DEEP LEARNING & IDEOLOGICAL RHETORIC

Brice D. L. Acree

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Political Science.

Chapel Hill
2016

Approved by:

James Stimson

Thomas Carsey

Michael MacKuen

Justin Gross

Timothy Ryan

ABSTRACT

Brice D. L. Acree: Deep Learning & Ideological Rhetoric.
(Under the direction of James Stimson)

Political conflict unfolds in language. To understand the quest for, and exercise of, power, we must understand political speech. For more than a decade, political methodologists have sought to develop methods for using language to study how politicians speak to the public, to the media, and to each other. This dissertation advances that line of research. Chapters 2 and 3 address 'deep learning' methods for analyzing political texts. These computational models, built on artificial neural network architectures, seek to automatically learn complex patterns in data. Chapter 2 introduces basic deep learning models, and Chapter 3 introduces distributional word representations and convolutional neural network models. The final chapter points these methods toward a substantive problem. As a discipline, we have studied, and debated, the nature of political ideology. We have often found that, for citizens and elites alike, ideology is oriented around a single primary dimension: liberalism to conservatism. In Chapter 4, I offer a different view. Using the language from political ideologues and presidential candidates, I show that ideology can be much richer and varied than a single dimension will capture, but that political debate compresses ideological expression into a single dimension.

for Lauren

intelligent, beautiful, wonderful

ACKNOWLEDGMENTS

North Carolina—land of the pines, the Tar Heels, and the craftiest ales brewed from that pernicious and wicked weed—is the only place I could imagine embarking on this epic quest we call graduate school. I could say that I have been *fortunate* to study here, but that hardly expresses the magnitude of my appreciation. I only hope that I can convey, in measures equal to my gratitude, how thankful I am for those people who have shown me uncommon support and friendship, and without whom I could not have sustained this project.

The department of political science at the University of North Carolina has provided fertile ground for graduate work. In particular, I thank the American Politics Research Group for their feedback, which sharpened my ability to clearly build and express arguments. I am also indebted to the graduate students for whom I had the pleasure to TA. They had to hear me roar the virtues of Bayesian inference week in and week out, and somehow never charged the white board in mutiny. Their patience and feedback taught me how to explain technical material, which I hope is apparent in the work that follows.

Two groups of scholars provided the data used in this dissertation. I had the privilege to work with Justin Gross, Noah Smith, and Yanchuan Sim on building and refining the Ideological Books Corpus. The ReactLabs data was kindly shared by Amber Boydstun, who collected the data with Philip Resnik and Jeffrey Korn.

I owe my growth as a scholar to an incomparable group of advisors. Dean Lacy, for reasons surpassing my understanding, thought I might succeed in graduate school, and encouraged me to apply to doctoral programs. Philip Paolino, my first graduate advisor, sparked my interest in statistical methodology, introduced me to Bayesian inference, and most importantly, taught me that methods are a means to an end, not the end in themselves. Michael MacKuen and Timothy Ryan served on my master's and dissertation committees, and always encouraged me to look beyond the methodological minutiae and think in terms of broader, theoretical contributions.

Justin Gross bears the most responsibility for my discovery of, and love for, computational text analysis. He included me in his research during my first semester, and sent me on a path to where I stand now. Thomas Carsey, too, deserves a share of the blame. He invited me to enroll in a machine learning course he was auditing (which he promptly stopped auditing after a few weeks). Tom encourages and challenges in equal measure, and he works harder on behalf of his students than anyone I have met. Most of all, I owe a lifetime of thanks to my advisor, mentor, and harshest critic, Jim Stimson. You can paint stripes on a toad, but that does not make it a tiger. Jim does not hesitate to tell you when you've sent him a toad of a paper, no matter the prose with which you've dressed it. Jim's feedback has made me a better scholar by pushing me to lay bare my assumptions, to express my arguments clearly, and to place periods inside the quotation marks.

That understates Jim's greatest contribution to this dissertation, however. Why is it that when someone builds a wall, somebody else immediately needs to know what's on the other side? Why else? Curiosity. Discovery. Jim's work is genuinely motivated by the thrill of discovering something heretofore unknown, and sharing it with others. That much was obvious during our first meeting, and it was contagious. Jim instills in his students an appreciation for broad thinking and diverse ideas, insisting that we do not reap what we do not sow. As his student, I have learned to sow many ideas, and this dissertation sees some of those grown to fruition.

My many friends, from Carolina and locations beyond, have supported me in ways large and small. I especially owe thanks to Kelsey Shoub, my partner in methodological crime; to Eric Hansen and Andrew Tyner, whose penetrating questions early in the dissertation process made me think seriously about my role in, and contribution to, the discipline; to Aziz and Ashaya, my soundtrack when the nights were dark and full of doubts; to Chelsea Estancona, Dan Gustafson, Ryan Williams, and Rob Williams, for their support on and off the basketball court; and to Chris Blankenship, Steve Durham, Alec Macaulay, and Dan Thele, for their sustaining friendship.

Finally, I reach that highest pantheon of my gratitude: those who have loved me unconditionally throughout this process. My wife, Lauren Milam Acree, has born the weight of this dissertation as much as I. She has ceaselessly encouraged and supported my research. Lauren's kindness is woven through the document you are about to read, and without her this thesis would surely have never come together. My father Doug, my brother Addison, and my grandmother Shirley have been unfailing pillars in my life, without whom I would not have reached graduate school in the first place. And I thank my grandfather Robert, and my uncle Scott, who would have been proud of this accomplishment. I wish they could read my dissertation—or at least skim a few pages before wondering aloud 'What's this have to do with politics, anyway?'.

Theodor Geisel, better known by the *nom-de-plume* Doctor Seuss, once asked, "My goodness how the time has flewn; how did it get so late, so soon?" This dissertation represents the culmination of many years, and the contributions of many friends and colleagues. My mantra in writing this project was to keep looking forward—that if I look back, I am lost. At the end of the journey, as I write these acknowledgments, I now pause to look back. So many people have helped to get me to this place. I cannot possibly pay my debts to them, but I will strive to be worthy of the confidence they have placed in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Politics comprises, in large part, the artful (and sometimes painfully inartful) use of language. Candidates for political office barnstorm for months at a time, making their cases and leveling criticisms of opponents' actions and points of view; legislators draft and debate bills; presidents issue statements; justices hand down legal opinions; information age rabble-rousers peddle conspiracy theories to anyone with an internet connection; and journalists, columnists and bloggers dissect, reframe and disseminate it all to the American public. For political scholars, the trove of data locked away in transcripts and manuscripts is both a blessing and a curse. The answers to many fascinating questions lie within the written word; yet harnessing the data in ways both efficient and reliable poses considerable methodological challenges.

I am not the first, nor will I be the last, to address these challenges. In political science, we have sought for decades to use text as a source of data. Over the past two decades, we have built our capacity to process, quantify, and model text using computational methods. With increased computing power and advanced statistical methodology, scholars have developed new, powerful, and efficient tools to extract patterns from, and test substantive theories using, text as data. My dissertation seeks to advance this rich history of methodological and substantive work. It does not represent a paradigmatic shift in political methodology. It represents a step forward, progress in the normal scientific march toward better tools and better understanding.

This dissertation addresses three major topics. The first, which comprises Chapters 1 and 2, addresses methods for large-scale text analysis. Specifically, these chapters introduce 'deep learning' methodologies for supervised text classification to a political science audience. The second topic, addressed entirely in Chapter 2, shows how we can improve how we quantify language using distributed word representations. The third topic for this dissertation, contained in Chapter 3, points these tools toward an important substantive political science issue: ideology.

In this introduction, I will briefly describe these topics. I will also seek to make clear where this dissertation fits within the broader field of political methodology, how the dissertation moves the field forward, and why the reader should bother him- or herself with the results.

1.1 Political Text Classification

Much of political conflict unfolds in language. From speeches to media coverage, platforms to treaties, tracts to broadcasts, power and influence require communication. As Grimmer & Stewart (2013, 1) put it, "To understand what politics is about we need to know what political actors are saying and writing." This recognition is not new. For decades, political scientists have manually coded texts for various features—partisan bias, racial undertone, policy frames, conflictual or negative language, specific policy or treaty provisions, and so on. Computational methods for text analysis, which use computer programs and statistical models to automate these tasks, simply make this process faster, more efficient, and scalable to collections of texts that would be infeasible to code by hand.

In political science, we often wish to achieve one of three goals with text data: classification, clustering/scaling, and prediction. Most political scientists will find these familiar, even if they have never used text as data. These tasks are essentially the same goals we pursue in our quantitative research. The first two of these tasks fall generally into measurement techniques; that is, given texts, can we extract useful measurements therefrom? The third attempts to leverage information in texts to make forecasts about future preferences, behavior, or events.

We can also partition text analysis approaches into two broad classes: supervised and unsupervised. The former uses hand-labeled data to train a model, which can then be used to predict the labels on held-out documents. For example, a researcher could hand-code documents according to their positivity or negativity (i.e., the sentiment) of the language therein. The researcher then fits a statistical model to estimate what words or phrases predict the sentiment of the documents, and then use this information to label the sentiment of many—perhaps thousands or millions of—new documents. Unsupervised methods, by contrast, do not use hand-labeled documents. Instead, these methods attempt to uncover latent patterns in documents. Analagous to principal components or exploratory factor analysis, this class of models produces groupings or scaled representations of documents that the researcher attempts to interpret after model estimation.

Many scholars have employed supervised (e.g., Laver, Benoit & Garry (2003)) and unsupervised (e.g., Slapin & Proksch (2008), Proksch & Slapin (2008) and Spirling (2011)) models to scale documents into a continuous policy space. Other scholars have attempted to leverage text for predictive purposes. Tumasjan, Sprenger, Sandner & Welpe (2010), Bermingham & Smeaton (2011), Beauchamp (2015), and Skoric, Poor, Achananuparp, Lim & Jiang (2012) attempt to predict electoral polls and outcomes using large-scale Twitter data. Diermeier, Godbout, Yu & Kaufmann (2012) and Beauchamp (2011) use language in Senate floor speeches to predict the ideological position of Senators, according to DW-Nominate. These, and the many others I do not cite, are all important projects. But this line of research falls outside the scope of this dissertation.

Supervision My dissertation focuses on supervised models classifying political texts. In other words, I train models to classify political texts into hand-coded, discrete categories. I focus on supervised methods because I believe that supervision provides a necessary theoretical control over automated methods. We can find many unsupervised, latent representations of text data. Textual data contains myriad signals—ideological, partisan, educational, sophistication, linguistic, translational. Unsupervised methods may unveil important summaries of the documents in any collection. Unfortunately, there is no guarantee that amongst the many signals in text, any summary will produce something of substantive interest. When attempting to measure nuanced concepts, like my analysis of elite ideology in Chapter 3, supervision makes more sense, because it grounds the measurement task in substantive expertise.

Classification We deal with classification questions routinely, even if we do not think of them as such. Can we predict whether individuals are liberal or conservative? Can we explain what factors make individuals more likely to vote than others? Or make countries more likely to engage in war? How do voters choose between many political parties, or insurgent groups between potential targets? In each of these cases, we attempt to use observable information about people, countries or institutions to classify them into groups: voters and nonvoters, liberals and conservatives, Labour, Tory, or Liberal Democratic voters, and so on.

In text analysis, we face a similar challenge: how can we take the language in a document to classify the document itself? In this dissertation, I focus on classifying documents by the topics, ideology, and tone they represent. I pay particular attention to the classification of ideology, which appears in all three chapters. I build on prior work by Sim, Acree, Gross & Smith (2013) and Iyyer, Enns, Boyd-Graber & Resnik (2014), who use political language to classify documents into ideological categories, and on work by Acree, Gross, Smith, Sim & Boydstun (2014), who use ideological classification as the basis for testing a theory of post-primary moderation in presidential campaigns.

Of course, we can imagine plenty of areas where our theories require us to classify political texts. Boydstun, Gross, Resnik & Smith (2013) test a theory of policy framing by tracing the evolution of frame over time through media. Tumasjan et al. (2010) classify Twitter statements by their sentiment, and use the aggregated results to forecast German federal elections. Of particular interest to institutionalists, scholars have also used supervised methods to classify policy stances in Congressional floor speeches (Thomas, Pang & Lee 2006, Hopkins & King 2007), editorials (Hopkins & King 2007), or online debates (Somasundaran & Wiebe 2010). In essence, methods for classification are useful because they can group documents into any category of substantive interest, and provide measurements for evaluating social and political theories.

1.2 Contribution

This dissertation seeks to advance this line of political research by introducing new tools from the computer science literature. I must be clear—I did not invent deep learning methods, nor do the following chapters represent a breakthrough in the deep learning literature. Instead, my aspiration is to begin to lay the foundation with this document: to prove the usefulness of deep learning methods for political scholars, to provide practical guidance for its use, and to assess whether model evaluation benefits from domain-specific tests.

I see the contribution in three parts: (1) I introduce deep learning methods which, while used in the natural language processing and computer science literature for more than a decade, have not appeared in political science scholarship. Though not universally superior to other methods for text classification, deep learning models have the advantage of adapting to complex interactivity and nonlinearities in language, which can render them more powerful in measurement tasks. (2) I describe distributed word representations, why they are useful, and show for the first time that domain specificity can matter as much as general model performance when using distributed word vectors. (3) I revisit a longstanding conflict in political science between what we believe about ideology, and how we typically measure it. With new data and a new perspective, I make an argument that ideology can be multidimensional, even when the vast majority of our measurements to date reduce to a single dimension.

1.2.1 Methods Contribution

The bulk of this dissertation centers on deep learning methods as applied to political texts. As I have mentioned already, these methods have yet to appear in political science scholarship, despite—as I show in the first two chapters—their usefulness for text analytic tasks. I offer two reasons why. First, like many machine learning methods, artificial neural networks work well in measurement or dimensionality reduction, but are ill suited to testing substantive theories. I explore this more in **Chapter 1**, but briefly, artificial neural networks make it very difficult to assess how variables affect each other. For measurement problems, we might not care, so long as the model performs its task well. Most of our work, however, centers on testing causal theories of social behavior. Artificial neural networks—no matter how deep or complex—are ill suited to that task.

Second, deep learning methods evolved out of a different tradition than the methods most political scientists learn and use. More common econometric and statistical methods tended to progress around principled problem solving. We have Gauss-Markov assumptions but our data are correlated over time, so we derive a model to account for this autocorrelation and an estimation routine to fit the model. Artificial neural networks, by contrast, tended to develop by heuristic. These models were developed with biological inspiration, and much of the structure seen in advanced models follows this pattern. Feed-forward artificial neural networks were designed to mimic how mammal brains learn, and convolutional neural networks were developed to mimic the behavior of mammalian retinas.

As a result of its separate development, deep learning has its own vocabulary, its own traditions, and its own intuitions. This creates a high barrier to entry for scholars outside of the field. Of course, the same could be said of statistical and machine learning methods over the past several decades. Many exciting developments in our substantive understandings of politics derived from scholars adapting tools from other fields. In this dissertation, I introduce these deep learning methods, show their usefulness in several text classification tasks, and develop a domain-specific model assessment dataset for a particular type of deep learning model, known as distributed word models.

We should, of course, question why we desire new tools for computational text analysis. Are the current methods used in political scholarship not up to the tasks we demand they perform? If deep learning requires us to adapt to a different tradition of computational models, is it really worth it? My contention is yes. Current methods perform well, but suffer from some weaknesses that ultimately limit their performance. I will offer three:

1. Our methods struggle to incorporate high-order interactivity that we know exists in language. For example, we might learn much about a document by observing that two words are used together (‘climate change’) and in conjunction with others (‘so-called’) (Iyyer et al. 2014). Our existing methods struggle to adapt to this without the researcher explicitly specifying these interactions *a priori*. Deep learning methods can adapt to such

interactivity automatically.

2. Current methods tend to ignore word order, or incorporate word order by n-gramming the corpus, or treating multi-word phrases as the unit of analysis (see the final section of this introduction). Word order clearly matters for language. Ignoring it sets a natural limit on how well our models match our understandings of political text. Even n-gramming, though useful, presents serious problems, most notably that the researcher much decide how to remove unimportant multi-word phrases. N-grams are typically generated by shingling, or considering every n-length phrase that occurs in a text (Spirling 2011). This results in most n-grams as meaningless phrases. Deleting uncommon n-grams, however, can accidentially remove important, informative but uncommon phrases. Deep learning can adapt to word order without needing to n-gram the corpus or deal with its challenges.
3. We nearly universally quantify text atomically, meaning that we treat words as independent, atomic units. This ignores that words are related in natural language, which again discards useful information and limits the performance of current methods. Deep learning methods can efficiently and reliably generate continuous, lower-dimensional representations of words that can capture the semantic (definitional) and syntactic (linguistic role) meaning of words.

My first two chapters seek to address these challenges. In **Chapter 1**, I introduce deep learning artificial neural networks. Deep learning methodologies present an exciting new area for exploration in political science research. These computational models attempt to replicate, in a starkly simplified form, the information processes of the mammalian brain. Deep learning methods distribute *learning* across many computational units, much as the brain distributes learning across many neurons. Artificial neural network models, which serve as the infrastructure for deep learning, have been described in the computer science and artificial intelligence literature for decades. For much of that time, only simple models could be estimated with extant computational power. The rapid development of faster computing has enabled researchers to build larger, richer, and more complex models.

I begin with a simple example: using an artificial neural network model to perform logistic regression. With this familiar case, I introduce some of the vocabulary in the deep learning literature, and cover the basic model structure. I also introduce the many choices scholars face when specifying neural network models—such as model parameters, non-linear 'activation' functions, the dimension of the network itself—along with, when possible, practical guidance. I present a Monte Carlo simulation study, which shows how deep artificial neural networks can adapt to complex data patterns, like those found in language. Using two data sources, I go on to show that artificial neural networks not only perform well compared to common text analytic models in the political science literature, but in many cases perform better.

Building on this foundation, **Chapter 2** marks a significant step forward for the analysis of political texts. To date, political science researchers have relied on 'atomic' word representations. That is, we typically measure words as atomic units, without concern for the semantic (definitional) or syntactic (linguistic role) similarities between words. This means that we ignore a wealth of valuable information contained in language. We wish to use language because we believe it conveys meaning. Whenever possible, our methods should try to measure that meaning.

Distributed word representations are a tool to do just that. In essence, estimating distributed representations of words is simply dimensionality reduction. Instead of representing words as atomic, independent units, we try to find latent, lower-dimensional representations of words that capture patterns in word use. The result is an encoding for words that places similar words close together in a common space, and dissimilar words further away, which in turn captures what words mean and how they are used in language.

Chapter 2 also builds a new evaluation tool for these word representation. In the deep learning literature, there is an unspoken assumption that distributed word representations built upon larger and more general data should be better. I show, however, that such a strategy can actually obscure domain-specific word meanings. For instance, the word 'gridlock' in general language can imply traffic or congestion; in politics, it usually refers to partisanship and polarization. I show that validating models with general-purpose evaluation data misses this nuance. I further show that using distributed word representations trained on multipurpose text data does not always outperform representations trained on smaller, domain-specific text collections.

Finally, in **Chapter 2**, I describe a particular type of deep learning model: Convolutional Neural Networks. These models attempt to leverage spatial dependence between words in documents to, e.g., classify documents into groups. This could provide a major step forward for political research. As I discuss in the final section of this introduction, scholars face many choices when using text data. One of the most important is how to account for word order. The typical solution is to treat as the analytical unit the multi-word phrase, or *n-gram*. This suffers from many problems, however. As I point out in **Chapter 1**, *n*-gramming requires the researcher to make many data cleaning choices, such as how long the multi-word phrases should be, and how to thin out meaningless or rare phrases. These choices can be highly consequential for model performance. Convolutional neural networks, by contrast, adapt to word order naturally. They do not require the researcher to *n*-gram the documents before estimating the model. As a result, model performance can prove more stable.

I intend the contribution for the methods community, then, to be two-fold: introducing methods that hold substantial promise for extracting useful information from textual data, and moving our methods away from atomic representations of language toward representations that capture the meaning of words.

