 a 6, the secretary of defense may carry out energy conservation projects under chapter 173 of title 10, united states code, in the amount of 800000` (does delegate authority to an administrative agency)

- `Section. 2. reemployment of foreign service annuitants...b the authority of the secretary to waive the application of subsections a through d for an annuitant pursuant to subparagraph c i of paragraph 1 shall terminate on September 30, 2008. c the authority of the secretary to waive the application of subsections a through d for an annuitant pursuant to subparagraph c ii of paragraph 1 shall terminate on September 30 , 2009` (does NOT delegate authority to an administrative agency)

In the above examples, it is clear both why the algorithm would have selected them as ambiguous classifications and why human readers would classify them correctly. Take the top section, dealing with "Energy Conservation Projects." A quick read of the section

makes it clear that Congress is delegating authority to the Defense Department (through the Secretary of Defense) to spend $800,000 on energy conservation projects. The classifier may have been tripped up by the added verbiage of the task (may carry out) and the addition of US code language in between. It is also possible that this was the first time the classifier had seen the secretary of defense as an actor. In the second section, where Congress is setting up "foreign service annuitants", it is clear that the agency is not being given an extra task or authority, but only describes the process through which applications must be processed (and when they terminate). This section is an example where the language (context-free) would indicate the possibility of delegating authority, but additional context (plus a close reading) makes it clear that this is not occurring. These are only two examples pulled from a random run of the active learning module, set up to illustrate the nature of the classification task.

Model Performance

For this project, we have two propositions that make predictions about how our models perform. The first prediction is that the convolutional neural network model will outperform traditional word-document-matrix based models of classification, regardless of specifications. Secondly, we predict that the active learning versions of the classifiers will outperform the same models using traditional random sampling.

To demonstrate the first proposition, we used a text classification model that is widely regarded as one of the more accurate machine learning approaches: the support vector machine (SVM) with tf-idf weights, where $tf\text{-}idf_{t,d} = tf_{t,d} \times \log \frac{N}{df_t}$. We used the standard set of preprocessing techniques for classification tasks: removed punctuation, removed numbers, removed symbols, reduced to lower case, removed stop words, stemmed, and tokenized to unigrams.[6]

[6]We tried to make as few assumptions as possible when doing this, keeping in mind the concerns raised by Denny and Spirling (2018).

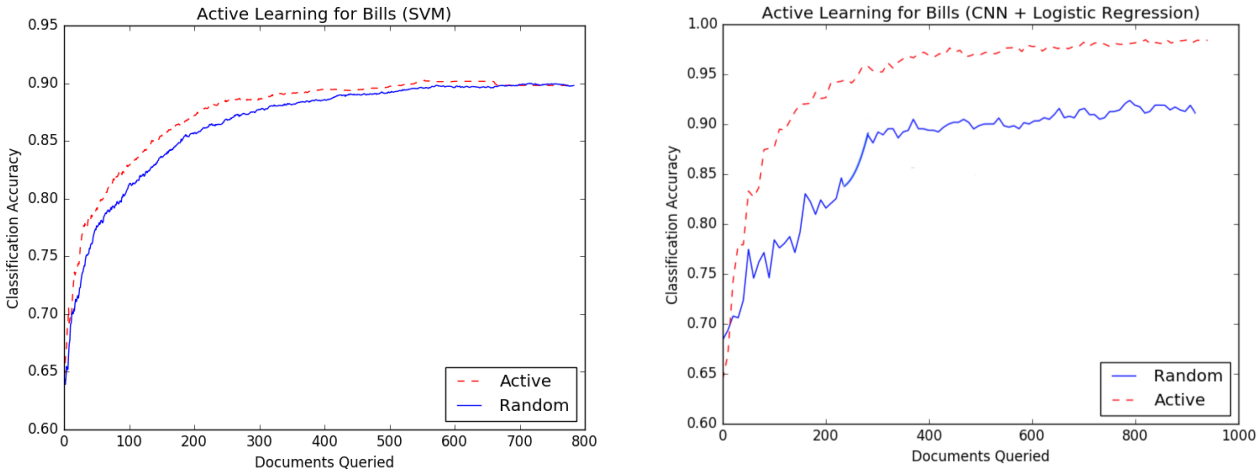


Figure 7: Active learning sampling versus random sampling for SVM and CNN models. Note that the active learning models generally outperforms the random sampled models and that the CNN generally outperforms the SVM.

The SVM takes our vectorized documents, given the tf-idf weights, and performs a series of high dimensional geometric reduction techniques to best separate our data. We may consider the tf-idf weighted vectors as points in a high-dimensional space, with dimensionality determined by the number of unique tokens in the entire corpus. Additionally, each of these points is assigned a label, which is what the SVM is trying to classify. Within this high-dimensional space, we may consider a hyperplane between the points of the two classes that serves as a decision boundary: all points on one side of the hyperplane receive one class designation, while all points on the other side receive the other class designation. The distance between this hyperplane and the closest points of both classes (known as the *support vectors*) is defined to be the margin. For an SVM, the decision boundary hyperplane is chosen to be the one for which the margin is maximized (Tong and Koller 2001; Gimmer and Stewart 2013).

To test our second proposition, that the active learner outperforms the randomized learner, we utilized a very standard active learning test. Given our hand-coded sample

of 2700 documents, we break our data into a training set and the test set. To test the validity of the active learning model, we start with a smaller set of documents and train the model on that set, before evaluating it on the remaining documents. A standard active learning test is to start with a minimal number of documents in both models—for example, 100—and run it, knowing the classification accuracy will start out poorly. We then have the active learner identify in the test set which ten documents were closest to the classification threshold of 0.5. We then rerun the model with the extra ten documents added in. For the non-active learner, we add in ten randomly selected documents. We set the upper threshold of held out documents to be 1700, so our classification set would always have 1000 in it. We did this because 1000 hand coded documents is a reasonable threshold for a standard researcher to obtain, and it means that some data will *always* be kept out, which prevents the samples from the two different classifiers from becoming the same.

Figure 7 shows the results of this for both the SVM on the left and the CNN on the right. We can see that in both cases, the active learner outperforms the random sampler. For the SVM, the active learner outperforms the random sampler for most of the sample, up until we get close to 800 documents. For the CNN, the active leaner always outperforms the random sampler, and the margin holds steady from about 400 documents on. Figure 7 also shows that the CNN generally outperforms the SVM, although the two random samplers perform mostly the same. This result suggests that our approach of using a CNN within an active learning framework is the best way to perform these classifications, although the traditional approaches would work moderately well too.

Classification and Delegation Results

Now that we have delegation estimates for each bill section for each version of each bill from the 110th and 111th Congress, the remainder of the paper will be exploring 3 factors

of these delegation estimates. First, how closely do our classification of delegatory sections map onto existing measures of discretion (in particular, we will be validating the much used verbosity measure made popular by Huber and Shipan (2002)). Second, what Congressional features predict total delegatory load for each bill; we will compare this to some results from Epstein and O'Halloran, but focus mostly on validating our measure against things that should clearly be true. And third, what agencies are getting the bulk of the delegations in Congress by party. We will also be looking at comparing the effects of divided government in the 110th Congress vs. unified during the 111th. The point of this section is at once to show that our measure of delegation is consistent with the most of the theories from the literature, but also to show that some of the most widely used measures miss out on some of the points a section-by-section delegation coded dataset provides. We are also the first example of examining how the legislative process changes delegatory language, and the first to test delegation models on bills throughout the process *including those that do not pass.*

What does Delegation Look Like?

The first thing we wish to do here is determine similar our direct measure of delegation is to the most widely used proxy: bill length, which was first introduced in Huber and Shipan (2002). The logic of the set up is rather straightforward; Huber and Shipan (2002) claim that the longer a bill is, the more likely it is to be requiring specific returns from agencies, and this generally entails more fine grained oversight and control. Shorter bills, ones that discuss delegation and discretion more broadly, are less likely to have the same requirements and are instead going to allow the agencies more free reign. It has been treated by the literature (see Clinton et al 2012 as an example) that longer bills = more delegation and more power granted to the agency. This is a straightforward test for us. We have the codings in two different forms. First, we have the bill sections labeled section by section. If their model is true, we would expect the length of a given section to predict delegation. Second, we have

the larger point about overall discretion. For that we turn to aggregated models (at the bill level) and a general measure of discretion that is widely used, namely the delegation ratio (Epstein and O'Halloran 1999; Anastapoulous and Bertelli 2020; etc.). The delegation ratio is defined as the total number of sections delegating authority divided by the total number of sections; essentially, how much of the bill is focused on delegating authority. The higher the number, the more total delegation proportionate to the bill is occurring. This penalizes omnibus legislating and other forms of massive legislation that tries to do many things at once, since pure aggregate counts of delegation would overweight these terms.