1.2.2 Substantive Contribution

In my final chapter, I address a longstanding issue in political science. We recognize as a field the importance of ideology—those core beliefs that inspire our preferences and motivate political behavior. We think of ideology as built on fundamental ideas about justice, fairness, morality, and ethics. We also describe ideological conflict largely in terms of left and right, the ongoing battle between liberalism and conservatism. This structure is born out time and again in empirical exercises, from Congress to courts to citizen policy preferences. In the vast majority of quantitative work, political researchers find a unidimensional representation of ideology as powerful as it is parsimonious (Jost 2006).

Yet *a priori*, there is no reason to assume that ideology *has* a unidimensional form. If ideology is truly about ideas, then it should extend beyond how we respond to yes/no policy questions. Is should also reflect the many distinct belief systems that inspire those preferences. I address this issue in **Chapter 3**. I present a theory of ideological censorship, which posits that the familiar unidimensional representation of ideology derives from the nature of two-party conflict. Ideology itself, I argue, may be nuanced and multidimensional for political elites with highly constrained political belief systems. Most citizens, however, show little ideological constraint beyond a general liberal or conservative orientation. Parties intent on winning power do not need to appeal to nuanced ideological beliefs. They simply need to offer a binary choice: Democrat or Republican? So long as these parties generally reflect the liberal-conservative divide, political conflict will look unidimensional.

To evaluate the merit of this theory, I use two sources of data. The first, the Ideological Books Corpus (Sim et al. 2013), comprises ideological writings from political elites, but outside of any immediate partisan or electoral competition. The second contains campaign speeches from presidential candidates from 2008-2016. Using the multidimensional and hierarchical ideological classifications provided by Sim et al. (2013), I show that many ideological classifications can be recovered using only the text that the authors employ. Further, the differences in language between ideological groups match our substantive intuition about what differentiates these groups from one another. Finally, I find evidence that a lower dimensional representation does not adequately summarize these texts. In other words, there are real and meaningful differences in the beliefs expressed by political elites that do not collapse to our traditional liberal-conservative dimension.

The same pattern does not appear in campaign speeches. To the contrary, presidential candidates do not appear to sample much of their language from particular ideological groups. Instead, candidates appear as ideological generalists, using language from many ideological groups on the left and the right. Further, the candidates' language can be easily summarized in a single dimension that maps cleanly onto a substantive understanding of left and right, with conservative Republicans on one end and liberal Democrats on the other. Even candidates in primary campaigns, where we might expect sharper ideological differentiation, appear to comply with our more traditional conceptualization of ideology, treating the main scope of conflict as one between liberalism and conservatism.

This revisits, and I hope revitalizes, an important discussion for political researchers. For a long while, our research has uncovered a single ideological dimension, and we have largely accepted this as the structure of ideological conflict in the United States. By seeking out another source of data, and by employing methods for analyzing large text collections, I show that this conclusion may be hasty. Ideology is not unidimemsional or multidimensional—it is both. The structure of beliefs, at least among elites, can be richly nuanced and multifaceted. When filtered through an partisan lens, however, ideological conflict simplifies to a battle between left and right. I hope that this theory, and the evidence I provide, advances how we conceive of, and measure, elite political ideology.

1.3 The Challenges of Text

If the tasks we perform with textual data is so similar to the rest of our quantitative enterprise, does text really differ from other sources of data? We face plenty of challenges in data analytics. Are those entailed in employing text as data really so different, so difficult, as to merit its own field of inquiry? The answer is, in the same breath, yes and no. Many methods used for text analysis are familiar, but textual data present unique problems that render it difficult to use in applied research. In this section, I discuss the challenges of dealing with textual data, as well as concepts related to how text data is processed and quantified. These concepts are essential for understanding the remainder of this thesis.

Text is challenging because it is *unstructured*. Unlike data that can be easily quantified, text is qualitative. There is no immediately apparent way to integrate text into statistical models. We face this issue plenty in other research, of course. How do we represent ideology—which we typically describe as 'liberal' or 'conservative'—quantitatively? For such problems, we make simplifying assumptions that allow us to ascribe numerical values to qualitative meaning. We assume some sort of dichotomy or scale, which we can plug into statistical models. The problem of quantification is as old as empirical political science itself.

How is text different? Text requires considerable processing to represent it quantitatively. How do we define the unit of analysis? Is it the letter, the word, the multi-word phrase, the sentence, the paragraph? Should we retain some, none, or all of the capitalization or punctuation? Should we treat documents as ordered sequences of words? As bags of word frequencies? Should we treat all words equally, or should we try to heavily weight words we find more important according to some criteria? Should we ignore very common words? The last section of this introduction addresses these questions more specifically.

Text also brings with it the legion complexities of natural language. Pause for a moment to consider how rich, nuanced, and complicated language can be. Teachers find it challenging enough to instruct students on how to properly communicate their ideas orally and in the written word. Language comprises grammar, semantic meaning, syntax, humor, negation, sarcasm… and the list continues. As humans, we have engineered the most incredible, the most ingenious, ways to communicate the complexities of emotion, desire, feeling, and belief. Set a monkey to your laptop for all eternity—I guarantee you it will not produce the works of Shakespeare. How can we possibly expect computers or statistical methods to comprehend, or leverage, the information contained in the spoken or written word?

The short answer is that we cannot, but we are getting closer. To date, the methods used in political science make simplifying assumptions that, while troublesome for a linguist, result in methods that perform fairly well. For instance, nearly all methods used in political science make the 'bag of words' assumption. This assumes that words are exchangeable, meaning that words in a document occur with some probability independent of the words that appear before or after. This clearly makes no sense. Word order matters quite a lot for humans to comprehend one another. Yet, perhaps surprisingly, ignoring word order and simply accounting for word occurrence can account for many patterns of substantive interest. Does the word 'alien' appear in a document addressing immigration? If so, we probably (though not certainly) can infer at least the general perspective of the author.

Advances in computer science include the development of deep learning methodologies. These methods have proven so successful that they power many technologies we use daily: speech recognition and talk-to-text applications on cell phones, facial recognition in social media platforms, and automated language translation, to name but a few. These tools have also proven useful in natural language processing and text analytic task. In this dissertation, I use them to estimate political tone and ideology, and to capture the semantic and syntactic meaning of words in political texts, but they have been used in more text analytic tasks than I can possibly recount here: sentiment analysis, topic modeling, authorship detection, generating new texts, and on and on.

The myriad methods—deep learning or otherwise—for text analysis makes this area both inviting and foreboding. Grimmer and Stewart (2013), in an invaluable overview published by *Political Analysis*, point out two facts about text methods. First, borrowing from Box (1979), "All quantitative models of language are wrong—but some are useful." Second, there is no single best automated text analysis method. In other words, what renders text analysis so challenging is not that the methods are impossibly complicated—most are not—but the diversity of methods available, and the many choices we must make in order to quantitatively analyze textual data.

1.4 Processing & Preparing Text for Analysis

In most applied political research to date, scholars apply *bag of words* models, which treat documents as exchangeable distributions of words or phrases. These methods underly the methods discussed in the first chapter of this thesis, and I discuss processing for this family of models first. I also discuss data processing for more advanced applications, which relax the exchangeability assumption and attempt to account for word order, which features heavily in the second chapter of this paper.

1.4.1 Bag of Words

Bag of word, or BOW, models treat documents as exchangeable distributions of words or phrases. According to this assumption, the order in which words occur does not matter. For actual language, this assumption clearly does not hold. For human comprehension, we require a logical sequence of words to express a point, and we can easily concoct sentences where word order is of paramount importance. (For example, compare the meaning of the sentences "I am not a crook." to "Am I not a crook?")

Perhaps surprisingly, though, BOW methods perform well despite this lack of verisimilitude. For many predictive tasks, knowing that a word (or phrase) occurs in a document provides sufficient evidence, even if we ignore *where* in the document that word appears. For example, if a news story contains the two-word phrase "second amendment" several times, the story is probably discussing gun rights. Where in the document this word occurs will not often matter. By assuming exchangeability of the words or word phrases, we maintain considerable predictive ability, while simplifying how we quantitatively represent documents, and how we apply models to these data.

Document-Term Representation In BOW models, we nearly always represent documents as a 'document-term' matrix (DTM). (Clearly the transpose of such a matrix would be a 'term-document' matrix, so these are equivalent.) The DTM contains a row for every document in the collection of texts, which we call the *corpus*, and contains a column for every unique word in the vocabulary. The most basic DTM contains word counts. For example, if the word "president" occurs three times in the first document, then the first row, "president" column of our DTM will contain a 3.

The DTM can also be populated with various weighted representations of the words. Frequent weights are document-proportions, which take term frequencies and divide them by the number of words in a document. This gives a sense of how prevalent a word is in a document, and helps to 'control' for document length. Another common weight is the term frequency-inverse document frequency (tf-idf) weight, which provides higher weights for terms that differentiate documents. If D is the number of documents in the corpus, $\mathbf{X}$ is the document-term matrix, and $x_{d,v}$ is the number of times that word v occurs in document d, the tf-idf weight is defined as:

$$\text{tf-idf}(d, v) = x_{d,v} \times \log \frac{D}{\sum_d \mathbf{I}(v \in d)} \tag{1.1}$$

where $\mathbf{I}(\cdot)$ is the indicator function, returning 1 if the argument is true, and zero otherwise, and $(v \in d)$ means that term v occurs in document d. In plain English, the tf-idf score gives higher weights to terms that (a) occur in many times in a document; (b) occur in only a few other documents; (c) a combination of (a) and (b). If a word occurs in every document in a corpus—for example, prepositions typically occur in nearly all natural language—the term frequency gets weighted by $\log(1) = 0$. If a word occurs in only ten percent of documents, the term frequency gets up-weighted by a factor of $\log 10 \approx 2.3$.

Unit of Analysis For many documents, the unit of analysis is straightforward. If we seek to measure the sentiment of user tweets on Twitter, we would probably treat the tweet as the unit of analysis. In such a situation, we would hand-label some subset of tweets as falling into our sentiment categories (e.g., 'positive,' 'neutral,' or 'negative'), fit a model to these data, and after tuning the model, use learned parameters to generate predictions for a large class of new data.

In many applications, however, the unit of analysis is not entirely straightforward. If we seek to classify the ideological content of political speech, should we attempt to classify the entire speech as a single document? Should we parse the speech into paragraphs, or sentences?

Maintaining larger documents can make classification more reliable. Language is noisy, and small perturbations in word use in short documents could spur large shifts in predictions. For example, if a candidate for office says "All my opponent cares about is the second amendment," a statistical model may infer that the use of 'second amendment' indicates that represents a conservative viewpoint. In a longer document, however, we would undoubtedly see far more indicators of a liberal perspective, which could reduce our mis-classification rate.

On the other hand, shorter documents can better represent mixtures of classifications. If we seek to observe how much a candidate is sampling his or her language from 'religious conservative' versus 'libertarian' ideologies, we may justifiably parse single speeches into its component sentences. This would allow us to model how a candidate switches between two ideological schools of thought in a single speech.

Choosing the unit of analysis for text modeling is task dependent, and requires the analyst to balance between the amount of information provided in a single document (i.e., longer documents provide more information) with the expected nuance of documents themselves.

Text Cleaning In many applications, researchers prefer to strip text of capitalization, punctuation, numerals, and non-typical characters. Under such a text cleaning regime, a document that says "I would like to buy 100 widgets!! Thanks!! :)" would be converted to the string "i would like to buy widgets thanks". Clearly some information is lost in this process, but in many applied settings the information loss is compensated for by simplifying the resulting vocabulary. (In the first instance, the single-word vocabulary would be "I", "would", "like", "to", "buy", "100", "widgets", "!", "Thanks", ":)", while the second would simply be "i", "would", "like", "to", "buy", "widgets", "thanks").

Stemming Stemming simply means reducing words to their linguistic stems, dropping suffixes that do not frequently alter the meaning of the word itself. For example, the words "repeal," “repealed,” and “repealing” convey much the same thing: that a piece of legislation is being, or has been, or should be repealed. Once we know that the author is discussing repealing, the suffix itself provides scant new information. Stemming also helps to reduce the dimensionality of the vocabulary, by collapsing variations on single words into a single term. In other words, instead of counting the three ‘repeal’ words separately, we define a single ‘repeal’ stem and count all versions of that term as instantiations of the same term.

N-Gramming Words do not always convey meaning in isolation. The word ‘pride’ conveys different meaning if combined with ‘gay’ versus with ‘American’. The BOW assumption that words are exchangeable would treat the word ‘pride’ the same, whether it occurs after ‘gay’ or after ‘American.’ To maintain the simplicity of the BOW assumption, but still allow for some local word ordering, we can generate vocabularies of *N-grams*, or $N-$word phrases. To generate bigrams, or two-word phrases, from the string “We should support gay pride movements,” we simply take every two-word phrase that occurs in the text: “We should,” “should support,” “support gay,” “gay pride,” “pride movements.” Each two-word phrase is treated as a unique term, as if these phrases were a single word.

Documents can be represented as collections of unigrams, bi-grams, tri-grams, or higher-order n-grams. Without some principled thinning of the vocabulary, however, this process can dramatically expand the dimension of the vocabulary. Even for the example sentence above, the vocabulary of unique words or word phrases is five unigrams, nine unigrams and bigrams, and twelve unigrams, bigrams and trigrams, and so forth. When applied carefully, however, n-gramming can ease the restrictiveness of the BOW assumptions, by allowing for *local* dependence of words in meaningful multi-word phrases.

Vocabulary Reduction Theoretically, we would like to employ all available information in a text corpus. This may include all unique words, as well as all multi-word phrases, that appear in a given corpus. Unfortunately, this will nearly always be too computationally expensive, and inefficient for predictive purposes. Many words, and most multi-word phrases, convey little substantive meaning, and thus provide little useful information to leverage in modeling situations. In the previously mentioned sentence, for example, all of the two-word phrases except "gay pride" convey little information about the perspective of the author. Including all one- and multi-word phrases as predictors in a statistical model will prove inefficient, computationally expensive and can reduce the accuracy of the resulting model.

Thinning the vocabulary is something of an art. Most commonly, researchers remove terms (either words or multi-word phrases) that do not occur sufficiently frequently in the corpus, or do not occur in a sufficient number of documents. This filters out very rare words, typographical errors, or other unique terms (like social media user names, or other hypertext mark-up) that will almost certainly not assist us in substantive research tasks. The process of thinning vocabulary frequently requires some trial-and-error, to balance efficiency against the potentially useful information conveyed by the inclusion of rarer terms.

Many researchers also choose to filter out *stop words*, or very common words, that convey little information by themselves. Stop words usually include prepositions, functional words like 'the,' 'is,' 'which,' 'that,' and pronouns. One can easily imagine situations where stop words convey necessary information, or else might convey information useful for substantive research. If we concern ourselves with the *gendered* nature of political language, for instance, we would want to preserve the use of gendered pronouns like 'him' or 'his' versus 'her' and 'her's.' On the other hand, if we seek to model the process by which states adapt legislation from other states, we may add functional words, like the names of state-specific agencies or institutions, to the list of stop-words. The ultimate choice is task-specific, though in applied research, it is more common to drop stop words than to retain them.

1.4.2 Preserving Word Order

Word order is essential for language comprehension. By making the exchangeability assumption of BOW models, we potentially ignore valuable information. Further, despite the modeling convenience by ignoring temporal dependence of word use, BOW models require the analyst to make many consequential choices, such as how large of n-grams to generate from a corpus, how to filter out the vast majority of uninformative n-grams, and generally how to represent local word dependence without making the resulting vocabulary too large to be practical.

To set aside the exchangeability assumption requires a different manner of quantitatively representing documents. Since words are no longer exchangeable, the document-term matrix no longer suffices. Instead, we need to construct a matrix representation of documents that preserves word order. Instead of representing a document as a row in a DTM, we represent a single document as a matrix, where each row represents a single word, and rows are stacked in the order of the words in a document.

I will discuss two common ways to encode the row values: one-hot vectors, and distributed word vectors. We can still stem, n-gram, strip punctuation and numbers, and convert all letters to lowercase, before constructing the one-hot or distributed word matrices. Since we preserve word-order, it's less important to n-gram, but we still typically convert to lowercase. The other options are left to the analyst.

One-Hot Word Representation The simplest way to represent a word is one-hot encoding. A one-hot vector for words will have an cell corresponding to every word in the vocabulary, and will show 1 in the cell corresponding to the word in question, and 0 elsewhere. For example, if we are one-hot encoding the document "Our children can achieve great things," the first row of the matrix will have a 1 for the entry corresponding to the word 'our' and 0 elsewhere; the second row will have a 1 for the entry corresponding to the word 'children' and 0 elsewhere; and so forth.

Distributed Word Representation One-hot representations are sparse, but tend to be very high-dimensional. Even in moderately sized corpora, there will tend to be thousands of unique words, meaning that the one-hot vectors will have length in the thousands. We may suspect, however, that the data in V dimensions may be well represented, or summarized, in far fewer (say $L \ll V$) dimensions. Such a suspicion would be well-founded in natural language modeling, since many words will be convey similar information, or be used in similar contexts. In Chapter 2 of this thesis, I will discuss one popular such reduction: neural word embeddings, made famous by Google's `word2vec` model and similar work.

Constructing a distributed representation of a document looks much the same as the one-hot. Instead of each word in the vocabulary being represented by a long and sparse one-hot vector, each word is represented by a dense vector of length L. Typical values for L are between 100 and 500, meaning that we represent each word as existing in some continuous 100 to 500 dimensional space. The document is represented with one row for every word, stacked in the order in which the words appear in the document.

Document Padding In the BOW approach, we represent an entire corpus as a single DTM, with one row for each document. By preserving the order of words, each *document* is represented as a matrix, with as many rows as there are words in the document. An entire corpus, then, will be represented as a three-dimensional array, where we stack all document matrices together.

For such a representation, however, we require each of the matrices to have the same dimension. Every word is represented in the same L dimensions, but each document will contain a different number of words. To fix this issue, it is common to *pad* or *truncate* documents to the same length. If the longest document in our corpus has 50 words, we may force all documents to have 50 rows. Any document that has fewer than 50 words will be padded with some filler word (which isn't a real word, usually just the spacer `</s>`). So for a thirty-word document, the first thirty rows will be populated by the L-dimensional word representations corresponding to those thirty words. The remaining twenty rows will be filled with the representation for the filler word. If some documents are too long, we may set the document length to be somewhere between the shortest and longest, in which case documents over, say, 50 words will be truncated—i.e., we might just cut off any words past fifty.

By preserving word order, we are typically required to specify a unit of analysis to be fairly small. Sentences, tweets, clauses or paragraphs may contain tens to low-hundreds of words, but entire speeches, news stories, chapters or books will contain many thousands to hundreds of thousands of words. (The reader must keep in mind that the number of words in a document is not the number of *unique* words, it is how many total words the document contains.) A single-spaced six-page paper may contain 3000-4000 words by itself. The multi-dimensional array representation for a corpus can get unwieldy very quickly, making it difficult to store the array in memory, much less fit a model to it. As such, longer documents (e.g., chapters of a book) are typically parsed into smaller chunks (e.g., paragraphs) before converting to an array.

2 DEEP NEURAL NETWORKS FOR TEXT CLASSIFICATION

The complications inherent in mining usable data from text are not new to political science. As Krippendorff (2004) and Hopkins & King (2010) note, researchers have been “content analyzing” political text for at least six decades. For much of this period, content analysis required scholars and trained coders to wade through stacks of documents, coding each by hand according to criteria established by the research project. Faster computing and more advanced technical capacity have permitted researchers to engineer tools for automated or semi-automated content analysis (Laver, Benoit & Garry 2003, Grimmer & Stewart 2013). Yet, as always, there must be tradeoffs. No algorithm or statistical model can perfectly process text for every scholarly application. More importantly, purely unsupervised approaches leave little room for theorists’ particular questions and concerns to guide the empirical process. In this article, I propose a supervised and highly customizable model, and present a class of questions it is well calibrated to answer.