Table 1: Comparing Total Words to Delegation Measures for 110th Congress

	Delegation		**Delegation ratio**	
	Logit	**Logit w/ Mixed-Effects**	**Beta Mixed-Effects**	
	Model 1	**Model 2**	**Model 3**	**Model 4**
Words/1000	0.829^{***}	0.942^{***}	-0.002^{***}	-0.001^{***}
	(0.008)	(0.006)	(0.0003)	(0.0003)
Number of Referrals				-0.057^{***}
				(0.011)
Sponsor Chair of Committee				-0.332^{***}
				(0.032)
Sponsor Chair of Subcommittee				-0.236^{***}
				(0.030)
Sponsor Republican				0.052^{**}
				(0.023)
Constant	-1.087^{***}	-1.584^{***}	0.002	0.132^{***}
	(0.007)	(0.009)	(0.010)	(0.021)
N	139714	139714	8847	8847
R-squared	0.124		0.006	0.032
Log Likelihood	−82265.290	−77319.430	754.258	876.266

$^{***}p < .01$; $^{**}p < .05$; $^{*}p < .1$

First, we see that the number of words relates directly to likelihood given bill section delegates authority. Total discretion in a bill—as measured by the delegation ratio (Epstein

and O'Halloran amongst others)—is not associated with total number of words. In reality, relationship between length of bills is only weakly associated with how much agency discretion there is and that indirect measures of this may have been confounding. Key people who use this measure: Huber and Shipan (2002); Clinton et al. (2012); McGrath (2013); among dozens more.

Table 2: Confusion Matrix for Logistic Regression on Words to Delegation by Section

Label from words	Does Not Delegate	Delegates
Does Not Delegate	84194	36312
Delegates	4892	14316

We see this more clearly in Table 2. This is the confusion matrix for Model 1 from Table 1, comparing predicted labels from the model to the actual delegation sections. Overall predictive accuracy is okay, around 71.2%. What we see is that relying only on word count, the model is able to do a pretty decent job of predicted non-delegatory sections: of the sections we designate as non-delegatory, the model gets 84,194/89,084, around 94.5% correct. For sections that delegate authority, however, the model performs a lot worse: it gets 14,316/50,628 correct, only 28.3% accurate. So, in terms of predictive modeling, knowing the total number of words does a good job of eliminating the non-delegatory sections, but is worse than random for longer bill sections. This is consistent with total word length for a section being a poor proxy for overall delegation and deference to an agency. Similarly, if we use a median words or mean words selection criteria (no model, all sections above the mean/median are coded as delegatory), we find that predictive accuracy stays about the same, at 72.1% and 69.4% accuracy respectively. This suggests that any paper that uses number of words, in some form or another, as a stand in for delegation or discretion is, at best only getting a weak signal: the performance of the logistic regression suggests even more strongly that these designations are almost certainly missing entire types of delegatory action, in most likely systematic ways. It is, as we see, a deeply flawed proxy.

What Bills Delegate?

Now that we've seen what delegation and discretion do not look like, we should next venture to see what they do in fact look like. For the remaining analysis performed in this section, we aggregate up the delegating activities at the bill version level and include data for both the 110th and 111th Congress.[7] So, in our data, we will have multiple versions of each bill as it progresses through the legislative process, but only gather delegation stats for each bill. We count up the sections that delegate authority as our main variable of interest here.

To make the analysis analogous to the delegation ratio, but not as constrained by forcing the results to be between 0 and 1, we include total number of bill sections as a predictor in each model. Because only around 60% of bills delegate authority, we ran these models as zero-inflated negative binomials (ZINB), though nothing major changes if we only ran a negative binomial model or OLS.