This paper addresses an emerging tool in machine learning, known broadly as ‘deep learning’ methods. This set of tools involves the use of neural networks with multiple layers of hidden nodes to learn patterns in data (LeCun, Bengio & Hinton 2015, Schmidhuber 2015)[1]. Multi-layer neural networks offer some strengths that other methods do not. Namely, neural network models provide straightforward ways to account for deep interactivity, interconnectivity and nonlinearities in the data (Pal & Mitra 1992, Belue & Bauer 1995). These strengths are particularly useful when modeling text corpora, since language involves high degrees of interactivity. For example, the phrases ‘climate,’ ‘change,’ ‘climate change,’ and ‘so-called climate change’ all convey different meaning. Dictionary-based methods, which might categorize documents into topics,

---

[1] What makes these models ‘deep’ is the multiple layers of hidden nodes, rather than using a single layer of hidden nodes.

frames, or ideological perspectives based on word occurrence, will fail to account for the interactive effect of combining ‘so-called’ and ‘climate change’ unless the researcher can specify the pattern *a priori* (Iyyer et al. 2014). More sophisticated structural approaches, like latent dirichlet allocation (LDA) or structural topic models (STM) and their variants, might model multi-word (n-gram) phrases (Blei, Ng & Jordan 2003, Blei 2012, Wang, Zhang & Zhai 2011, Roberts, Stewart, Tingley, Airoldi & Others 2013). Deep learning methods offer some natural, and flexible, ways to automatically account for such interaction between words and phrases in natural language (Collobert, Weston, Bottou, Karlen, Kavukcuoglu & Kuksa 2011, Collobert & Weston 2008).

That said, deep learning methods also suffer some weaknesses. This class of models tends to be computationally intensive, especially as the size of the data grows. This inefficiency arises from several sources, the primary of which are the large numbers of parameters the models require us to estimate, and the need to iteratively update parameters in stages (Glorot & Bengio 2010). The flexibility of artificial neural networks (ANNs) also make it extremely easy to overfit to training data, requiring analysts to regularize and ideally—though this can be challenging considering computational efficiency—cross-validate the models (Tu 1996, Zhang 2000, Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov 2014). Finally, ANNs can appear as ‘black box’ tools that provide few ways of assessing the model estimates, aside from the model’s predictive power. Structural models, like regression-based approaches, allow for the analyst to see parameter estimates and assess their face validity. For example, the analyst could view the words that receive the highest weighting vis-à-vis a topic category. The analyst could then evaluate whether the words most closely associated with a topic make sense. ANNs, however, often involve hundreds or thousands of parameters, which interact with each other at the multiple layers of the network. As mentioned above, this has the advantage of accounting for high-order interactivity in the data, but it renders is difficult to evaluate what information the model finds most useful in making predictions.

Many varieties of neural networks exist, some of which alter this architecture in whole or in part. I will discuss a promising extension in the following chapter. In this chapter, however, I focus entirely on fully connected, feed forward networks. I begin by defining feed-forward ANNs, then discuss some issues of model specification, before presenting a short simulation experiment, along with two applications using data on the tone of statements in a presidential debate, and the ideological content of political books.

2.1 Neural Network Models

Artificial neural network models comprise a family of methods based on a stylized understanding of cognitive function. A detailed explanation of brain functioning goes beyond the scope of this paper (and beyond the ability of the author). In broad strokes, brains comprise dense networks of neurons connected by axons and dendrites, the former responsible for sending electrical and chemical signals across the network, and the latter responsible for accepting signals from elsewhere in the network. The responsibility of the neuron is to perform very basic computation on the signals and to decide how and where to pass it through the network. Taken by itself, a single neuron is not capable of learning in any meaningful way. Embedded in a network with millions or billions of other neurons, however, the networked structure provides incredible information processing and pattern learning power.

ANNs loosely replicate this style of learning by connecting individual, simple computations, forming an arbitrarily large and complicated network. Put simply, neural networks are distributed learning models. ANNs take training data as an input, and distribute processing across units called *nodes*. Information is passed between those nodes along *edges* in an attempt to discover (potentially complex) patterns in the data. The information being passed are vectors or matrices of data, or more often, the weighted and transformed vectors.

In this section, I will make these concepts more concrete. I begin with a basic ANN, its structure and how parameters are fit. I then continue to describe richer neural network models, including the ‘deeper’ architectures with multiple layers of hidden nodes.

2.1.1 Neural Network Basics

To begin, I will address some basic terminology. Neural networks aid researchers in learning patterns in data that can map inputs onto outputs. For illustration, see Figure 2.1. *Inputs* (the x's in the figure) are the variables we use to make predictions about the *outputs* (the y). For example, we might seek to model the topics of documents using the words contained in those documents. In that situation, the topic for each document is the output, while we could use vectors of word counts for each document as inputs. In a supervised learning task, which will be the focus of this paper, these topic labels are known.[2] The additional input labeled **1** in Figure 2.1 represents the *bias*, or the equivalent to the intercept in a regression framework. The units of the ANN, represented by circles in the graph, are called *nodes*. Nodes are graphically connected to other nodes via *edges*, which symbolize weights. Inputs are weighted and then combined and potentially transformed at the nodes. If the network has multiple layers of nodes, these 'messages' are then passed forward through the network. If, as in Figure 2.1, the network only has two layer of nodes, the input and the output, the 'message' is weighted to directly form predictions of the output.

Regression Example

Logistic Regression Because this can all seem quite abstract, we can consider a simple case to build the intuition of ANNs. Consider some training data $\{Y, X\}$ with target (i.e., dependent variable) $\mathbf{Y}$ and inputs (i.e., predictors) $\mathbf{X}$. We will index the training cases with $i = (1, 2, ...N)$, and will index the inputs with $p = (1, 2, ...P)$. A typical approach to the problem would be to

---

[2] The contrasting situation is unsupervised learning, where part of the inference task is to learn the topics themselves. The difference between supervised and unsupervised can be thought of as the difference between confirmatory and exploratory factor analysis. In the former setting, the analyst has a theoretically-grounded sense of the underlying dimensions and is trying to use that pre-defined structure to learn from the data; in the latter, the researcher will try to find underlying structure in the data and interpret the results post-estimation. In topic modeling, supervised methods require training data where documents come with hand-labeled topics, while in unsupervised models the documents are 'clustered' by similar language, and the analyst tries to identify the meaning of the clusters after fitting the model.

specify some mapping from inputs to outputs, for example

$$\begin{aligned}y_i =& g(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... \beta_P x_{iP} + \epsilon_i) \\ =& h(\beta_0 + \mathbf{X}_i \boldsymbol{\beta} + \epsilon_i) \qquad (2.1)\end{aligned}$$

This familiar setup involves a linear combination of inputs ($\beta_0 + \mathbf{X}_i\boldsymbol{\beta}$) and some transformation ($h(\cdot)$) of that linear combination. With continuous outputs, we might allow h to simply be the identify function. For binary outputs, we may instead perform a sigmoid transformation, often with the inverse logit (softmax) function or with the standard normal cumulative distribution (probit) function. For this example, we will assume that $y_i \in \{0, 1\}$, and that the link is the inverse logit, i.e., $h(z) = \exp(z)/(1 + \exp(z))$.

This problem can be recast as a simple ANN, shown in Figure 2.1. In this example, the inputs are the predictors in the regression, such that $\mathbf{X}_1$ in the figure represents the length-N vector of inputs for the first predictor. As in a regression framework, the $\mathbf{1}$ input corresponds to a length-N vector of ones for the intercept. The edges represent the parameters in the regression model. The edge connecting $\mathbf{X}_1$ to the node $\hat{\mathbf{y}}$ represents β_1, the edge connecting $\mathbf{1}$ to node $\hat{\mathbf{y}}$ represents the β_0, and so forth. The computation happening at node $\hat{\mathbf{y}}$ is twofold. First, we take the linear combination of the weights and inputs ($\beta_0 + \mathbf{X}\boldsymbol{\beta}$) and then we transform the resulting vector using the inverse logit function.

Fitting We can estimate the weights in this model by minimizing some loss function via gradient descent. For example, in many applied settings, we minimize the sum of the squared error. In that situation, we would initialize the β weights to starting values, take the transformed linear combination of the inputs and weights at the hidden node, and compute the squared differences between the predictions and the target outputs. To minimize the loss, we take the derivative of the loss at the current parameter estimates, and adjust these values in the direction of the gradient. If we superscript current estimates at iteration t, and signify the inverse logit function with $\sigma(\cdot)$, the error and gradient are

$$E^{(t)} = \sum_{i}^{N} \left((y_i - \sigma(\beta_0^{(t)} + \mathbf{X}_i\boldsymbol{\beta}^{(t)}))\right)^2 \tag{2.2}$$

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial \beta} &= \sum_{i}^{N} \sigma(y_i - \beta_0 - \mathbf{X}_i\boldsymbol{\beta}^{(t)})(\beta_0 - \mathbf{X}_i\boldsymbol{\beta}^{(t)})(1 - \beta_0 - \mathbf{X}_i\boldsymbol{\beta}^{(t)})\mathbf{X}_i \\ &= \sum_{i}^{N} \sigma(y_i - \hat{y}^{(t)})(\hat{y}^{(t)})(1 - \hat{y}^{(t)})\mathbf{X}_i \end{aligned} \tag{2.3}$$

where we substitute $\hat{y}^{(t)}$ for the predicted value of y at the current value of the parameters, i.e. $\hat{y} = \sigma(\beta_0^{(t)} + \mathbf{X}_i\boldsymbol{\beta}^{(t)})$, for simplicity.

With a binary output, we might prefer a different loss. The cross-entropy loss function, given in Equation 2.4, offers an alternative.

$$E^{(t)} = -\sum_{i}^{N} \left(y_i \ln \hat{y}_i^{(t)} + (1 - y_i)\ln(1 - \hat{y}_i^{(t)})\right) \tag{2.4}$$

$$\frac{\partial E^{(t)}}{\partial \beta} = \sum_{i}^{N} (\hat{y}_i^{(t)} - y_i)\mathbf{X}_i \tag{2.5}$$

I skip some algebraic manipulation to arrive at Equation 2.5, which we obtain with two applications of the chain rule. The result is rather elegant. As the equation shows, the derivative of the cross-entropy function simplifies to an expression in terms of $(\hat{y}_i^{(t)} - y_i)$, or the difference between the prediction and the target output value. Contrast this with the expression in Equation 2.3, where the magnitude of the derivative depends on $\sigma(y_i - \hat{y}_i^{(t)})$. Referring to Figure 2.2, one can see that the sigmoid function curve levels off at either extreme. In the classification (or logistic regression) scenario, this would correspond to situations where the predicted probabilities are either close to zero, or close to one.

As a result, the gradient descent algorithm will tend to slow when predictions approach one or zero. Similarly, updating can be slower with the squared error loss because errors are simply penalized less. In a situation where the target output for observation i is 1 and the predicted probability $\hat{y}_i$ is 0.2, the loss under squared error would be $(1 - 0.2)^2 = 0.64$. Conversely, the loss under cross-entropy would be $-1(\ln 0.2) = 1.61$. This can be seen more clearly in Figure 2.2, which shows the loss incurred under squared error and cross-entropy.

Algorithm 1 A simple gradient descent optimization procedure.

```
Initialize parameter values
while Making Progress & Not Too Tired do
    Compute predictions $\hat{\mathbf{y}}^{(t)} = \sigma(\beta_0^{(t)} + \mathbf{X}\boldsymbol{\beta}^{(t)})$
    Compute loss $E^{(t)} = \ell(\hat{\mathbf{y}}^{(t)}, \mathbf{y})$
    Update weight proportional to gradient at $\hat{\mathbf{y}}^{(t)}$
end while
```

The fitting procedure for this case, then, involves defining the desired loss function, and then implementing a gradient descent algorithm. This is straightforward to do. Parameters are initialized to starting values, and predictions are generated by feeding the inputs forward through the network. In this case, that simply means multiplying the inputs by the appropriate weights, adding the bias term, and transforming this quantity. We then estimate the error, and update the parameters in the direction of the gradient. In practice, this typically involves evaluating the gradient at the current prediction, and multiplying this quantity by a small 'step size' (e.g., 0.01), to prevent the algorithm from moving too quickly along the error surface and overshooting the optimum. Mathematically, this means we would update parameters according to the rule $\beta^{(\mathrm{t+1})} = \beta^{(t)} + \frac{\partial \ell(y,\hat{y})}{\partial \beta} \times h$, where $\ell(\cdot)$ is the desired loss function, and h is the step size.

2.1.2 Richer Networks

The primary flexibility of ANNs rests in model specification. We can add more hidden nodes, and we can stack multiple layers of hidden nodes. Doing so enhances the model's ability to adapt to complex patterns in the data. In this section, I will introduce how larger and deeper networks can be constructed, and how to adapt the parameter estimation procedure to account for this additional structure.

Adding Nodes

Description We can include additional hidden nodes in the network, as shown in Figure 2.3. In our running example, this essentially transforms the logistic regression problem into an ensemble of regressions. Each input in the network is connected to each of the hidden nodes by a weight. The hidden nodes perform the same transformations as before, but the output from each of the hidden nodes is weighted to generate the final prediction.

The advantage of this set-up is the additional flexibility for the model to learn complex patterns in the inputs. Consider situations where the data contains interactivity between inputs. In a regression setup, the analyst would need to specify appropriate interactions *a priori*. A neural network with multiple hidden nodes can adapt to this interactivity automatically. To see why, consider Figure 2.3. The inputs x_1 and x_2 are connected to hidden nodes $h_1, h_2, ...h_k$, which allows for different combinations of weights for x_1 and x_2. For example, maybe x_1 gets a higher weight when the weight on x_2 is low, and vice versa. This is interactivity of the kind we might specify in a regression model, except that we can learn these patterns without defining them before fitting the model.

Such a network is easily represented using matrix notation. Unlike the zero-node network above, the multi-node network treats the weights $\boldsymbol{\beta}$ as two matrices. Assume that we have P input variables defined for our N observations. We can stack these inputs to a $(N \times P)$ input matrix. The first matrix of weights of dimension $(P \times K)$ connects the input layer to the layer of K hidden nodes. Each hidden node thus represents a $(N \times 1)$ vector, which horizontally stacked yields a $(N \times K)$ matrix.

To generate the predicted y values, we need to postmultiply the hidden nodes by the K weights connecting the hidden layer to the output. This is represented in Figure 2.3 by the edges from the hidden layer to the output y. Again, we represent this stage of the network in matrix form, multiplying the stacked $(N \times K)$ hidden layer by the $(K \times 1)$ weights, yielding $(N \times 1)$ predictions for the N observations.

Estimation via Error Backpropagation We fit a one-layer ANN with multiple nodes in the hidden layer using gradient descent via *back propagation* (Chauvin & Rumelhart 1995, Goh 1995). To see why we require an adaptation of vanilla gradient descent, consider the feed-forward procedure for generating predictions using the network. Inputs are first linearly combined with the first set of weights, and transformed with our chosen activation function. The hidden nodes are then linearly combined with another set of weights connecting them to the output, and again transformed via the chosen activation function. That is, we have two sets of weights to learn: the $(P \times K)$ weights connecting the inputs to hidden layer, and the $(K \times 1)$ weights connecting the hidden layer to the output[3].

Recall that 'learning' the weights just means optimizing the weights vis-à-vis some loss function. In the logistic regression example, we may seek to minimize the cross-entropy loss, meaning that we update the weights in the direction of the loss gradient. With a hidden layer of nodes, however, the loss is a function of both sets of weights. The predictions, and therefore the errors and loss, are functions of the weights connecting the output to the hidden nodes, *which are themselves* functions of the inputs and the weights that connect them to the hidden layer.

To make this point clearer, let us employ some mathematical precision. Assume that we seek to predict a binary output $\mathbf{y}$ of length $N = 100$ using an ANN with one hidden layer of five hidden nodes. We are using 10 inputs, i.e. predictors, for the task, so that our input matrix $\mathbf{X}$ has dimension 100×10 . Assume that the activation function is the sigmoid function, $\sigma(\cdot)$.

Subscripts will index positions in a matrix. For example, $x_{i,p}$ is the value for the i^{th} observation on the p^{th} input. The stacked hidden layer is $\mathbf{H}$, such that $h_{i,k}$ is the value for the i^{th} observation at the k^{th} hidden node, and $\mathbf{H}_{\cdot,k}$ refers to the entire length-N vector for the k^{th} hidden node. Superscripts refer to sets, so that $\beta^{(1)}$ refers to the weight matrix connecting the inputs to the first (and in this case only) layer of hidden nodes, and $\boldsymbol{\beta}^{(2)}$ is the weight matrix connecting the hidden layer to the output. This implies that $\boldsymbol{\beta}^{(1)}$ exists in (10×5) dimensional

---

[3] For further explanations, see Gori & Tesi (1992), Hecht-Nielsen (1989), Werbos (1994), Yu & Deng (2011).

space, and $\beta^{(2)}$ exists in (5×1) dimensional space.

We randomly initialize the weights in the network and feed our inputs forward. First, we compute the value of the hidden layer: $\mathbf{H} \equiv \sigma(\mathbf{X}\beta^{(1)})$. Second, we compute the value of the network output: $\hat{\mathbf{y}} \equiv \sigma(\mathbf{H}\beta^{(2)})$. This second expression explains why we require backpropagation, though it's not immediately apparent. If we substitute in the first expression into the second, we see that $\hat{\mathbf{y}} \equiv \sigma(\sigma(\mathbf{X}\beta^{(1)})\beta^{(2)})$. In other words, the output (and thus the incurred loss from our errors) is a composite function, i.e., a function of a function. In order to take the derivative of the loss with respect to the weights, we need to differentiate the composite function.

Fortunately, the calculus produces an elegant and intuitive optimization algorithm. Imagine we have fed forward our inputs the randomly initialized weights, thereby obtaining our outputs $\hat{\mathbf{y}}$, which are the predicted values of our target variable $\mathbf{y}$. With cross-entropy loss, the total loss for the network is given:

$$\begin{aligned} E &= -\sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \\ &= -\sum_{i=1}^{N} (y_i \log(\sigma(\mathbf{H}_{i,\cdot}\beta^{(2)})) + (1 - y_i)\log(1 - \sigma(\mathbf{H}_{i,\cdot}\beta^{(2)}))) \end{aligned} \tag{2.6}$$

Let us deal only with the $\beta^{(2)}$ parameter matrix for the moment. To obtain the update rule for $\beta^{(2)}$, we differentiate the loss with respect to $\beta^{(2)}$, which requires the chain rule. To wit:

$$\frac{\partial E}{\partial \beta^{(2)}} = \frac{\partial E}{\partial \sigma(\mathbf{H}\beta^{(2)})} \frac{\partial \sigma(\mathbf{H}\beta^{(2)})}{\partial \mathbf{H}\beta^{(2)}} \frac{\partial \mathbf{H}\beta^{(2)}}{\partial \beta^{(2)}} \tag{2.7}$$

To conserve space, I skip some algebraic manipulation, but we can consider each piece individually. The first term takes the derivative of the cross-entropy function w.r.t. the output values $\sigma(\mathbf{H}\beta^{(2)}) - \hat{\mathbf{y}}$:

$$\frac{\partial E}{\partial \sigma(\mathbf{H}\beta^{(2)})} = -\sum_{i=1}^{N} \frac{\partial E}{\partial \hat{y}_i} = -\sum_{i=1}^{N} \left(\frac{y_i}{\hat{y}_i} + \frac{1 - y_i}{1 - \hat{y}_i} \right) = \sum_{i=1}^{N} \frac{\hat{y}_i - y_i}{\hat{y}_i(1 - \hat{y}_i)} \tag{2.8}$$

For the second term, we differentiate the output with respect to the linear transformation input to the sigmoid function:

$$\frac{\partial \sigma(\mathbf{H}\boldsymbol{\beta}^{(2)})}{\partial \mathbf{H}\boldsymbol{\beta}^{(2)}} = \sum_{i=1}^{N}\sum_{k=1}^{K} \sigma(h_{i,k}\beta_k^{(2)})(1-\sigma(h_{i,k}\beta_k^{(2)})) = \sum_{i=1}^{N} \hat{y}_i(1-\hat{y}_i) \tag{2.9}$$

where we drop the summation over k since $\hat{y}_i = \sum_{k=1}^{K} h_{i,k}\beta_k^{(2)}$.