We included as predictors several variables from the literature that should predict the size and scope of the delegation portions of the bills. First, we the number of committees a bill is referred to. Next we include information about the bill sponsor; whether or not they are a referred to committee chair or subcommittee chair, and their party. We then include variables about bill progression; a dummy for whether or not the bill is reported out of committee, and another if the bill passes the chamber. In a lot of ways, these models are mostly exploratory and confirmatory; since we believe this to be a good model of bill delegation, we would expect each element that makes a bill more encompassing (multiple committee referrals, having the chairs champion the bill) would correspond with additional delegation. This should confirm with the political/institutional accounts of delegation first analyzed empirically in Epstein

[7]We didn't include the 111th in the first set of analysis in this section so we could validate our measure against length directly, factoring both hand-coded sections and machine coded sections. Since all of the 111th Congress is machine coded we know that it would a) entirely replicate the 110th results (which it does and can be seen in the appendix), b) not allow for their to be differences between hand and machine coding, although we found no significance difference between the two for the 110th. We believed this created more problems than it fixed to include it, so we simply moved the 111th Congress analysis to the appendix.

and O'Halloran (1999), but we will also be addressing partisan issues as well.

Table 3: Zero-Inflated Negative Binomial Model of Number of Delegating Sections

	Delegating Sections		
	Model 1	**Model 2**	**Model 3**
Number of Delegating Sections: Negative Binomial			
Number of Referrals	0.130^{***}	0.125^{***}	0.138^{***}
	(0.012)	(0.012)	(0.012)
Number of Bill Sections	0.031^{***}	0.030^{***}	0.030^{***}
	(0.001)	(0.001)	(0.001)
Sponsor Chair of Committee		0.243^{***}	0.169^{***}
		(0.033)	(0.036)
Sponsor Chair of Subcommittee		0.175^{***}	0.095^{***}
		(0.033)	(0.034)
Sponsor Republican			-0.148^{***}
			(0.027)
Report out of Committee			0.303^{***}
			(0.030)
Pass Chamber			-0.260^{***}
			(0.032)
Delegation: Logit			
Number of Referrals	0.078	0.063	-0.532^{***}
	(0.071)	(0.072)	(0.114)
Number of Bill Sections	-1.488^{***}	-1.471^{***}	-1.480^{***}
	(0.066)	(0.065)	(0.068)
Sponsor Chair of Committee		0.384^{**}	0.427^{**}
		(0.176)	(0.183)
Sponsor Chair of Subcommittee		-0.002	0.069
		(0.149)	(0.157)
Sponsor Republican			0.175^{**}
			(0.089)
Report out of Committee			-0.083
			(0.123)
Pass Chamber			1.083^{***}
			(0.135)
N	15192	15192	15192
Log Likelihood	-24896.710	-24858.580	-24726.710

$^{***}p < .01$; $^{**}p < .05$; $^{*}p < .1$

What Agencies Get Delegated To?