Finally, the third term differentiates the linear combination of $\mathbf{H}$ and $\boldsymbol{\beta}^{(2)}$ w.r.t. each $\beta_k^{(2)}$:

$$\frac{\partial \mathbf{H}\boldsymbol{\beta}^{(2)}}{\partial \beta_k^{(2)}} = \frac{\partial}{\partial \beta_k^{(2)}} \sum_{k=1}^{K} h_{i,k}\beta_k^{(2)} = \sum_{i=1}^{N} h_{i,k} \tag{2.10}$$

Putting these together yields the expression for the gradient of the loss w.r.t. each $\beta_k^{(2)}$:

$$\frac{\partial E}{\partial \beta_k^{(2)}} = \sum_{i=1}^{N} \frac{\hat{y}_i - y_i}{\hat{y}_i(1-\hat{y}_i)}\, \hat{y}_i(1-\hat{y}_i)\, h_{i,k} = \sum_{i=1}^{N} (\hat{y}_i - y_i) h_{i,k} \tag{2.11}$$

We now have our updating rule for each of the $\beta_k^{(2)}$: we will update the weights in proportion to the gradient in Equation 2.11. In other words, $\Delta\beta_k^{(2)} \propto \sum_{i=1}^{N} (\hat{y}_i - y_i) h_{i,k}$.

At this point, we need to perform a similar procedure for the first set of weights, $\boldsymbol{\beta}^{(1)}$. Doing so appears complicated, but will simplify nicely. As before, we need to differentiate the loss, now w.r.t. $\boldsymbol{\beta}^{(1)}$. Recall that $\boldsymbol{\beta}^{(1)}$ is a $(P \times K)$ matrix, such that $\beta_{p,k}^{(1)}$ connects the input vector $\mathbf{X}_{\cdot,p}$ to hidden node $\mathbf{H}_k$, i.e. that $h_{i,k} = \sigma(x_{i,p}\beta_{p,k})$. Let us consider the update rule for one such weight, $\beta_{p,k}$ thus:

$$\begin{aligned} \frac{\partial E}{\partial \beta_{p,k}} &= \frac{\partial}{\partial \beta_{p,k}} \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i) \\ &= \frac{\partial E}{\partial \beta_{p,k}^{(1)}} \sum_i y_i \log(\sigma(\sigma(x_{i,p}\beta_{p,k})\beta_k^{(2)})) + (1-y_i)\log(1-\sigma(\sigma(x_{i,p}\beta_{p,k})\beta_k^{(2)})) \end{aligned} \tag{2.12}$$

Given the nested nature of these functions, we need to yet again apply the chain rule. It can be shown that

$$\frac{\partial E}{\partial \beta^{(1)}_{p,k}} = \sum_i \Big(\frac{\partial E}{\partial \hat{y}_i} \; \frac{\partial \hat{y}_i}{\partial h_{i,k}\beta^{(2)}_k} \; \frac{\partial h_{i,k}\beta^{(2)}_k}{\partial h_{i,k}} \; \frac{\partial h_{i,k}}{\partial x_{i,p}\beta^{(1)}_{p,k}} \; \frac{\partial x_{i,p}\beta^{(1)}_{p,k}}{\partial \beta^{(1)}_{p,k}} \Big) \tag{2.13}$$

Fortunately, we have already computed some of these components in the first stage above, and the ones that we have not are easily derived. I will again skip some manipulations, but it can be readily verified that:

$$\frac{\partial E}{\partial \beta^{(1)}_{p,k}} = \sum_i (\hat{y}_i - y_i)\beta^{(2)}_k h_{i,k}(1 - h_{i,k})x_{i,p} \tag{2.14}$$

It is possible to lose the forest for the mathematical trees, so let us consider the intuitive meaning of Equation 2.14. In deciding how much to update a particular weight $\beta^{(1)}_{p,k}$ connecting the input p to the hidden node k, we end up considering four pieces: (1) the magnitude of the error, given as $(\hat{y}_i - y_i)$; (2) the weight connecting that error to the hidden node k, given as $\beta^{(2)}_k$; (3) the value of the hidden node k, given as $h_{i,k}$; and the value of the p^{th} input, $x_{i,p}$.

All else equal, larger updates for $\beta^{(1)}_{p,k}$ occur when errors are large, or when $\beta^{(2)}_k$ is large, or when $h_{i,k} = \sigma(x_{i,p}\beta^{(1)}_{p,k})$ is large. In other words, we apportion 'blame' for errors to the various weights in proportion to their magnitude. This should make sense. If $\beta^{(2)}_k$ were very small, this would imply that the hidden node $\mathbf{H}_k$ has little influence on the predictions, and thus bears little responsibility for the errors. We therefore apportion even less blame to the parameters $\boldsymbol{\beta}^{(1)}_{\cdot,k}$ that contribute to the value of $\mathbf{H}_k$. Put simply, with great weight comes great responsibility, and we update parameters accordingly.

The above also suggests an iterative approach to updating the weights. If we start by updating the $\boldsymbol{\beta}^{(2)}$ weights, we will already have computed the first term in Equation 2.14 when we move on to updating the $\boldsymbol{\beta}^{(1)}$. After a full backward pass through the network, all weights will have been updated, after which we feed inputs forward, generate predictions and the new loss can be computed.

Algorithm 2 Backpropagation gradient descent algorithm for neural network with one hidden layer.

```
Initialize parameter values
while Making Progress & Not Too Tired do
    Feed forward inputs through the network: (a) $\mathbf{H} = \sigma(\mathbf{X}\boldsymbol{\beta}^{(1)})$; (b) $\hat{\mathbf{y}} = \sigma(\mathbf{H}\boldsymbol{\beta}^{(2)})$
    Compute loss $E = \ell(\hat{\mathbf{y}}, \mathbf{y})$
    Update weights $\boldsymbol{\beta}^{(2)}$: $\Delta\beta_k^{(2)} \propto \sum_{i=1}^{N}(\hat{y}_i - y_i)h_{i,k}$
    Update weights $\boldsymbol{\beta}^{(1)}$ via backpropagation: $\Delta\beta_{p,k}^{(1)} \propto \sum_i(\hat{y}_i - y_i)\beta_k^{(2)}h_{i,k}(1-h_{i,k})x_{i,p}$
end while
```

Computationally, much of this process can be simplified via vectorization and linear algebra. Many software packages (e.g., `R`, Python or Julia) offer fast and efficient matrix operations, often called from lower-level languages (e.g., `C` or `C++`). The scalar representations present a more intuitive explanation, but implementation will nearly always benefit from vectorization.

Adding Layers

To account for additionally complex patterns in data, we can add more layers of hidden nodes to the network (Schmidhuber 2015, LeCun, Bengio & Hinton 2015, Hinton, Osindero & Teh 2006). This is essentially what transforms the network from a basic artificial neural network to a 'deep' neural network. Figure 2.4 represents one such network, with two layers of hidden nodes.

The deep network operates in a similar manner as the one-layer version. Inputs are linearly combined with weights and transformed, yielding N-dimensional vectors for each node in the first hidden layer. These vectors are then linearly combined with the next set of weights and transformed, yielding an N-dimensional vector for each node in the second hidden layer. These are in turn weighted and transformed, as in the one-layer example, to generate predictions for the output y.

In the figure, both hidden layers contain K nodes, but deep neural networks can handle different numbers of hidden nodes in each layer. The number of hidden layers, and the number of nodes in each layer, are parameters that the analyst must specify. As the complexity of the network increases, so too does its ability to learn complicated patterns in the input data. Similarly, we need not assume that the same activation function operates on each hidden layer—or indeed, even on each node within a layer!—but such an assumption makes the mathematical derivation for the optimization routine simpler.

Estimation via Error Backpropagation Fortunately, we need to only make minimal changes to the backpropagation algorithm discussed in the previous section to fit ANN models of arbitrary depth. In fact, we have already seen that gradient of the the one-hidden-layer ANN just requires repeated applications of the chain rule. The same applies for deeper networks. We first differentiate the loss w.r.t. the weights connecting the last hidden layer to the output, then proceeding to update the weights from the previous layer to the last layer, and so forth (LeCun, Bengio & Hinton 2015). The notation over the entire network gets unnecessarily complicated, but viewed iteratively, the process is quite simple.

Before moving on to issues of model specification, I pause to remind the reader about the defining characteristic of these ANN models. Deep or shallow, the models are all *fully connected*. This means that all nodes in the current layer are connected to all nodes in the following layer. Put another way, there is an edge along which we can travel from any node to any node further in the network. Within the feed forward architecture, users can alter the full connectedness by fixing certain cells in the weight matrices to zero, which makes it so that some nodes to not connect to some others. That said, doing such reduces the flexibility of the model to learn complex patterns in the data, and would only be useful if the analyst has strong pre-existing prior beliefs about how input data interact to form predictions of the outcome variable. In such a case, however, other modeling approaches would be more straightforward.

2.2 Model Specification

To summarize the chapter to this point: artificial neural network models can provide flexible methods for learning complex patterns in data. The power of ANNs arises from their ability to adapt to nonlinearities and high orders of interactivity in input data, and map them onto an output of interest. These models can be designed as simply as a traditional regression model, but can expand to form what is essentially a large ensemble of classifiers. In most cases, a sufficiently deep ANN can outperform more traditional statistical methods on adapting to patterns in training data.

Yet applied researchers would then undoubtedly wonder: how 'deep' is sufficient? How many layers of hidden nodes are appropriate? How many nodes should be in each layer? Is there a cost to a model that is built too deep, or not deep enough? What loss or activation function should be used? In short, the model specification for ANNs is daunting. In this section, I discuss some useful metrics for evaluating model performance, and some of the most important parameters a researcher needs to specify when fitting an ANN model. There is no one-size solution to model specification, and the answer in most cases requires some trial-and-error.

2.2.1 Model Evaluation

We need to take care when assessing model specifications and their effectiveness. When fitting regression models, we can nearly always increase model fit *to the training data* by regressing on additional predictors, or regressing on interactive, quadratic or polynomial transformations of the predictors. Yet we also know that these complicated models are undoubtedly overfitting to noise in the data. When comparing model specifications, we need to assess model performance on a data that were not used to train the model parameters. In many scenarios, data can be split into two sets, with somewhere between 50 and 80 percent used for model training (called the *training set*) and between 20 and 50 percent used for model evaluation (called the *test set*). We will typically prefer model specifications that yield the best test-set performance.

In comparing model performances, we can compute any number of predictive accuracy metrics. After fitting a candidate model to a set of training data, we simply use the learned parameters to generate predictions (i.e., complete a feed-forward step) with the test data set, and compare the test outcomes to the network's predictions. For continuous outcomes, the mean-square error or root mean-square error offer popular metrics. For classification problems, the ‘accuracy’, or percent of observations correctly classified, is the most common choice. One typically chooses the modal class probability as the model prediction, and computes the percent of cases correctly classified.

In cases where the researcher is more concerned about particular types of error, he or she can instead select models based on adjusted criteria. For example, if an analyst wished to use an ANN to detect ‘abusive’ language in news story comments, he or she may select a model that minimizes false positives rather than false negatives, since falsely flagging comments may deter user participation more than allowing some false negatives through. If the user wishes to detect rare events, he or she may prefer to minimize false negatives, even if this means the model makes more false positive errors.

2.2.2 Loss & Evaluation

Loss functions, or cost functions, define how we penalize models for inaccuracy. In this section, I cover the two most common loss functions in deep learning methods: cross-entropy and squared error. We have been introduced to these already. These are not the only possible loss functions, simply the most frequently used. They work well in practice, and bring the added convenience of simple differentiation and thus optimization. Analysts are free, of course, to specify other losses, especially if certain inaccuracies bring higher costs. In many such cases, a common loss function can be adjusted to reflect the preference of the researcher.

Multi-Class Cross-Entropy Loss In the running binary-classification example used previously, I employed the cross-entropy loss function. The cross-entropy loss can be generalized to a multi-class problem, in a similar way as we generalize binary to multinomial logistic regression. If we were classifying text documents into one of M categories, the output is now M dimensional rather than one-dimensional. Accordingly, $\hat{\mathbf{y}}_i$ is also an $m-$dimensional vector, where each element is the probability that observation i belongs to category m. If we let $\mathbf{I}(\cdot)$ be the indicator function, taking 1 if the internal argument is true and 0 otherwise, the cross-entropy loss generalizes:

$$\ell(\mathbf{y},\hat{\mathbf{y}}) = -\sum_{i=1}^{N}\sum_{m=1}^{M}\mathbf{I}(y_i = m)\log(\hat{y}_{i,m}) \tag{2.15}$$

where we can easily see that, in the case of a binary classification problem, the expression simplifies to the loss given in Equation 2.4.

Squared Error Loss For continuous-valued outputs, we typically prefer squared-error loss, given in Equation 2.2 for a regression problem, or more generally as

$$\ell(\mathbf{y},\hat{\mathbf{y}}) \;=\; \frac{1}{2}||\mathbf{y}-\hat{\mathbf{y}}||^2 \;=\; \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.16}$$

Note that the 1/2 is constant with respect to the parameters, but simplifies the differentiation of the quadratic term. As discussed previously, the squared-error loss can work with binary or multi-class classification problems. In many cases, it is better to use cross-entropy loss.

2.2.3 Number of Nodes & Layers

We face two competing interests in choosing how ‘large’ of a model to specify. We want a model flexible enough to adapt to complex patterns in the input data. On the other hand, we need to bear three points in mind: 1. Models too large will tend to overfit the training data and generalize poorly. 2. We have a general scientific preference for parsimony—i.e., we desire the simplest possible model that achieves our objectives. 3. Large models require more parameters and thus are more computationally expensive (Hinton, Osindero & Teh 2006). With P inputs, L hidden layers with K_l hidden nodes in each layer, and single target (i.e., a binary outcome), we need to estimate $PK_1 + (\sum_{l=1}^{L-1} K_l * K_{l+1}) + K_L$ parameters for a fully connected model, plus an additional $L+1$ parameters if we include bias (intercept) terms. In fact, the computation required for the network tends to scale quadratically in the number of nodes, i.e., $O(K^2)$ if K is the number of hidden nodes.

Unfortunately, there is no task-agnostic prescription for how many nodes, or how many layers of nodes, is sufficient but not excessive. Some researchers have suggested rules-of-thumb, like starting with one layer, and setting the number of nodes between the number of inputs and outputs. The best general advice, however, is to start with a simple network, and continue expanding the structure until predictive error on the test data stops decreasing (Hinton, Osindero & Teh 2006, Bengio 2012). As Bengio (2012) points out, with proper regularization, the primary cost of building a network too large is computation time, and with modern computing, even fairly complex networks can be fit in reasonable times.

2.2.4 Overfitting

Theoretically, we can add an arbitrarily large number of hidden nodes. With a sufficiently large network, training data can be perfectly determined. Unfortunately, such a network will perform quite poorly on new data. To avoid overfitting, we can introduce some *regularization* to the network. Regularization attempts to constrain model parameters, thereby preventing the model from learning noise in the input data. I will discuss two forms of regularization: L_2 or Ridge regularization, and dropout. In both cases, I provide some guidance, but analysts need to bear in mind that the degree of regularization depends largely on the size of the network specified. Larger networks, combined with more regularization, may both prove flexible and generalizable (Bengio 2012).

L_2 **Regularization** Users of sparse regression techniques will be familiar with likelihood penalty methods such as Ridge regression. In order to constrain model parameters from adapting overmuch to the noise in the input data, the L_2 penalty increases the cost of solutions that involve large weights. The L_2 penalty is simply the sum of the squared weights. If using the cross-entropy loss from Equation 2.4, the penalized cross-entropy loss would simply be:

$$E = -\sum_{i}^{N} \big(y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i) \big) + \frac{\lambda}{2} \sum_{\beta} \beta^2 \tag{2.17}$$

Note that the total loss for any particular solution is now augmented by the sum of all the squared parameters. When minimizing the loss, among two otherwise equivalent solutions, we will prefer the model with smaller average parameter magnitudes. We typically omit the bias (intercept) parameters from this calculation, however. The logic is similar to the reason we do not typically suppress intercepts in regression models. The bias terms account for the mean value of the outputs. Shrinking this parameter does not tend to help generalization, and can often make predictions worse on average.

The λ in Equation 2.17 is a hyperparameter that the user must define. Small values will impose less regularization, which might lead to overfitting (Tibshirani 1996). Higher values will impose heavy regularization, and may limit the model's ability to learn meaningful patterns from data. The L_2 regularization is sometimes known as *weight decay*, since it pushes solutions toward having weights of low magnitude (Collobert & Bengio 2004). The weight decay factor is set as $1 - \frac{h\lambda}{2}$, where h is the *step size* or *learning rate* for the gradient descent algorithm. Defining the regularization parameter in terms of weight decay helps to adjust for sample and step size, which both can affect the degree of regularization. A decent starting value for weight decay is around 0.99, but this of course will need to be tuned.

Dropout Another form of regularization in ANN models is node *dropout*. As the name suggests, dropout is a probabilistic process that randomly selects nodes to 'drop out' or 'switch off' during each stage of the learning process (Srivastava et al. 2014). For each iteration of the optimization routine, nodes are selected with probability p of having their output set to zero for that iteration. This effectively nullifies the incoming and outgoing weights from the node, which in turn makes it harder for the model to overfit. For smaller networks, lower p values are necessary; otherwise, too much of the network's learning ability will be reduced. For larger networks, values between 0.2 and 0.5 are typical.

2.2.5 Activation Functions

In the examples so far, I have discussed the 'sigmoid' activation function. Any differentiable non-linear function can be used, but several have proven useful in applied work.

Hyperbolic Tangent The hyperbolic tangent, or 'tanh' function, maps real-valued inputs onto the interval (-1,1). For some input z, the hyperbolic tangent is defined as

$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \tag{2.18}$$

Rectified Linear Unit The ReLU, or 'ramp' function, is widely used in learning deep network architectures. It is defined as $\text{ReLU}(z) = max(0, z)$. In other words, it takes a real-valued input and either leaves it untransformed, or truncates it at its lower bound of zero. The ReLU activation works well in practice, and can be more computationally efficient for deep networks since it doesn't require computing exponentials like the sigmoid activation function. It also tends to help at learning sparse representations, which is helpful with text analytic models, since most words in a vocabulary aren't useful at modeling most outcomes of interest. ReLU can suffer from problems in training, however, like causing nodes to 'die'—i.e., setting a node's output to zero and never adjusting away from that value.

Soft Rectified Linear Unit To adjust for the hard threshold at zero in the ReLU activation, some researchers propose the use of a 'soft' rectified linear unit, or a *softplus* function. The soft ReLU smooths the sharp boundary at zero by incorporating a small constant adjustment:

$$\text{softplus}(z) = \log(1 + \exp(z)) \tag{2.19}$$

When $z \approx 0$, the output of the soft ReLU function is quite small (for $z = 0.001$, the soft ReLU value is 0.69) but nonzero. Yet the function still strongly (i.e., exponentially) favors positive values. The soft ReLU activation function is also convenient insofar as its derivative takes a recognizeable form: that of the sigmoid function.

2.2.6 Learning Rate

In any gradient-based optimization routine, we need to specify a step size, or *learning rate* in the ANN parlance, to prevent overshooting optima on the gradient surface. If the gradient represents a valley, overshooting simply means stepping too far in a single step, passing the lowest point of the valley and arriving instead on the other slope.

When training ANNs with backpropagation, we move *in the direction of* the loss gradient, but we scale this step by a small number to avoid overshooting. Setting the learning rate too large increases the chance of overshooting, while setting it too small increases the required number of iterations to find a stable minimum, thus increasing computation time. In practice, we decrease the learning rate over the course of training procedure, so that it is larger during earlier iterations (aiding quick convergence) but declines as we approach a minimum (helping us to narrow in without overshooting). Typical learning rates vary between 1.0 and 0.001, but for a standard ANN an initial learning rate of 0.1 serves as a good starting point.