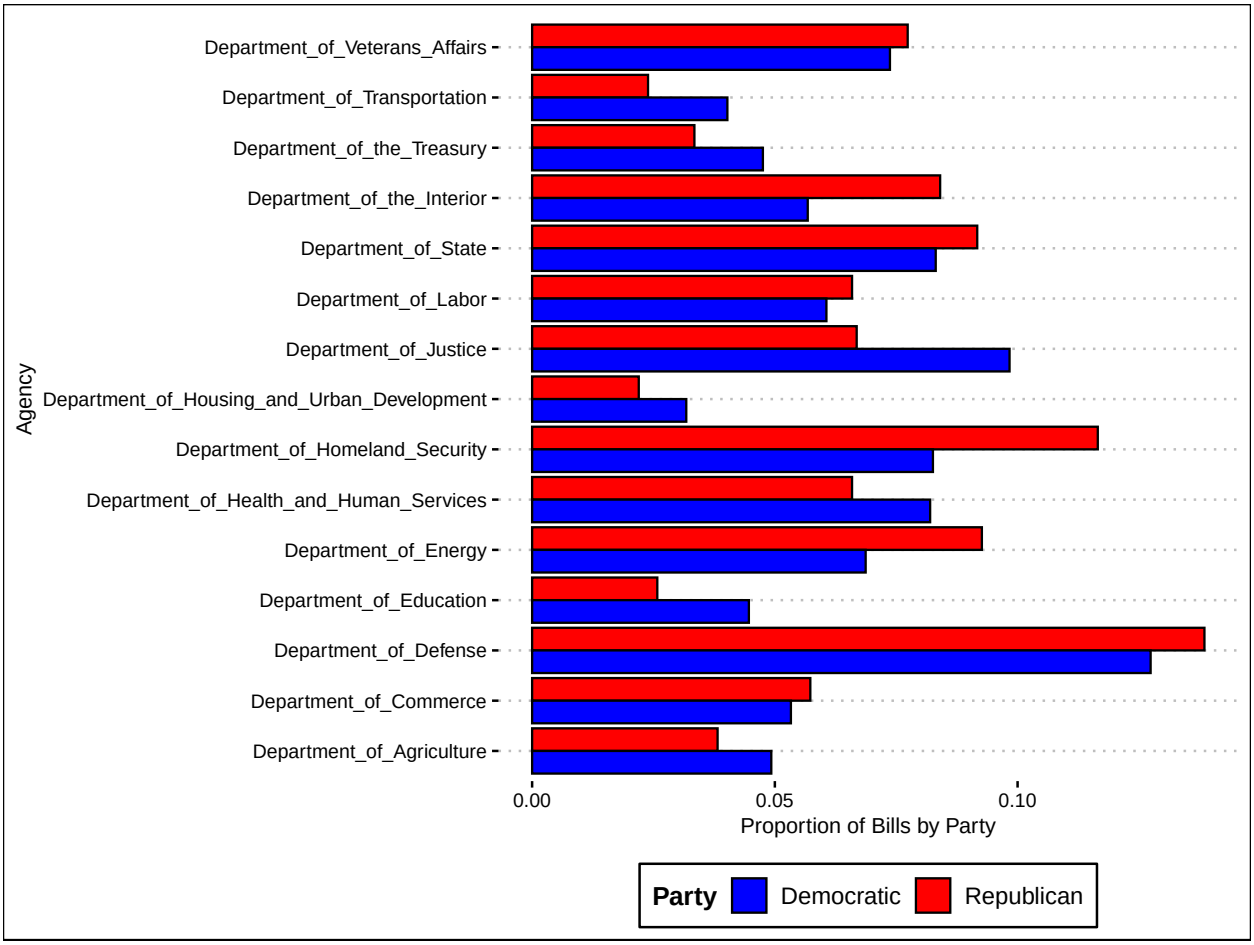


Figure 8: Proportion of Bills by Party Delegating to Cabinet Level Agencies

Clear ideological picture emerges, but not as stark as agency ideology literature suggests. See Lowande (2018) for a plausible alternative account that shows that changes in agency ideology matter less to Congress in terms of it's impact on Congressional oversight than do interpersonal connects. See also Johnson and Chen (2014) data for comparable mapping.

Table 4: Negative Binomial Model of Number of Agencies in Delegating Bills.

	Number of Agencies		
	Model 1	**Model 2**	**Model 3**
Number of Referrals	0.016	0.023**	0.033***
	(0.010)	(0.010)	(0.011)
log(Number of Bill Sections)	0.725***	0.700***	0.698***
	(0.010)	(0.010)	(0.010)
Sponsor Chair of Committee		0.071**	0.036
		(0.033)	(0.036)
Sponsor Chair of Subcommittee		0.272***	0.230***
		(0.032)	(0.033)
Sponsor: Republican			−0.116***
			(0.030)
Report out of Committee?			0.127***
			(0.031)
Pass Chamber?			−0.125***
			(0.033)
Constant	−1.052***	−1.071***	−1.048***
	(0.026)	(0.026)	(0.029)
N	8847	8847	8847
Log Likelihood	−13760.730	−13723.260	−13706.220
θ	2.179*** (0.086)	2.265*** (0.092)	2.281*** (0.093)
AIC	27527.460	27456.530	27428.430

***p < .01; **p < .05; *p < .1

Do Parties Delegate Differently Under Unified or Divided Government?

Discussion

In this paper, we have demonstrated how an active learning convolutional neural network for classifying text can be used aid in the study of a notoriously tricky problem in political science: assessing agency delegation in Congress. We hope that the methods we have proposed are clear and usable for other applications and that gains in classification accuracy–all the while reducing the needs for extra documents to be hand labeled–fills a niche in the discipline in helping researchers tackle challenging problems in a more cost-efficient manner.