2.2.7 Fitting

The time to fit an ANN (of fixed specification) scales linearly with the number of training observations. In text analysis, we often wish to train on a large number of documents—e.g., on many thousands to many millions of speeches, sentences, tweets or comments. The loss functions, which sum over all observations in the training data, can become computationally expensive with large sets of data. To mitigate this cost, we often prefer *stochastic gradient descent*. The SGD algorithm partitions the data into 'mini batches' on each iteration, and evaluates the loss for a mini batch per iteration. Setting the mini batch size to be large reduces computational efficiency gains over regular gradient descent. Setting the mini batch size to be too small can render the algorithm unstable, resulting in a non-smooth reduction in training loss on each iteration. Typical values for mini batch size vary between 30 and 200, but this parameter too requires tuning during training.

2.3 Simulation Example

At this point, we have discussed ANN models, and the parameters that need to be decided by the user to fit such models. To ground the theory, I present a short simulation study mapping a set of input variables onto a binary output. In this section, I compare a traditional logistic regression to a variety of neural network model specifications. For the purpose of comparison, I simulate data from two data generating processes: one simple DGP, and one with high levels of interactivity between inputs and the output of interest. Specifically, for the straightforward DGP, I simulate $N = 500$ observations, with $y \in \{0,1\}$ and five input variables $X_1, X_2, \ldots X_5$ from the following process:

$$\begin{aligned}
&\mathbf{X} \sim \mathscr{U}(-10,10)\\
&\vec{\beta} \sim \mathscr{U}(-1,1)\\
&y_i \sim \text{Bernoulli}(\sigma(X_i\vec{\beta}))
\end{aligned} \tag{2.20}$$

I also specify an interactive DGP, which allows for a more complex mapping from inputs and the output.

$$\begin{aligned}
&\vec{\beta}, \vec{\gamma} \sim \mathscr{N}(0,0.02)\\
&y_i \sim \text{Bernoulli}\big(\sigma(X_i\vec{\beta} + \gamma_1 X_1^2 + \gamma_2 X_1 X_2 + \gamma_3 X_1 X_2 X_3\\
&\qquad\qquad + \gamma_4 X_4 X_5 + \gamma_5 X_4 X_5 X_6)\big)
\end{aligned} \tag{2.21}$$

To assess in-sample model fit, I consider the percentage of outputs correctly classified, the cross-entropy error (see Equation 2.4), and the mean square error. As mentioned previously, ANN models can easily overfit to data, so I evaluate the models on both prediction on the data used to train the model, and on out-of-sample predictions on 500 new observations.

Results appear in Table 2.2. There are several results to unpack. First, note the equivalence in the results between the logistic regression model and the one-layer, one-node ANN. As I described earlier, the two models are functionally equivalent, and the only difference in the results arises from the different optimization routines. Second, referring to the top panel of the table, note that additional ANN model complexity increases the predictive accuracy on the training data, but actually decreases accuracy in the testing (out-of-sample) data. This is a classic case of overfitting. No ANN model outperforms the logistic regression on the test set when the DGP matches the canonical logistic regression.

Moving to the lower panel of the table, however, we see the advantage of the neural network model. With a more complex DGP, the logistic regression model naturally performs worse, only correctly predicting about 66 percent of training outputs and 56 percent of test outputs. Adding additional nodes to the networks dramatically improves the training predictive accuracy and lowers the mean square error on the training set, just as with the simple DGP. Unlike earlier, though, the ANNs substantially outperform the logistic regression model on the test set. The logistic regression model correctly predicts about 56 percent of test cases, with a mean square error of 0.25. Compare that to a one-layer, ten-node network that correctly predicts 77 percent of test cases, with mean square error 0.18. As before, we also see some evidence of overfitting with additional model complexity. Comparing the training and test performance, a more complicated network specification of three layers with ten nodes each predicts 97 percent of the training cases, but does not outperform the one-layer, ten-node network on the test set.

We might finally want to compare the neural network model to the correctly-specified logistic regression model. If we include all appropriate quadratics and interactions in the logistic regression, the test set percent correctly predicted jumps to 0.79, with mean squared error 0.21. In other words, a logistic regression model with the correct functional form outperforms even the best neural network model.

Seen from that perspective, the neural network results do not appear impressive. But readers should keep in mind that the ANN models are automatically learning patterns in the data. To repeat an earlier point, ANNs are not optimal tools for building explanatory models. The results from the ANN are largely uninterpretable, since inputs are weighted at potentially multiple stages and in multiple combinations. The advantage of ANNs rests largely in their ability to approximate arbitrarily complex functions. In applications, like text analysis, complex patterns exist in data. These patterns are largely unknown *a priori*, making the automated learning ability of ANNs appealing.

2.4 Applications

To show how deep artificial neural networks can be used to model political language, and thereby serve as an important measurement tool for political science, I apply deep ANNs to two supervised learning tasks. First, I train a series of ANNs to extract the sentiment of political expressions drawn from the a 2012 presidential debate between Mitt Romney and Barack Obama. Second, I train ANN models to predict ideology from political treatises.

For each application, I also fit several baseline models, to which I compare the performance of the ANNs. I use a simple naïve Bayes model (Soelistio & Surendra 2015), a penalized logistic or multinomial logistic regression (Nigam, Lafferty & McCallum 1999), and a support vector machine classifier (Diermeier et al. 2012). These three are some of the most commonly used text classification[4] models used in recent political science literature.

2.4.1 Political Tone

The tone of political speech is as important as it is amorphous. We understand intuitively that tone matters. A political leader can react to an interviewer angrily or with good humor, a candidate can speak to our anxieties or to our aspirations, and a reporter can provide a negative or positive context for current events. These are not simply consequential for understanding elite behavior. These choices in tone matter for citizen perceptions and beahavior (Ansolabehere, Iyengar, Crigler, Holbrook, Huckfeldt & Sprague 1999, Kaid 2004, Sheafer 2007, Ridout & Franz 2008, Druckman, Kifer & Parkin 2010, *inter alia*).

Although tone can usually be conveyed and recognized in speech, measuring tone is fraught with complications. Unlike other features we typically extract from text, like topics (e.g., is this person talking about Social Security or immigration?), tone can comprise any of the affective, emotional, or attitudinal images the speaker is trying to impress on his or her audience (Hart, Childers & Lind 2013). Tone will leave measureable linguistic traces, but detecting them proves

---

[4] I am not using any text scaling methods, like the popular Wordscores or Wordfish models.

challenging. For this reason, many scholars have relied solely on hand-coding. Yet as I've already discussed, this imposes limits on our ability to measure data quickly and efficiently.

Most coding of political tone centers on positivity and negativity of language. Though not the only aspect of tone—we could also consider anger, fear, hopefulness, cynicism, and so forth—positivity and negativity are catch-all categories that nevertheless capture quite a lot of what we mean by tone. For example, we might question whether candidates engage in negative campaign advertising (Ansolabehere et al. 1999, Kaid 2004, Geer 2008, Ridout & Franz 2008, Druckman, Kifer & Parkin 2010), or to explain variation in the positivity or negativity media coverage of presidents (Eshbaugh-Soha 2010) or politics writ large (Lengauer, Esser & Berganza 2011). In other words, we often wish to understand how, when, and why political figures employ positive or negative language when addressing the public.

Sentiment analysis, or analyzing the positivity or negativity of language, is a canonical task in computational linguistics and natural language processing. Many text classification methods are evaluated on popular toy datasets, including the IMDB movie review datasets, first used by researchers at IBM and Cornell University (Pang, Lee & Vaithyanathan 2002). The movie review data includes user-written reviews, along with a star rating. The task, then, is often to predict whether each user liked the movie or did not, based on how positive or negative their reviews are.

Sentiment analysis is far less common in political research, perhaps because there are fewer sources of data so conveniently pre-coded. Words that convey tone tend to be task-specific (e.g., 'negative' words in a movie review differ from 'negative' words in a campaign advertisement), so pre-existing data from other domains, or even sentiment dictionaries built for general purposes, may prove less than useful.

I attempt to measure tone in a political context: during a presidential debate between Barack Obama and Mitt Romney. During the 2012 presidential election, researchers with ReactLabs collected moment-by-moment reactions of a large participant pool to the October 3 debate between Barack Obama and Mitt Romney (Julien & Resnik N.d., Boydstun, Glazier, Pietryka & Resnik 2014). The data also includes a time-stamped transcript of the debate. The authors follow Boydstun et al. (2013), parsing the transcript into clauses, and code each clause for topic area (according to the Policy Agendas codebook), the frame employed (moral, legal, economic, safety, bureaucratic, political, patriotic, et cetera) and important for this task, the tone (positive, negative or neutral). The ReactLabs data thus gives us a domain-specific set of hand-coded data. If we can learn patterns that map political language onto political tone, we may be able to use this information to code much larger sets of new textual data.

For the following evaluations, I partition the data into two parts: a training set of 600 observations, or roughly 80 percent of cases, and a test set of 170 observations, or roughly 20 percent of cases. The unit of analysis is the sentence or clause, as parsed by Boydstun et al. (2014). I summarize results using total accuracy, or the number of documents correctly classified, as well as precision and recall. The precision measure is the percent of cases the model classified as 'positive' that were also hand-coded as positive. Recall is the percent of cases hand-coded as positive that the model also classifies as positive. The patterns presented in Table 2.3 remain largely the same for other metrics, such as cross-entropy or mean square error, and are omitted for space.

Text Processing

The data comprise 1179 hand-coded sentences or clauses, categorized as being positive, negative or neutral. Positive expressions are those that convey a positive, optimistic or constructive outlook, while negative expressions are those focusing on problems, or are else critical. For example, Mitt Romney's line "There's some parts of Dodd-Frank that make all the sense in the world" is coded as positive, while his line "Dodd- Frank was passed, and it includes within it a number of provisions that I think have some unintended consequences that are harmful to the economy" focuses on problems, and is thus coded as negative. For this example, I preserve the positive and negative observations only[5].

The results that follow all use the same quantitative representation of the documents. The sentences/clauses are converted to lower-case, stemmed using the popular Porter stemmer (Porter 2001), and broken into uni-, bi- and tri-grams. Before thinning the vocabulary, this text processing routine results in nearly 24,000 unique words or multi-word phrases. To reduce this dimensionality somewhat, I also drop English stopwords and remove any term that does not occur at least twice, and in at least two documents. The resulting document-term matrix has 770 rows (documents) and 1116 columns (unique terms).

The effectiveness of the models can vary considerably based on the text cleaning and thinning regimens employed. I experimented with many different ways to clean and thin the texts. The cleaning procedure above yielded the best results on average across the models. In the results section, I will show some selected results from other procedures, when informative.

Neural Network Setup

I present the results from several model specifications, varying the number of hidden nodes and hidden layers in the networks. I also vary the activation functions used, employing the sigmoid, hyperbolic tangent and soft ReLU functions. For simplicity, I apply the same activation function to all layers, although technically one could experiment with different activation functions at each layer of the network. To speed convergence in the stochastic gradient descent algorithm, I also apply a momentum parameter, which augments the parameter updates at time t by the scaled magnitude of the update at the previous time step. This can speed convergence by updating parameters more when the algorithm is on a steady downward slope on the loss surface. The scaling parameter, called the momentum, is fixed at 0.9 for all experiments. Other parameters for the networks are given in the descriptions below, and were chosen by cross-validation. The results of the models fit during the tuning process are not shown.

- **ANN**$_{0\times0}$: A neural network model with no hidden nodes, and no hidden layers. This model

---

[5] Omitting the neutral category removes about one-third of the statements. The results including the neutral category are not appreciably different from those including it.

is functionally equivalent to a binary dependent variable regression model with cross-entropy loss. Other parameters: learning rate 0.25, mini-batch size 100, momentum 0.9.

- **ANN**$_{1\times1}$: A neural network with one hidden layer containing one hidden node. Other parameters: learning rate 0.25, mini-batch size 100, momentum 0.9.
- **ANN**$_{1\times10}$: A neural network with one hidden layer containing ten hidden nodes. Other parameters: learning rate 0.25, mini-batch size 100, momentum 0.9.
- **ANN**$_{2\times10}$: A neural network with two hidden layers, each containing ten hidden nodes. Other parameters: learning rate 0.25, mini-batch size 100, momentum 0.9.
- **ANN**$_{2\times50}$: A neural network with two hidden layers, each containing fifty hidden nodes. Other parameters: learning rate 0.1, mini-batch size 100, momentum 0.9, dropout ratio on each layer 0.5.

Benchmarks

For comparison, we can consider the performance of several benchmark models:

- **Naïve Bayes** Given the binary label, {positive, negative}, we seek to model 'posterior' probability that a document belongs in the positive or negative class, conditional on the words (or phrases) in the document. Using Bayes' rule and leveraging the exchangeability assumption, this simplifies to the *a priori* probability that the document belongs to a certain class, multiplied by the product, taken over the words in the document, of the words appearing conditional on the class[6]. For more, see the detailed description provided by Soelistio & Surendra (2015).
- **Regularized Logistic Regression** The binary label on a document is modeled as a linear function of the words in the document, passed through the inverse-logistic link function.

---

[6] For example, if the word 'miserable' occurs 50 times out of 5000 total words in all negative documents, and 10 times out of 4000 total words in all positive documents, the probability of observing the word 'miserable' conditional on the document being negative is 50/5000 = 0.01 and 10/4000 = 0.0025 conditional on the document being positive. The product is taken over all the words in the document.

Because most words (or phrases) are not informative when estimating tone, we apply two penalties to the model likelihood: a Ridge, or L_2 penalty, discussed previously; and the least absolute shrinkage and selection operator (LASSO), or L_1, penalty, which seeks to minimize the sum of the absolute values of the parameters. When both penalties are used, the model is known as the elastic net. For more information, refer to Nigam, Lafferty & McCallum (1999).

- **Support Vector Machine** Diermeier et al. (2012) find good performance in predicting the ideology of members of Congress based on speeches from the Congressional Record using support vector machines (Support Vector Machine). The Support Vector Machine optimizes for the vectors that best separate the data based on their labels. For example, the Logistic Regression port Vector Machine optimizes the vectors in the V-dimensional space in which our documents exist that best serve to separate documents with positive versus negative tone (Leopold & Kindermann 2002). For more information, see Hearst, Dumais, Osman, Platt & Scholkopf (1998) or Steinwart & Christmann (2008).

Results

The modeling results are summarized in Table 2.3. The training results are shown in the left panels of the table, while the testing results are shown in the right panels. I present the overall accuracy, or the percent of labels correctly predicted, as well as the precision and recall. Precision is the percentage of ‘positive’ observations correctly classified, and recall is the percent of observations classified ‘positive’ that were truly positive. Perhaps surprisingly, there are no substantial differences between precision and recall for any model, which probably results from the balanced training sample, which had nearly even numbers of positive and negative documents. Bolded entries in the table signify the best performance across all models on each metric.

We care more about out-of-sample model performance, so we will focus on the right panel of the table. In both the training and testing sets, most of the neural network models perform at least as well as the baseline models, shown in the bottom panel of Table 2.3. The $\text{ANN}_{0\times0}$ model, which is functionally equivalent to a logistic regression model, tends to perform worst of the ANNs. (The reader will also note that the results for the $\text{ANN}_{0\times0}$ model are the same across activation functions, since there are no hidden nodes in the model, and thus nothing to activate.) The $\text{ANN}_{2\times10}$ and $\text{ANN}_{2\times50}$ models with Soft ReLU activation give the best overall performance, though there is only a small difference between how these two models perform across the activation functions. Amongst the comparison models, the support vector machine gives the best performance, with 71 percent accuracy in the test set, though even the Support Vector Machine performs roughly six points worse than the deep ANNs.

The results in Table 2.3 also provide a warning about overfitting. If we switch our focus to the left panel of the table, we see that the neural network models achieve better than 90 percent accuracy on the training data. The $\text{ANN}_{2\times50}$ even consistently achieves perfect accuracy on the training data, correctly predicting every training label.

This is disconcerting, but not unexpected, when fitting deep neural network models. As the networks get larger—more hidden nodes, and more hidden layers—they develop the capacity to learn complex patterns in the input data. This includes systematic patterns in mapping language to tone, but also includes noise that does not generalize to new data. This lack of generalization can be seen in comparing the left and right panels of the table, which shows that even models achieving perfect training accuracy get, at most, 77 percent accuracy on the training data. For this reason, a researcher using an ANN, or even one of the comparison models mentioned here, must take care to evaluate model performance on an independent training set.

The results in Table 2.3 reflect a series of parameter tuning for each model specification. To briefly show the consequences of this tuning process, I present Figure 2.5, which shows the learning for three ANNs over the course of their training. The solid line shows accuracy on the training set, while the dashed line shows accuracy on the test set. Panel (a) shows the learning process for the ANN$_{2\times10}$, which evinces a dip in accuracy just before the twentieth epoch of the stochastic gradient descent algorithm. This occurs because the optimization routine has briefly gotten stuck in a local minimum of the loss. Had the algorithm been stopped too early, it would not have reached its ultimate best performance, which is reaches at around the thirtieth training epoch.

Panel (b) shows the consequence of setting the learning rate too high, at 0.35 rather than the better-performing 0.25. The erratic behavior of the algorithm occurs because the optimization routine steps too far at each epoch, making it difficult to make steady progress toward an optimum. This behavior also lengthens the time needed to fit the model. The model in (b) does not reach a steady state until training epoch 130, compared with epoch 30 for the same model specification with learning rate 0.25 shown in Panel (a). Panel (c) evinces much better behavior overall, with a smooth learning pattern, and despite the more complicated architecture, reaches a stable solution in less than 100 epochs.

Finally, we can consider other text cleaning procedures, and whether these affect the performance of the various models. For simplicity, I present the results from the three comparison models, and from the best-performing ANN, the ANN$_{2\times50}$ with Soft ReLU activation. Figures 2.7 through 2.8 give three tokenization schemes: unigrams, bigrams and trigrams. Some experiments with longer phrases yielded poor results, and are not included. Stopwords are dropped in every cleaning procedure, and all words are stemmed.

For each figure, the x-axis shows the percent of the vocabulary ‘thinned’ or removed from the document-term matrix, based on infrequent use. For example, thinning 0 means that all unique words or multi-word phrases are used in the models. Thinning 0.9 means that the 90 percent least-frequent words or multi-word phrases are removed, leaving only the 10 percent most frequently used words in the corpus. The y-axis gives the mean accuracy across 10 random splits of the corpus into 80/20 training and testing sets.

In Figure 2.6, we see a consistent drop in all models’ performance as we thin unigrams from the vocabulary. This presumably happens because the most common English words have already been filtered out by removing stopwords, so by thinning unigrams we end up removing relevant words, even if those words do not occur frequently in the corpus. For unigrams and bigrams in Figure 2.7, however, model performance increases with considerable thinning. Since bigram tokenization returns all possible two-word sequences in a text, the vast majority of bigrams are uninformative, and occur only once or twice in the corpus. Removing these terms removes noise from the model. The same applies even more powerfully to Figure 2.8, which shows a dramatic increase in performance as we filter out the vast majority of the least-frequently occurring and uninformative uni-, bi- and trigrams. Despite the variable performance, however, we see that the the rank-ordering of modeling performance seen in Table 2.3 remains fairly consistent.

2.4.2 Political Ideology

Social scientists have concerned themselves for decades with understanding, explicating and measuring political ideology. Broadly defined as a set of fundamental beliefs about society and its relationship with government, ideology presents unique challenges in measurement. We can never directly observe ideology, left instead to induce the structure of beliefs from outward manifestations like policy or candidate preferences.

In the past several years, political scientists have begun to recognize that a quantitative analysis of rhetoric may help us to measure ideological, political thought. Political speeches or writings allow individuals to express their beliefs on their own terms. Unlike measures based on roll-call votes or interest group endorsements, individuals are not constrained by Congressional or court dockets, the whims of party leaders or even the public's agenda. Political tracts and speeches allow for elites to express the full range of their beliefs in a fairly unrestricted manner. We can surely learn much about an author or speaker's fundamental political beliefs by which topics she chooses to address, and what she says about them.

Sim et al. (2013) introduce the Ideological Books Corpus, a collection of more than 200 books and magazine articles written by political figures and commentators between 2008 and 2012. The researchers hand-code the books by ideology, using a novel hierarchical ideology coding scheme. They code the books as belonging to three broad ideological classes (left, right and center) and eight fine-grained ideological subclasses (mainstream liberal, progressive, religious liberal and socialist on the left; moderate conservative, libertarian, religious conservative, and populist conservative on the right). In the original paper, the authors use the IBC as a training corpus. They train a sparse additive model to extract ideological signals from the text, which they then use to measure the political content of campaign speeches from the 2008 and 2012 presidential elections.Using the IBC, I replicate the measurement task using various ANN models.

Text Processing

The IBC contains just over 200 books, which have been parsed into paragraphs. The resulting corpus has over 170,000 documents, partitioned into three coarse classes and eight subclasses. For the following application, I convert all words to lowercase, replace all numbers with a placeholder token, and I tokenize the corpus for uni-, bi- and trigrams. This yields an initial, unthinned vocabulary size of 14 million unique words or word phrases. Even after removing stopwords and tokens that do not occur in at least 30 documents and at least 5 times in the corpus, we still left with roughly a vocabulary of 28,000 unique terms and 22 million total terms. Table 2.4 shows the breakdown of the corpus by ideological classes and subclasses. Note that some documents are given coarse labels but not subideological labels; only these documents are counted for coarse counts.

Even in sparse matrix format, the resulting document-term matrix ties up nearly 2 gigabytes of memory. To reduce the memory required, and to reduce the time required to fit the following models, I partition the data into random 50,000 document subsets. When partitioning the data, I also ensure that the same number of documents from each ideological class are sampled. This does not match the distribution of documents in the corpus, but it both speeds convergence of the ANNs and makes the model results more stable across random partitions. For each training data partition, I also randomly select a sample of 10,000 documents for the testing set. The testing sets are *not* balanced by their ideological class, which ensures that the models built on balanced inputs can still perform well on out-of-sample data, which in practice will often be unbalanced by class. The baseline models for comparison remain the same, though they are obviously multiclass generalizations of the binary classifiers from the previous example. For the support vector machine classifier, which is inherently a binary classifier without an obvious multiclass generalization, predictions are generated based on a series of binary classifications (e.g., 'libertarian or not,' 'religious liberal or not') and the resulting probabilities are normalized to yield a multiclass probability distribution (for more information, see Manning, Raghavan, Schütze & Others (2008)).

Neural Network Setup

As in the political tone application, I experiment with several ANN specifications. I exclude the $\text{ANN}_{0\times 0}$ and $\text{ANN}_{1\times 1}$ specifications, since their behavior has been exhibited sufficiently in the prior application. Given the additional training data available in the IBC, and the fact that we are attempting to model a multi-class outcome (i.e., we now have multiple ideological subclasses, not just two tone classes), the models I present here are larger than those presented in the tone application. For this application, I specify three ANNs: $\text{ANN}_{2\times 50}$,$\text{ANN}_{2\times 100}$ and $\text{ANN}_{3\times 100}$. For each, I employ the sigmoid activation function (though the soft ReLU evinces very similar results), dropout probability 0.5, and learning rates 0.2, 0.15, and 0.10 respectively. For these experiments, I also decrease the learning rate by a factor of 0.9 every 10 epochs, which can help the optimization routine to move quickly toward a solution, but 'slow down' and not overstep as it approaches a solution.

Results

A summary of the results can be found in Table 2.1. Recall that each model was fit to ten partitions of the data, and each tested using a withheld sample of 10,000 paragraphs from the IBC. The primary estimates in Table 2.1 show the average accuracy across the ten cross-validation partitions. The numbers in brackets give each model's best and worst performance among the ten partitions.

We note, again, that the ANN models perform quite well compared to the baseline models. The $\text{ANN}_{3\times50}$ model performs best overall, obtaining an average accuracy of 68 percent, with a best performance of 72 percent. The $\text{ANN}_{3\times50}$ model achieves comparable success to recent methods published in the computer science literature (Sim et al. 2013, Iyyer et al. 2014). By comparison to the political tone application, these numbers may seem disappointing. Yet compared with tone, ideology is a much more complicated concept. The differences between ideological groups can be subtle, and the documents in the corpus address many different topics, issues and preoccupations (Freeden 2003, Putnam 1971, Apter 1964). Though language gives us an optimal window into these complex beliefs and how they are expressed, developing automated machinery to detect ideological perspective is quite challenging (Iyyer et al. 2014, Acree et al. 2014).

Figure 2.9 shows the mean classification precision for each of the eleven ideological classes and subclasses, using the $\text{ANN}_{3\times50}$ model across the ten cross-validation partitions. In many classes and subclasses, the model correctly classifies 66 to 73 percent of the testing documents. The figure also shows some expected and encouraging patterns. Darker colors represent a higher proportion of test documents belonging in each cell. We note two darker blocks of (off-diagonal) cells: a block among the conservative classes, and a block among the liberal classes. This means that when the model mis-classifies a document, it more often than not still correctly predicts that the document was written by a liberal or conservative author, even if the predicted fine-grained ideological label is incorrect. Though not immediately apparent in the figure, the only case of high mis-classification across the left-right divide occurs between the two religious classes. Fifteen percent of religious conservative test documents were mis-labeled as belonging to the religious liberal subclass, and thirteen percent of religious liberal test documents were mis-labeled as belonging to the religious conservative subclass. Both of these are many times higher than the mis-classification rates for the other subclasses.

The results show a similar capacity to overfit to the training data, as we have seen before. The best-fitting model to the training data is the largest model fit in this experiment, yet on the training set the $\text{ANN}_{3\times100}$ model performs no better than the elastic-net logistic regression or the support vector machine classifier. The $\text{ANN}_{3\times50}$ model, by contrast, achieves slightly worse performance on the training set, but outperforms all other models on the test set.

Taken together, the ANN model performs quite well compared to the benchmarks, and compared to other published results. The classification precision is fairly even across the subideological classes, with the exception of the poor performance on classifying the centrist category, which has been noted to be notoriously difficult in previous research (Sim et al. 2013, Acree et al. 2014).

2.5 Discussion

Artificial neural networks offer a flexible, powerful tool to political science scholars hoping to leverage text in their research. Neural networks and deep learning methods have been used for many years in the machine learning scholarship, with impressive results when applied to varied natural language processing tasks. Despite this, ANNs have not featured in political research. In this paper, I introduce how ANNs can be specified, fit and applied to text classification tasks for political corpora. In a simulation exercise, and in two applications with political debate and ideological books data, ANN models outperform benchmark models commonly used in the current literature.

Though promising, the neural network approach suffers some weaknesses, perceived or otherwise. At the most basic level, ANNs represent a paradigm foreign to most applied political research. Yet as I hope to have made clear, at their core ANNs share much in common with more traditional models. Clearly, ANNs are ill-suited to testing theoretical models in substantive research. The results of ANNs are largely uninterpretable. The high degree of interactivity, combined with the frequent lack of unique solutions to overspecified models, render it inadvisable to rely on ANNs for theory evaluation. In measurement tasks, however, minimizing errors and maximizing precision are important goals. ANNs achieve competitive performance on this count, and thus should be considered promising tools for text analytic and measurement tasks.

The reader may also worry about the seeming complexity of the model. As the third section of this paper lays out, there are many parameters to be specified by the user, and this process will nearly always require tuning on the analyst's part. This hardly differs from other methods, however. Any useful classifier has parameters (regularization parameters, loss functions, kernel functions, et cetera) that require tuning and experimentation. Recently published software packages, like the `mxnet` framework available in multiple languages and even callable on mobile devices, can make many common specifications fairly straightforward to estimate efficiently (Chen, Li, Li, Lin, Wang, Wang, Xiao, Xu, Zhang & Zhang 2015).

Artificial neural networks are not a one-size solution to challenges in extracting meaningful information from political texts. Specified poorly, neural network models can fail to converge, or can seemingly perform well on training data, only to fail to generalize when applied to new data. By themselves, though, ANNs can achieve competitive results, and can prove quite useful to researchers interested in using text data to answer substantive questions. Further, as I discuss in the following chapter, ANNs serve as the basic engine for more advanced and powerful models.

2.6 Tables

Table 2.1: PREDICTIVE ACCURACY ON THE IDEOLOGICAL CLASSIFICATION TASK

Model	**Training**	**Testing**
Activation: *Sigmoid*	Accuracy	Accuracy
$\text{ANN}_{2\times50}$	0.85	0.63
	[0.84,0.87]	[0.59,0.70]
$\text{ANN}_{3\times50}$	0.90	**0.68**
	[0.88,0.91]	[0.69,0.72]
$\text{ANN}_{3\times100}$	**0.94**	0.63
	[0.90,0.97]	[0.55,0.71]
Comparison		
Naïve Bayes	0.67	0.53
	[0.55,0.69]	[0.52,0.55]
Logistic Regression	0.69	0.63
	[0.67,0.70]	[0.61,0.64]
Support Vector Machine	0.72	0.65
	[0.70,0.73]	[0.63,0.66]

Each model was fit to ten cross-validation partitions of the data, and then evaluated on a random sample of 10,000 withheld documents. Bolded numbers show the highest average predictive accuracy within the training and testing sets. Numbers in brackets give the worst and best model performance on the ten cross-validation partitions.

Table 2.2: COMPARING REGRESSION TO NEURAL NETWORKS

Model	**Training**			**Testing**		
Simple DGP	Accuracy	Cross-Entropy	MSE	Accuracy	Cross-Entropy	MSE
Logistic Regression	0.93	110.93	0.05	0.89	126.34	0.08
ANN 1 layer, 1 node	0.90	112.62	0.07	0.89	124.37	0.08
ANN 1 layer, 5 nodes	0.94	120.06	0.05	0.88	286.51	0.11
ANN 2 layers, 5 nodes	0.96	483.76	0.04	0.85	1021.68	0.13
Complex DGP						
Logistic Regression	0.66	314.42	0.22	0.56	318.25	0.25
ANN 1 layer, 1 node	0.66	311.83	0.22	0.57	327.36	0.25
ANN 1 layer, 5 nodes	0.70	291.08	0.20	0.62	348.04	0.25
ANN 1 layer, 10 nodes	0.75	269.65	0.17	0.77	448.06	0.18
ANN 2 layers, 5 nodes	0.90	239.93	0.08	0.72	1094.32	0.26
ANN 2 layers, 8 nodes	0.96	291.65	0.04	0.75	1170.27	0.22
ANN 3 layers, 10 nodes	0.97	270.57	0.03	0.77	2139.35	0.23

The top panel shows results fitting the models to data generated with the simple data generating process (DGP) given in Equation 2.20. The bottom panel shows the results from fitting the models to data generated with the complex DGP given in Equation 2.21. The training results are show model performance on the training set, or the set of data used to train the model. The test set results give performance on a held-out set used to evaluate the models. When the relationship between the inputs (the independent variables) and the outputs (the dependent variable) is a straightforward transformed linear combination of inputs and weights, the logistic regression model performs well on both the training set and the test set. The deeper neural network models perform better, however, when the data contains more interactivity between the inputs. We see clear evidence of overfitting, however, with the deeper networks. The ANN with three layers correctly classifies 97 percent of the training cases, making it the best model on the training set. But that model only correctly classifies 77 percent of the test cases, a result matched by the far simpler ANN with one hidden layer of 10 hidden nodes.

Table 2.3: Comparing Models for Text Classification

Model	**Training**			**Testing**		
Activation: *Sigmoid*	Accuracy	Precision	Recall	Accuracy	Precision	Recall
ANN$_{0\times0}$	0.94	0.93	0.95	0.62	0.62	0.54
ANN$_{1\times1}$	0.94	0.94	0.95	0.62	0.66	0.65
ANN$_{1\times10}$	0.96	0.93	0.93	0.73	0.72	0.73
ANN$_{2\times10}$	0.98	0.93	0.98	0.75	0.76	0.76
ANN$_{2\times50}$	0.99	0.98	0.99	0.75	0.76	0.74
Activation: *Tanh*						
ANN$_{0\times0}$	0.94	0.93	0.95	0.62	0.62	0.54
ANN$_{1\times1}$	0.92	0.94	0.94	0.65	0.69	0.60
ANN$_{1\times10}$	0.93	0.94	0.94	0.72	0.72	0.72
ANN$_{2\times10}$	0.97	0.92	0.96	0.76	**0.77**	0.76
ANN$_{2\times50}$	**1.00**	**1.00**	**1.00**	0.75	0.74	0.74
Activation: *Soft ReLU*						
ANN$_{0\times0}$	0.94	0.93	0.95	0.62	0.62	0.54
ANN$_{1\times1}$	0.94	0.94	0.95	0.64	0.64	0.70
ANN$_{1\times10}$	0.96	0.93	0.93	0.72	0.72	0.72
ANN$_{2\times10}$	0.98	0.93	0.98	**0.77**	**0.77**	0.76
ANN$_{2\times50}$	**1.00**	**1.00**	**1.00**	**0.77**	0.76	**0.78**
Comparison						
Naïve Bayes	0.82	0.83	0.81	0.59	0.58	0.59
Logistic Regression	0.84	0.84	0.83	0.62	0.63	0.62
Support Vector Machine	0.89	0.88	0.90	0.71	0.72	0.71

The predictive accuracy for tone, based on the hand-coded presidential debate transcript data from ReactLabs. Accuracy is the percent of observations correctly classified. Precision is the percent of cases classified as being 'positive' over the total number of positive cases; recall is the percent of 'positive' documents that were so labeled. The training set is a random sample of 600 documents used to train the models; the testing set is the remaining 170 documents not used for training. Bolded entries signify the best performance across all models on each metric.

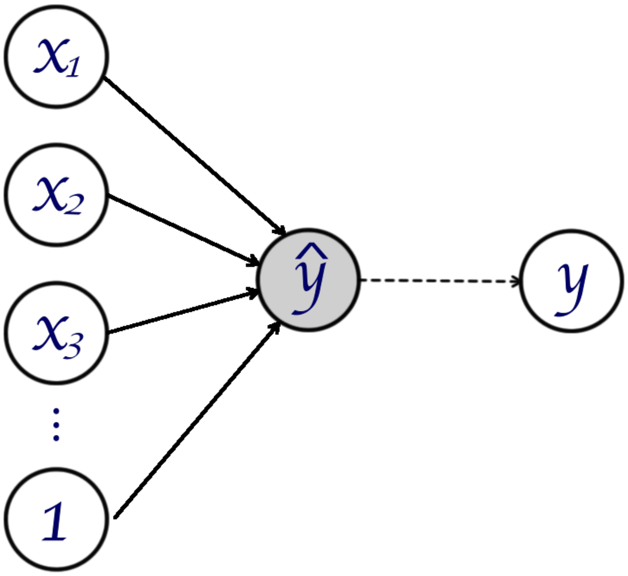
Table 2.4: SUMMARY OF THE IDEOLOGICAL BOOKS CORPUS

Ideology	**Documents**	**Tokens**
Center	6227	0.4m
Left	0	0
Mainline Liberal	15226	1.3m
Socialist	6027	0.7m
Religious Liberal	5367	1.1m
Progressive	27805	3.3m
Right	28323	4.2m
Moderate Conservative	8484	1.3m
Religious Conservative	6648	1.1m
Populist Conservative	7611	1.3m
Far Right	48799	5.2m
Libertarian	11566	1.7m

The number of documents (paragraphs) belonging to each ideological class, and the approximate number of total filtered uni-, bi-, and trigrams (in millions). Some documents were assigned to the Right class without being given an ideological subclass label, but all documents from the Left were given an ideological subclass label.

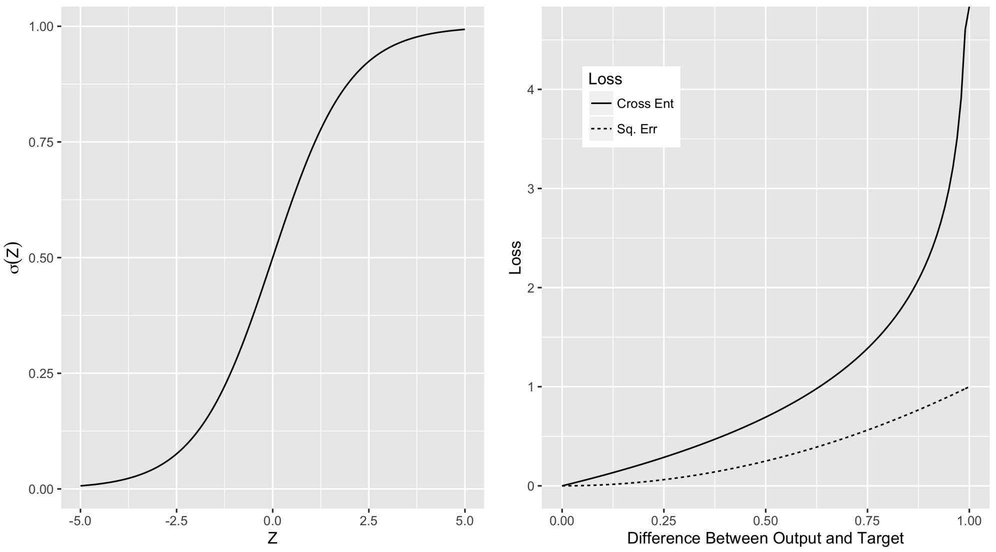
2.7 Figures

Figure 2.1: A BASIC ARTIFICIAL NEURAL NETWORK



A basic neural network with inputs x, an output prediction $\hat{y}$, and for clarity the true target, y, which represents the actual value of the dependent variable we seek to predict.

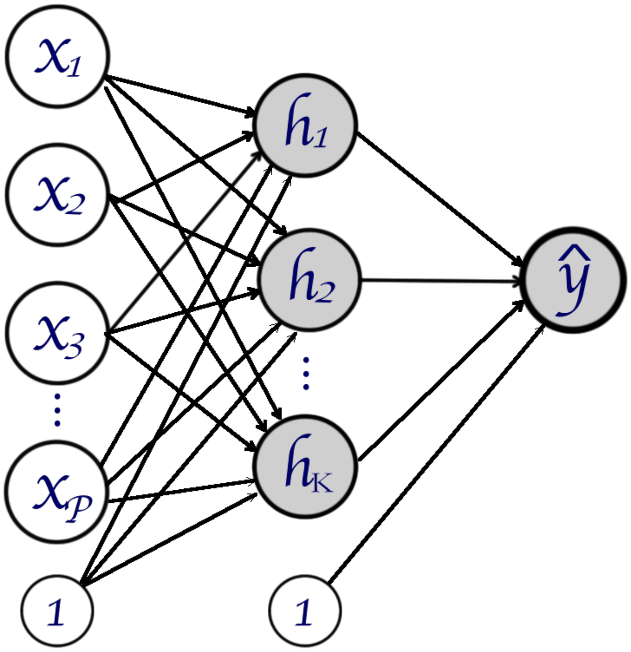
Figure 2.2: Sigmoid Activation & Cross-Entropy Loss



(a) (b)

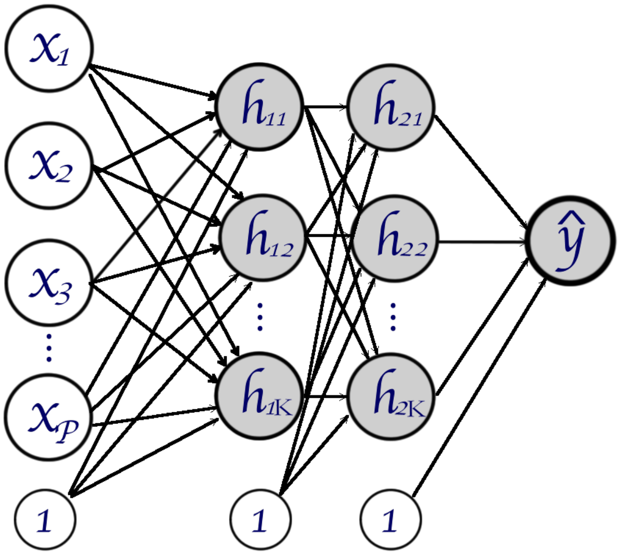
Visualizing the sigmoid activation cross-entropy loss functions. (a) The sigmoid (inverse logit) function. (b) Cross-entropy (solid) applies a larger penalty than squared error loss (dashed), especially for predictions far from the target value.