This paper is the first significant step in this project, establishing exactly how we go about doing the classifications that will enable us to do much more work in the figure. We are continuing to improve the classifiers and use the better working model on the remainder of our data, including looking at some ensemble performance metrics to see if improvements can be made on top of what has already been done. Doing these classifications on the whole set is both time-consuming and arduous, so we strive to make sure our model is performing at its best before we start using our newly learned labels in an empirical setting. We are also in the process of building an entity extractor with our names of agencies set aside to match each section that delegates authority to a specific agency.

Ultimately, we will then have every bill section classified by whether or not it delegates authority to an administrative agency, which agencies are having their authority augmented, and other relevant information learned from the bill as a whole. With that, we can learn a lot about which members write bills that delegate authority, to what agencies, and how much money they appropriate to agencies to go along with the increased powers. This information intends to be part of a more massive project on how we can learn about how Congress uses statutory language to enact its agenda and how the modern legislative process does, and does not, provide oversight and guidance to the implementation of policies. These classifications will hopefully help provide the means through which we can test our theories

of delegation and Congressional oversight on a larger scale basis and provide the nexus for increased research into the implications of statutory language.

References

Abadi, Martin, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. "Tensorflow: A system for large-scale machine learning." In *OSDI*, vol. 16, pp. 265-283. 2016.

Balla, Steven J. "Administrative procedures and political control of the bureaucracy." American Political Science Review 92, no. 3 (1998): 663-673.

Bawn, Kathleen. "Political control versus expertise: Congressional choices about administrative procedures." *American Political Science Review* 89, no. 1 (1995): 62-73.

Bendor, Jonathan, and Adam Meirowitz. "Spatial models of delegation." *American Political Science Review* 98, no. 2 (2004): 293-310.

Clinton, Joshua D., Anthony Bertelli, Christian R. Grose, David E. Lewis, and David C. Nixon. "Separated powers in the United States: The ideology of agencies, presidents, and congress." *American Journal of Political Science* 56, no. 2 (2012): 341-354.

Collingwood, Loren, and John Wilkerson. "Tradeoffs in accuracy and efficiency in supervised learning methods." *Journal of Information Technology & Politics* 9, no. 3 (2012): 298-318.

Denny, Matthew J., and Arthur Spirling. "Text preprocessing for unsupervised learning: why it matters, when it misleads, and what to do about it." *Political Analysis* 26, no. 2 (2018): 168-189.

Drutman, Lee, and Daniel J. Hopkins. "The Inside View: Using the Enron Email Archive to Understand Corporate Political Attention." *Legislative Studies Quarterly* 38, no. 1 (2013): 5-30.

Epstein, David, and Sharyn O'Halloran. "Administrative procedures, information, and agency discretion." *American Journal of Political Science* (1994): 697-722.

Epstein, David, and Sharyn O'Halloran. "A theory of strategic oversight: Congress, lob-

byists, and the bureaucracy." *Journal of Law, Economics, & Organization* (1995): 227-255.

Epstein, David, and Sharyn O'Halloran. *Delegating powers: A transaction cost politics approach to policy making under separate powers.* Cambridge University Press, 1999.

Farhang, Sean, and Miranda Yaver. "Divided government and the fragmentation of American law." American Journal of Political Science 60, no. 2 (2016): 401-417.

Firth, John R. . "A synopsis of linguistic theory 1930-1955". *Studies in Linguistic Analysis.* Oxford: Philological Society: 1-32. 1968

Franchino, Fabio. 2007. *The Powers of the Union: Delegation in the EU.* Cambridge University Press.

Gailmard, Sean, and John W. Patty. "Formal models of bureaucracy." Annual Review of Political Science 15 (2012): 353-377.

Grimmer, Justin. *Representational style in Congress: What legislators say and why it matters.* Cambridge University Press, 2013.

Grimmer, Justin, and Gary King. "General purpose computer-assisted clustering and conceptualization." *Proceedings of the National Academy of Sciences* 108, no. 7 (2011): 2643-2650.

Grimmer, Justin, and Brandon M. Stewart. "Text as data: The promise and pitfalls of automatic content analysis methods for political texts." *Political Analysis* (2013)

Hopkins, Daniel J., and Gary King. "A method of automated nonparametric content analysis for social science." *American Journal of Political Science* 54, no. 1 (2010): 229-247.

Huber, John D., and Charles R. Shipan. *Deliberate discretion?: The institutional foundations of bureaucratic autonomy.* Cambridge University Press, 2002.

Kiewiet, D. Roderick and Mathew D. McCubbins. *The Logic of Delegation.* University of Chicago Press, 1991.

Kim, Yoon. "Convolutional Neural Networks for Sentence Classification." In *Proceed-*

ings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746-1751. 2014.

Laver, Michael, Kenneth Benoit, and John Garry. "Extracting policy positions from political texts using words as data." *American Political Science Review* 97, no. 02 (2003): 311-331.

Mayhew, David. 1991. *Divided We Govern: Party Control, Lawmaking, and Investigations, 1946-1990.* New Haven: Yale University Press.

McCubbins, Mathew D. 1985. "The Legislative Design of Regulatory Structure." *American Journal of Political Science* 29, 4 (1985): 721-748.

McCubbins, Mathew D., Roger G. Noll, and Barry R. Weingast. "Administrative procedures as instruments of political control." *Journal of Law, Economics, & Organization* 3, no. 2 (1987): 243-277.

McCubbins, Mathew D., Roger G. Noll, and Barry R. Weingast. "Structure and process, politics and policy: Administrative arrangements and the political control of agencies." *Virginia Law Review* (1989): 431-482.

McCubbins, Mathew D. and Rodriguez, D.B., 2011. "Statutory Meanings: Deriving Interpretive Principles from a Theory of Communication and Lawmaking" *Brooklyn Law Review*, 76(3), p.979.

McCubbins, Mathew D., and Thomas Schwartz. "Congressional oversight overlooked: Police patrols versus fire alarms." *American Journal of Political Science* (1984): 165-179.

McCubbins, Mathew D and Talbot Page. 1986. "The Congressional Foundations of Agency Performance." *Public Choice* 51, 2: 173-190.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111-3119. 2013.

Minhas, Shahryar, Jay Ulfelder, and Michael D. Ward. "Mining texts to efficiently gen-

erate global data on political regime types." *Research & Politics* 2, no. 3 (2015): 2053168015589217.

Mnih, Andriy, and Koray Kavukcuoglu. "Learning word embeddings efficiently with noise-contrastive estimation." In *Advances in neural information processing systems*, pp. 2265-2273. 2013.

Moe, Terry M. "An assessment of the positive theory of 'congressional dominance'." *Legislative Studies Quarterly* (1987): 475-520.

Niskanen, William A. *Bureaucracy and Representative Government.* Transaction Publishers, 1971.

Quinn, Kevin M., Burt L. Monroe, Michael Colaresi, Michael H. Crespin, and Dragomir R. Radev. "How to analyze political attention with minimal assumptions and costs." *American Journal of Political Science* 54, no. 1 (2010): 209-228.

Roberts, Margaret E., Brandon M. Stewart, Dustin Tingley, Christopher Lucas, Jetson LederLuis, Shana Kushner Gadarian, Bethany Albertson, and David G. Rand. "Structural topic models for openended survey responses." *American Journal of Political Science* 58, no. 4 (2014): 1064-1082.

Settles, Burr. "Active learning." *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, no. 1 (2012): 1-114.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." In *Advances in neural information processing systems*, pp. 3104-3112. 2014.

Taylor, Michael and Mark Koning. *Machine Learning with Neural Networks: An In-depth Visual Introduction with Python: Make Your Own Neural Network in Python: A Simple Guide on Machine Learning with Neural Networks.* Blue Windmill Media. 2017

Tong, Simon, and Daphne Koller. "Support vector machine active learning with applications

to text classification." *Journal of machine learning research* 2, no. Nov (2001): 45-66.

Wilkerson, John, and Andreu Casas. "Large-scale computerized text analysis in political science: Opportunities and challenges." *Annual Review of Political Science"*20 (2017): 529-544.

Zhang, Ye, Matthew Lease, and Byron C. Wallace. "Active Discriminative Text Representation Learning." In *AAAI*, pp. 3386-3392. 2017.