Figure 2.3: A One-Layer Basic Artificial Neural Network



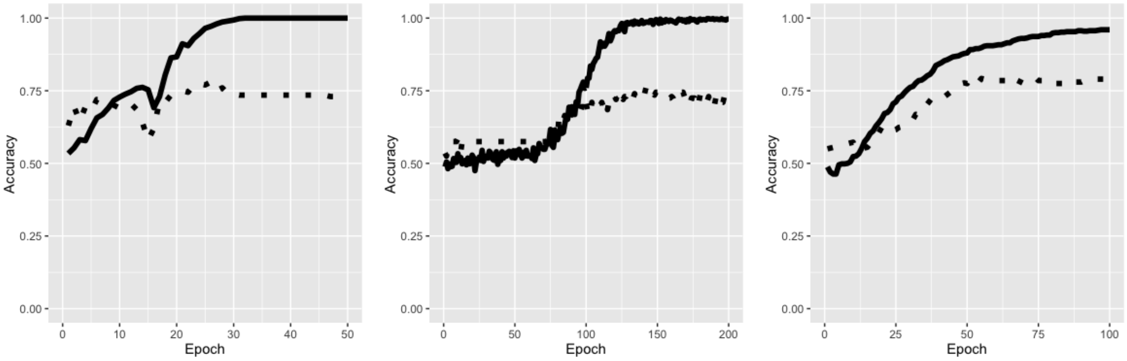
Graphical representation of a one-layer neural network. A neural network with inputs x, k hidden nodes $h_1^{(1)}, h_2^{(1)}, \dots h_{k^{(1)}}^{(1)}$ in the hidden layer, and an output y.

Figure 2.4: A ONE-LAYER BASIC ARTIFICIAL NEURAL NETWORK



Graphical representation of a two-layer neural network. A neural network with inputs x, k hidden nodes $h_1^{(1)}, h_2^{(1)}, \dots h_{k^{(1)}}^{(1)}$ in the first layer and $h_1^{(2)}, h_2^{(2)}, \dots h_{k^{(2)}}^{(2)}$ in the second layer, and an output y. In this figure, both layers contain the same number (K) of nodes, i.e. $k^{(1)} = k^{(2)} = K$, but models can be specified with a different number of nodes in each layer.

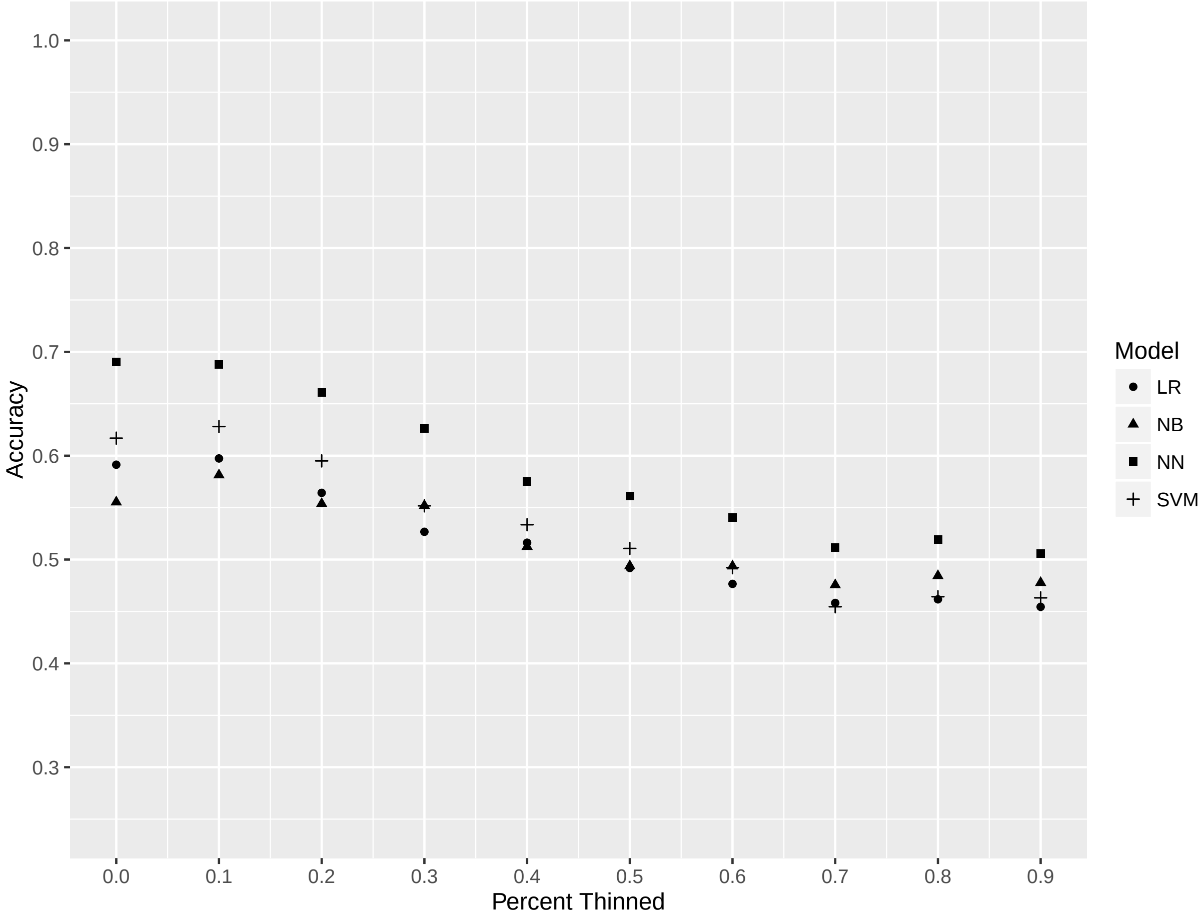
Figure 2.5: THE LEARNING PROCESS OF THREE ANN MODELS.



(a) (b) (c)

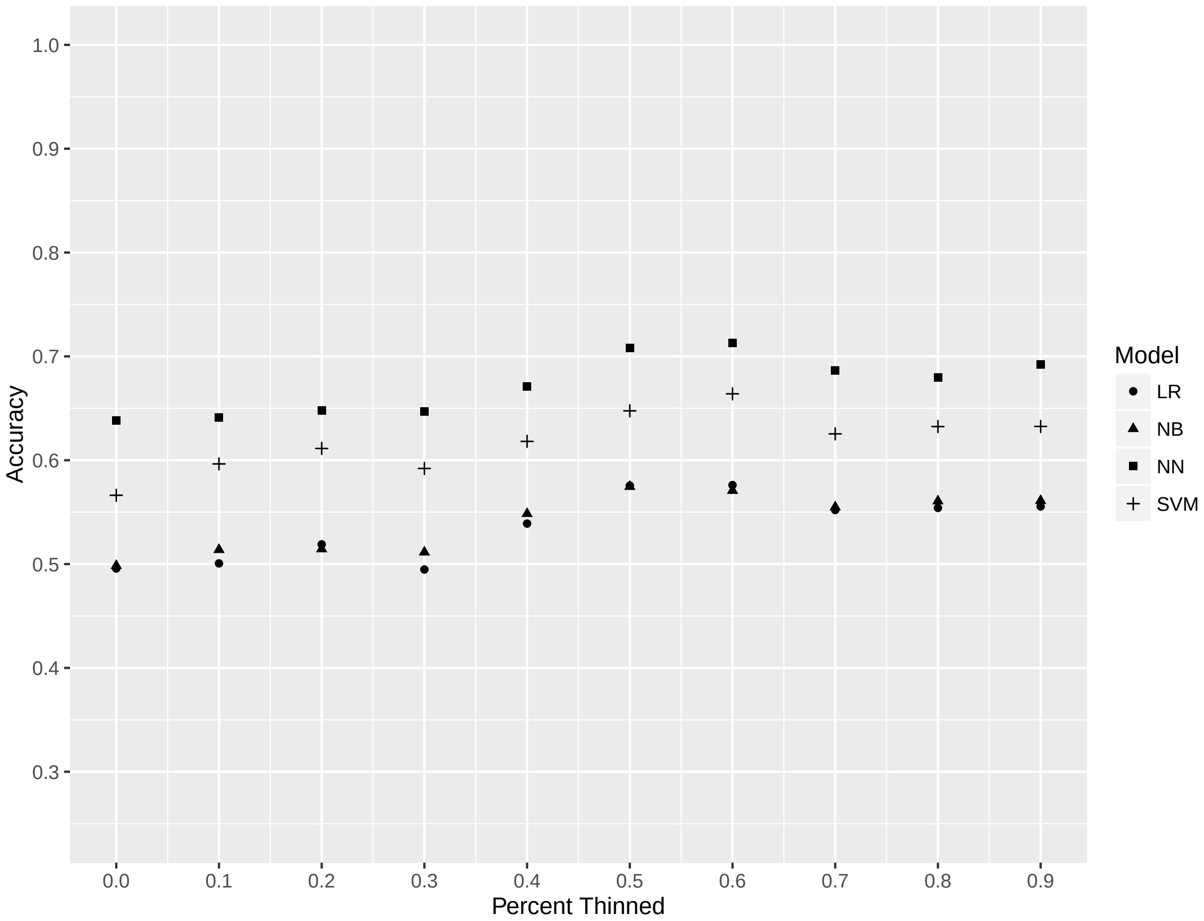
The learning process of three ANN models. The x-axis shows the epoch of the stochastic gradient descent algorithm; the y-axis shows the accuracy. Solid lines represent the training data, while the dashed lines represent the test data. (a) The $\text{ANN}_{2\times 10}$ model, which shows a dip in accuracy where the network gets stuck at a local minimum. (b) The $\text{ANN}_{2\times 10}$ model, with learning rate 0.35. This learning rate is too high, which results in the model stepping too far on most iterations, seen through the jumpy, uneven progress toward convergence. This also slows convergence, requiring more than 100 epochs to reach a stable accuracy. (c) The $\text{ANN}_{2\times 50}$ model, showing smooth convergence and a narrower gap between training and testing performance. All models presented use the sigmoid activation function.

Figure 2.6: THE EFFECT OF THINNING THE UNIGRAM VOCABULARY ON MODEL PERFORMANCE



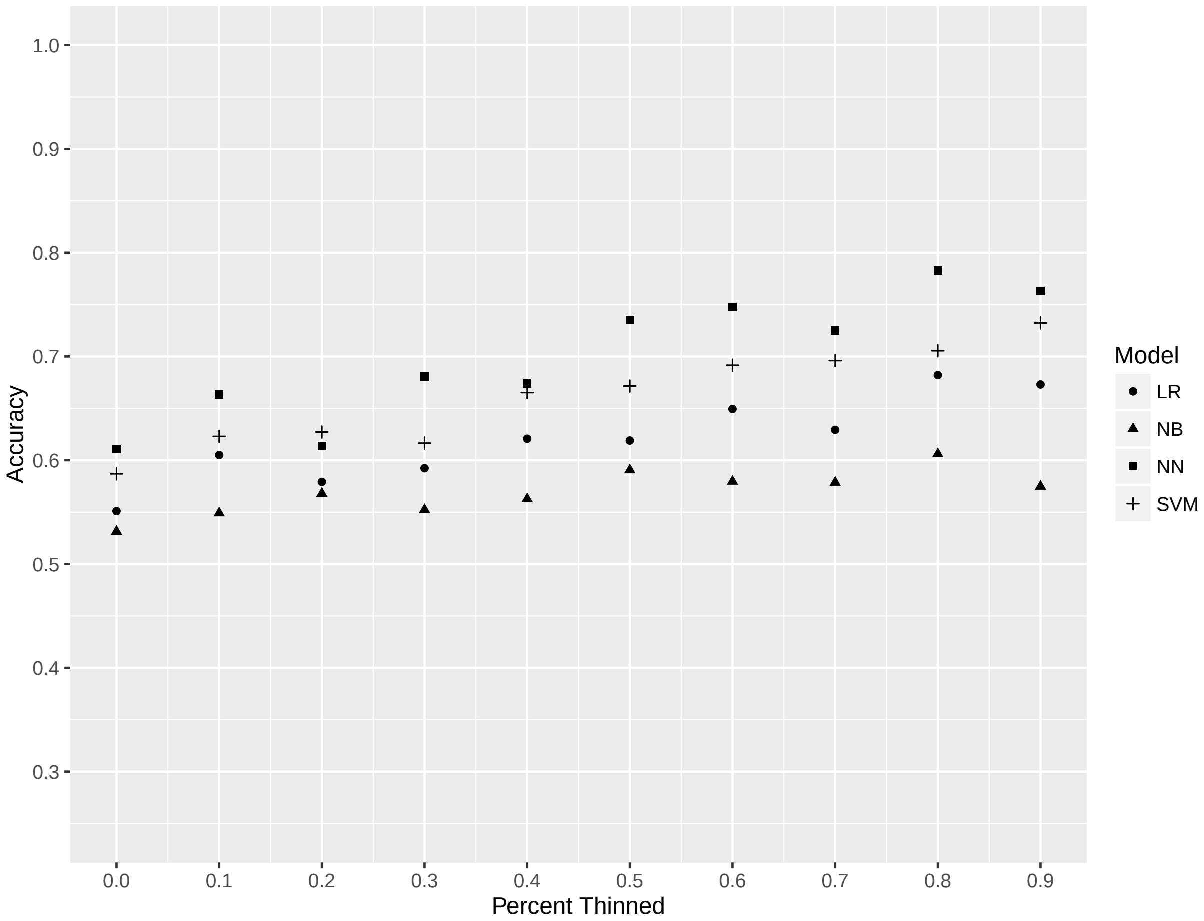
The y-axis gives the percent of the vocabulary thinned based on infrequent use in the corpus. Thinning of 0 means no words or word phrases are thinned, while 0.9 means the 90 percent least-occurring words or phrases are thinned. Logistic Regression is the elastic-net logistic regression, Naïve Bayes is the naïve Bayes model; SVM is the support vector machine model; and NN is the two-layer, 50-node neural network (NN2 × 50). (a) Shows unigrams only, (b) shows unigrams and bigrams, and (c) shows unigrams, bigrams and trigrams.

Figure 2.7: THE EFFECT OF THINNING THE UNIGRAM AND BIGRAM VOCABULARIES ON MODEL PERFORMANCE



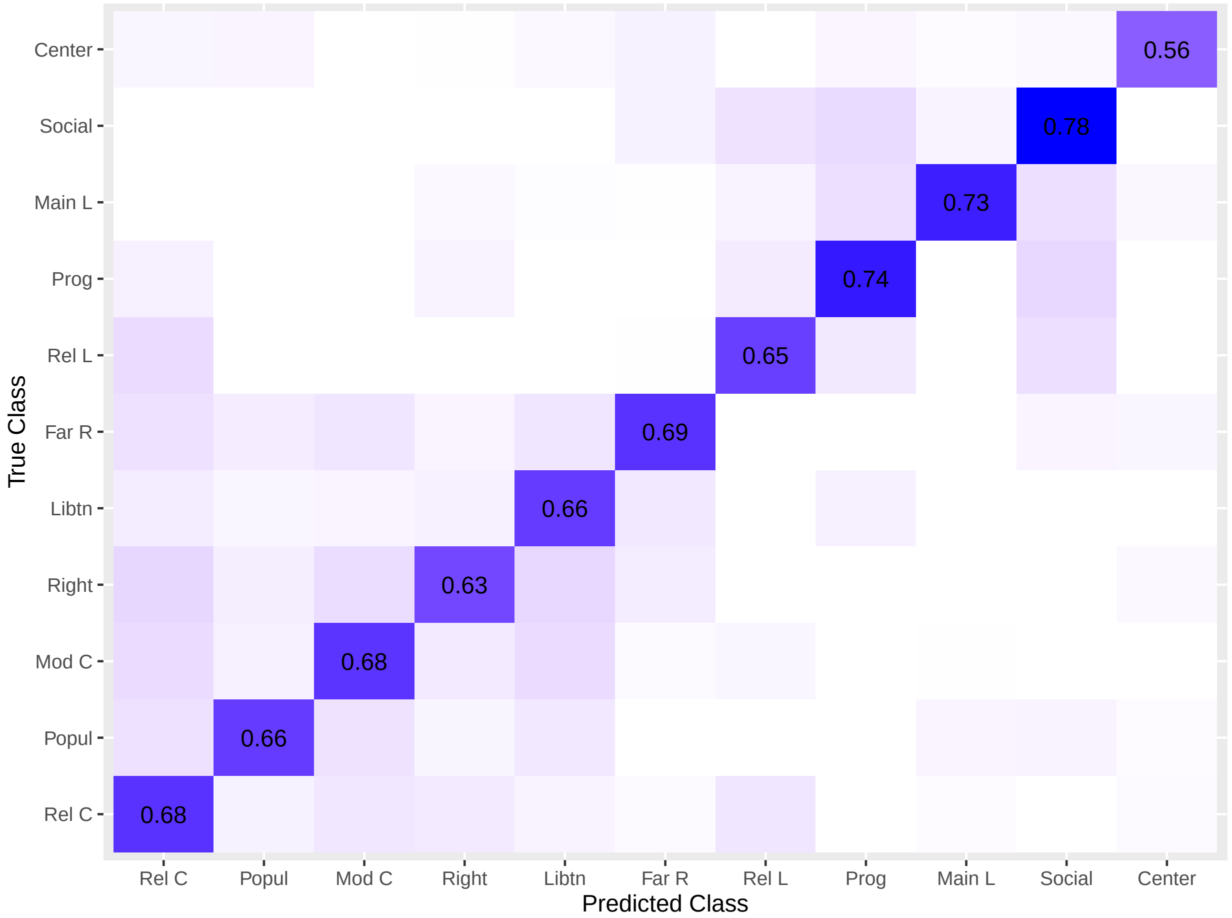
The y-axis gives the percent of the vocabulary thinned based on infrequent use in the corpus. Thinning of 0 means no words or word phrases are thinned, while 0.9 means the 90 percent least-occurring words or phrases are thinned. Logistic Regression is the elastic-net logistic regression, Naïve Bayes is the naïve Bayes model; Support Vector Machine is the support vector machine model; and NN is the two-layer, 50-node neural network (NN2 × 50). (a) Shows unigrams only, (b) shows unigrams and bigrams, and (c) shows unigrams, bigrams and trigrams.

Figure 2.8: THE EFFECT OF THINNING THE UNIGRAM, BIGRAM, AND TRIGRAM VOCABULARIES ON MODEL PERFORMANCE



The y-axis gives the percent of the vocabulary thinned based on infrequent use in the corpus. Thinning of 0 means no words or word phrases are thinned, while 0.9 means the 90 percent least-occurring words or phrases are thinned. Logistic Regression is the elastic-net logistic regression, Naïve Bayes is the naïve Bayes model; Support Vector Machine is the support vector machine model; and NN is the two-layer, 50-node neural network (NN2 × 50).

Figure 2.9: Mean Cross-validation Predictive Precision for Subideological Class.



Means are taken across the ten cross-validation partitions of the Ideological Books Corpus using the $\mathrm{ANN}_{3\times 50}$ model. The x-axis gives the predicted class, and the y-axis gives the true class according to the IBC. Cell entries along the diagonal give the percent of cases correctly classified into each class.

6

3 DISTRIBUTED WORD REPRESENTATIONS FOR POLITICAL TEXT

A common challenge in using text in quantitative research is the sheer complexity of natural language. In nearly all political research to date, words or word-phrases are treated atomically, or as independent members of a vocabulary. This representation, while powerful, has ignored the natural interconnectedness of language (Harris 1954). Words carry semantic meaning—i.e., the definitional meaning of words and phrases—and syntactic meaning—i.e., the role of words in forming syntactically proper and meaningful expressions. Words can thus be similar to certain words and different from others, either in terms of what the words mean or the role they play in language. Researchers have long proposed solutions to these problems: modeling with n-grams instead of single words, stemming to remove pluralization and tenses, clustering words by part of speech, and so forth.

In recent years, some scholars have proposed methods for capturing the semantic and syntactic information in natural languages (Bengio, Schwenk, Senécal, Morin & Gauvain 2006, Mikolov, Chen, Corrado & Dean 2013, Lai, Liu, Xu & Zhao 2015). This class of methods, known as *word embeddings* or *distributed word representations*, seek to take atomic word representations and reduce them to a meaningful, lower-dimensional space. The resulting representation of a word provides key information about how the word is used in natural language. Words with similar representations will tend to be semantically or syntactically similar, meaning that the word representations naturally capture the relationships between words. In 2013, researchers with Google published a highly efficient method for generating such representation, known as the `word2vec` model. Since 2013, word representations have been shown to achieve state-of-the-art results in many natural language processing tasks.

To date, these methods have not been leveraged in political science research. Part of the challenge in using distributed word representations is that many traditional methods, such as topic models, support vector machines, regularized regression models, inverse regression or naïve Bayes models, do not easily integrate the information provided by distributed word representations. Further, it is not clear how to combine word information to represent entire documents.

In this chapter, I introduce a common distributed word embedding model, fit the model to a corpus of political books, and present several examples of how the word embeddings capture semantic and syntactic information. I then experiment with several ways to use word embeddings for document classification tasks. I pay particular attention to a deep learning method, convolutional neural networks, which combines word embeddings together with word order, to achieve very high classification accuracy. Convolutional neural networks, or CNNs, are to date unused in the political science literature, to the best of my knowledge, but offer a promising tool for text analytic research.

3.1 Distributed Word Representations

We start from the traditional atomic representation, or a one-hot encoding, of words in a corpus. For each unique word in a corpus, there is a corresponding vector of length V, where V is the number of unique words in the corpus. The one-hot vector is populated by zeros in all cells but one. If the first element of the one-hot encoding corresponds to the word 'president,' and there are ten unique words in the vocabulary, then the one-hot embedding for 'president' would be `[1,0,0,0,0,0,0,0,0,0]`. Note two important features of the one-hot representation: it is very sparse, and the vectors have no relationship to each other. To wit:

president	`[1,0,0,0,0,0,0,0,0,0]`
executive	`[0,1,0,0,0,0,0,0,0,0]`
potato	`[0,0,1,0,0,0,0,0,0,0]`

We would think that 'president' and 'executive' would be similar, yet in their one-hot encoding there is zero correlation between them. There is no calculation on these vectors that would lead us to conclude that 'president' was any more similar to the word 'executive' than it is to the word 'potato.' Further, this type of encoding tends to be very high-dimensional, as most corpora will have many thousands of unique words.

Word embedding methods are, at the most fundamental level, dimensionality reduction techniques. Much like principal components analysis or exploratory factor analysis, word embedding models aim to take very high dimensional data and summarize them in a much lower-dimensional space. As with tools more traditionally used in political science, the goal is to distill the information provided by the data, and in the process uncover latent patterns that are too obscure, noisy or high-dimensional in the original data. Survey respondents' answers to a single survey item are surely informative, but we can usually glean more information about respondents' attitudes by searching for patterns in multiple responses. By the same token, atomic word representations carry some information, but latent patterns among words can prove more informative.

We begin with the *distributional hypothesis*, proposed by Harris (1954), which asserts that the meaning of words can be best understood by the context in which they appear. Or, as Firth (1957, 11) more colorfully put it, "You shall know a word by the company it keeps." Words that tend to occur in similar contexts, such as 'smart' and 'intelligent,' share similar meaning. That is, 'smart' and 'stupid' share semantic similarity (similar meaning) and syntactic similarity (they have similar role in English syntax). Words like 'smart' and 'stupid' will also occur in many similar contexts, so they are syntactically similar, though their semantic meanings are quite different. 'Smart' and 'stupid' are thus also similar to each other, yet they will not occur in contexts *as similar* as 'smart' and 'intelligent.' We would therefore conclude, quite rightly, that 'smart' is more similar to 'intelligent' than it is to 'stupid,' but all three are more similar to each other than a word that would rarely if ever occur in context with the others, like 'carpet' for instance.

We can derive many methods to exploit the distributional hypothesis and generate word embeddings that represent these contextual similarities. Bullinaria & Levy (2007), Bullinaria & Levy (2012), and Levy & Goldberg (2014) all discuss matrix factorization methods, like singular value decomposition, to compute word embeddings to capture semantic and syntactic meaning. In this chapter, I will discuss another approach, that underlying the `word2vec` model, which has achieved state-of-the-art results in many applied natural language processing tasks. The model is built on a neural network architecture, scales linearly with the size of the training corpus, and has efficient implementations in popular languages `R` and Python, which call highly optimized `C` routines.

I introduced deep learning feed forward neural networks in the previous chapter. In this section, I will build on that understanding to explain how distributed word representations can be estimated from a training corpus. The task is relatively straightforward: to reduce the dimensionality of words from their atomic representations to a distributed representation. That is, we want to go from representing words as one-hot vectors, which are sparse and have length equal to the size of the vocabulary, to representing words as dense, lower-dimensional real-valued vectors.

3.1.1 Continuous Bag of Words

Building Intuition To begin, assume we have a corpus that contains V unique words and N total words. For instance, if our corpus was the sentence 'The president is an executive, but the president is not a potato,' the document has $N = 12$ total words, and $V = 9$ unique words. We will index the sequence of words with $i = 1, 2, \dots N$ and unique words $v = 1, 2, \dots V$. The sequence of words in the corpus is $w_1, w_2, \dots w_i \dots w_N$. Each of the w_i can be represented as a V-dimensional one-hot encoding, which we will call x, implying that if $w_i = v$, then x_i is a vector such that $x_{i,v} = 1$ and $x_{i,v'} = 0$ for all $v' \neq v$. In other words, the corpus is a sequence of words, where each unique word is represented as it's own one-hot vector. For our example sentence, then, the words are corpus can be represented as:

To lighten some of the notational burden, we can consider the word-embedding procedure for a single word, say w_1, or the first word in the corpus. Given the distributional hypothesis, we expect that a single word can be best summarized by the context in which it appears. To

the	$w_1 = 1$	`[1,0,0,0,0,0,0,0,0]`	$= \vec{x}_1$
president	$w_2 = 2$	`[0,1,0,0,0,0,0,0,0]`	$= \vec{x}_2$
is	$w_3 = 3$	`[0,0,1,0,0,0,0,0,0]`	$= \vec{x}_3$
an	$w_4 = 4$	`[0,0,0,1,0,0,0,0,0]`	$= \vec{x}_4$
executive	$w_5 = 5$	`[0,0,0,0,1,0,0,0,0]`	$= \vec{x}_5$
but	$w_6 = 6$	`[0,0,0,0,0,1,0,0,0]`	$= \vec{x}_6$
the	$w_7 = 1$	`[1,0,0,0,0,0,0,0,0]`	$= \vec{x}_7$
president	$w_8 = 2$	`[0,1,0,0,0,0,0,0,0]`	$= \vec{x}_8$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
potato	$w_{12} = 9$	`[0,0,0,0,0,0,0,0,1]`	$= \vec{x}_{12}$

capture this context, we will attempt to predict the words near our current word, conditional on having observed the current word. For the moment, assume we use a one-word context. We would seek to estimate the probability $p(w_2 = \text{president}|w_1 = \text{the}), p(w_2 = \text{is}|w_1 = \text{the}), p(w_2 = \text{an}|w_1 = \text{the}), \dots p(w_2 = \text{potato}|w_1 = \text{the})$. More generally, we seek to estimate the probabilities $p(w_2 = v'|w_1 = v)$.

We can estimate these probabilities empirically, without any model. For all words in the vocabulary, the probability of following the word ‘the’ would simply be the proportion of times that each word follows ‘the’—i.e., the number of the times ‘president’ follows ‘the’ divided by the number of times ‘the’ occurs; the number of times ‘is’ follows ‘the’ divided by the number of times ‘the’ occurs, and so forth. In our example document,’ the resulting vector of probabilities would be `[0,1,0,0,0,0,0,0,0]` since the word ‘president’ follows the word ‘the’ both times that ‘the’ appears in our corpus. If we repeat that exercise for every word in the vocabulary, and stack the matrices horizontally, we would obtain a normalized *co-occurrence* matrix, where the v, v' entry is just the percentage of times that word v' immediately follows word v in the corpus. This could also be generalized to larger word contexts by, for example, computing the proportion of times v' occurs within a c-word window of v.

This does not achieve our goal of representing words in a lower-dimensional space, however. We could use matrix algebra to find some linear subspace of the co-occurrence matrix, e.g., using principal components analysis or singular value decomposition. But these methods scale poorly, and better results have been achieved using the following neural network set-up.

Neural Structure Again, for simplicity, let us continue dealing with a single word and single-word context. We wish to predict the next word in the corpus. Both the current word, 'the,' and the next word, 'president,' are represented as a V-dimensional one-hot encoding. We could fully connect the input vector to the output vector, but doing so would not provide an estimate of a lower-dimensional representation. In fact, this set-up would just reduce to the co-occurrence matrix formulation above.

To generate the lower-dimensional representation, we will impose a *bottleneck* in the network structure. The distributed embedding approach seeks to project words into a D dimensional space where $D \ll V$, i.e., we want some transformation such that $\vec{x}_i = v \in \Re^V \to \vec{m}_v \in \Re^D$. To achieve this, we choose D, and specify a one-hidden-layer artificial neural network with D hidden nodes. See Figure 3.1.

Some notation will make this clearer. The input to the matrix is a single word (the first word in the corpus), represented as a one-hot vector of length V, $\vec{x}_1$. Note that we are treating each element of the vector as an individual, scalar input. The hidden layer has D nodes, but these nodes too are scalars, so we will stack the nodes and treat them as a $D \times 1$ vector, $\vec{h}$. The target is the one-hot vector representing the next word in the corpus, so $\vec{y}_1 = \vec{x}_2$. The output of the model, then, is a $V \times 1$ vector of predicted values, $\vec{\hat{y}}_1$, where the v^{th} entry would represent the predicted probability that the second word in the corpus would be word v given that the first word is 'the.'

The network is fully connected. The weights connecting the input to the hidden layer, $\mathbf{Z}$, is a $V \times D$ matrix of real-valued weights. We will not perform any non-linear activation function at the hidden layer—it's unnecessary and would make computation more intensive. The hidden layer is thus computed as:

$$\vec{h} = \mathbf{Z}^{\mathrm{T}} \vec{x}_1 \tag{3.1}$$

The weights connecting the hidden layer to the output layer, **A**, is a $V \times D$ matrix of real-valued weights. Since the linear combination of real-valued h and real-valued **A** will not be constrained to the probability simplex, we will perform a softmax[1] transformation on the raw outputs. This will ensure that the outputs represent a valid probability distribution over the words. The model output is computed as:

$$\vec{\hat{y}}_i = \text{softmax}(\mathbf{A}^{\mathrm{T}}\vec{h}) \tag{3.2}$$

It is worth considering for a moment what the matrix multiplications in Equations 3.1 and 3.2 are doing. In the first case, note that $\vec{x}_1$ is a one-hot vector representation of the first word. By pre-multiplying the one-hot vector by $\mathbf{Z}^{\mathrm{T}}$, we are simply extracting the first column of $\mathbf{Z}^{\mathrm{T}}$, or the first row of **Z**. Thus the hidden node $h_1 = z_{1,1}, h_2 = z_{1,2} \ldots h_D = z_{1,D}$. More generally, if the vector $\vec{x}_i$ represented word v, then the hidden node is $h_d = z_{v,d}$.

Moving to the next stage of the network, we are essentially trying to use the hidden layer to predict the target. The task here looks very much like a multinomial logistic regression model. The categorical outcomes are the V words that could potentially be the next word in the corpus. As in a multinomial logit, predictors $(h_1, h_2 \ldots h_D)$ receive a separate set of weights for every outcome. If we knew $\vec{h}$ *a priori*, then the inference task would reduce to a multinomial logistic regression, i.e., to optimize the $V \times D$ weights that give the best mapping from $\vec{h}$ to the observed outcome, $\vec{y}_1$.

Again, it is worth pausing to consider what that means: we are using the $D-$dimensional representation of the word to predict the next word in the corpus. If this prediction exercise works well, it will mean that we have found a lower-dimensional representation of the word w_1 that is nevertheless still contains useful information for predicting the following word. Put another way, we will have compressed the information represented by the one-hot vector into

---

[1] The softmax function, defined $\text{softmax}(u_1) = \exp(u_i)/\sum_i \exp(u_i)$, is the same as used in multinomial logistic regression. It can take positive or negative inputs u_i, make them positive through exponentiation, and normalize them so that they sum to 1.

a much smaller space. Unlike the one-hot embedding, though, the D-dimensional embedding will be real-valued, meaning that we can compare words to each other in a continuous space.

To estimate the weights in the network, our natural inclination would be to use backpropagation. We can define the loss initially using cross-entropy, $\ell(\vec{y_1}, \vec{\hat{y}_1}) = -\sum_{v=1}^{V} y_{1,v} \log(\hat{y}_{1,v})$. Conveniently, since the target y_1 is one-hot encoded, the only cross-entropy loss derives from the predicted probability corresponding to the actual target value. (In other words, if the target word is ‘president’, only the entry in the one-hot vector $\vec{y}_1$ corresponding to ‘president’ will contribute to the loss, since every other predicted probability is being multiplied by zero.) As a corollary, we can view the task as maximizing the $\hat{y}_{1,v*}$, where $v*$ is the index corresponding to the actual target word. And for ease, to simplify the softmax function, we can maximize the *log* probability. Taking the log of the $\hat{y}_{1,v*}$ yields $\log(\exp(\mathbf{A}^{\mathrm{T}}_{v*,\cdot}\vec{h})) - \log(\sum_{v=1}^{V} \exp(\mathbf{A}^{\mathrm{T}}_{v,\cdot}\vec{h})) = \mathbf{A}^{\mathrm{T}}_{v*,\cdot}\vec{h} - \log(\sum_{v=1}^{V} \exp(\mathbf{A}^{\mathrm{T}}_{v,\cdot}\vec{h})$. This is our objective function that we wish to maximize, or alternately, we can try to minimize the negative objective. Thus our new loss to minimize is $\ell = -(\mathbf{A}^{\mathrm{T}}_{v*,\cdot}\vec{h} - \log(\sum_{v=1}^{V} \exp(\mathbf{A}^{\mathrm{T}}_{v,\cdot}\vec{h}))$[2].

The backpropagation updates for the weights in the **A** are:

$$\Delta a_{d,v} \propto -(\hat{y}_{1,v} - y_{1,v}) h_d \tag{3.3}$$

$$\Delta \vec{a}_{\cdot,v} \propto -(\hat{y}_{1,v} - y_{1,v}) \cdot \vec{h} \tag{3.4}$$

and the backpropagation updates for the weights in **Z** are:

$$\Delta z_{v,d} \propto \ell h_d \tag{3.5}$$

$$\Delta \mathbf{Z}_{v,\cdot} \propto \ell \cdot \vec{h} \tag{3.6}$$

---

[2] This should intuitively make sense: when $\hat{y}_{1,v*}$ is very near 1, it will mean that the model has generated a good prediction. For $\hat{y}_{1,v*}$ to be near 1, it must be the case that $\mathbf{A}^{\mathrm{T}}_{v*,\cdot}\vec{h}$ is much larger than all of the other $\mathbf{A}^{\mathrm{T}}_{v,\cdot}\vec{h}$, $v* \neq v$, so that the loss would be very small. If $\hat{y}_{1,v*}$ is not near 1, the model is incorrect, and the loss will be proportional to how far off $\hat{y}_{1,v*}$ is from 1.

Generalization 1: Context Size Training a neural network model to predict trailing words gives us what is essentially a 'bigram' model, or one that will estimate the word vectors to produce valid two-word phrases. Words near each other in the resulting D-dimensional embedding will be words that tend to appear together in two-word phrases. This can begin to capture semantic and syntactic meaning, but a wider context tends to work better. For example, by modeling the probability that a word w_i appears, given the preceding words $(w_{i-3}, w_{i-2}, w_{i-1})$ and trailing words $(w_{i+1}, w_{i+2}, w_{i+3})$, we can learn embeddings that capture the meaning of a word based on its context.

Fortunately, using a multi-word context does not overly complicate our approach. Let C be the *context window size*, such that will consider the C words around w_i to predict w_1. A two-word context corresponds to using the two preceding and two trailing words. In many applications, a larger window size will perform better. In Figure 3.2, I show the schematic representation of a $C = 2$ word context around the target word.

We maintain the representation of the network in large part. As inputs now we have the C concatenated word vectors representing the one-hot encoded words, C preceding and C following the current word under consideration. The input now has dimension $V \times 2C$, the weights (still) have dimension $V \times D$, so the linear transformation in Equation 3.1 yields a $D \times C$ matrix. To combine these to maintain the $D \times 1$ shape of the hidden layer, we simply take the average over the rows. Equivalently, we can treat the input vector as taking the average over the $2C$ one-hot input vectors: $\vec{\hat{x}}_i = \frac{1}{2C}(x_{i-C}, \dots x_{i-1}, x_{i+1}, \dots x_{i+C})$, and then leaving the hidden layer unaveraged. This is what is shown in Figure 3.2, where the four context words are combined to a single input, which is then connected to the hidden layer. The network from this point onward is the same as before.

Generalization 2: Multiple Words Clearly we do not wish to fit the model to a single context to predict a single word. Intuitively, we can imagine fitting the model by iterating over the words in the corpus, though that would be quite inefficient for large corpora. (A modest corpus for training word embeddings can be in the millions of words; many example models have been fit to billions of words.) If we transition from the 'bigram' model to a larger context, we would need to consider every unique *context* for words. While some multi-word contexts will occur frequently—like the four-word window around 'the' in the expression 'president of the united states'—we will still end up with a very large number of unique multi-word contexts to sum over. In short, while gradient descent methods to optimize the parameters in the neural network model is theoretically feasible, such a model could prove quite computationally expensive. To avoid such computational complexity, Mikolov et al. (2013) propose a *negative sampling* algorithm.

The basic idea behind negative sampling is fairly straightforward: instead of considering all output words, we can consider only a portion of them (Rong 2014, Goldberg & Levy 2014). Clearly for every context in the corpus, we need to maintain the actual target word $v*$ in the sample. We then randomly select words that do *not* belong in the context, hence *negative* samples. Mikolov et al. (2013) describes an empirical distribution for this random sample, which is basically proportional to the unigram distribution in the text—i.e., we randomly select negative samples proportional to the frequency with which words appear in the corpus. On each iteration, we sample every true outcome word, and thus update the vectors associated with these words; and we randomly sample some words that do not belong in the context, and update those vectors too. This still yields a sufficient number of updates during training to reach stable and meaningful embedded word vectors.

3.1.2 Word Embeddings

Once the optimization routine converges, we obtain two parameter matrices: the $V \times D$ matrix $\mathbf{Z}$, and the $D \times V$ matrix $\mathbf{A}$. The former projects the one-hot or context-averaged input vector to a $D-$space, and the former projects the $D-$dimensional hidden vector back to $V-$space. In essence, then, we have used the neural architecture to factorize a version of the contextual co-occurrence matrix (Levy & Goldberg 2014). As Levy & Goldberg (2014) point out, an adapted singular value decomposition method can perform roughly as well as the neural network model on certain tasks, though the neural network model seems to perform better on average.

As stated above, the D-length hidden vector represents the input word (or several context words) in a lower dimensional space that nevertheless maintains the ability of the context to predict valid word sequences. The input-to-hidden layer matrix $\mathbf{Z}$ thus gives us the projection of the context words into their low-dimensional representation. The vector $\mathbf{Z}_{v,\cdot}$ is the D-dimensional embedding for word v.

To see how these word embeddings capture semantic and syntactic similarity, I fit the continuous bag of words model to the Ideological Books Corpus, a collection of more than 200 books and magazine articles written by political figures and commentators. The corpus has just over 20 million words, and roughly 51,000 unique words, after lower-casing and removing numbers. For the following examples, I specify the model with D =300 dimensional embeddings, though clearly this parameter will need to be tuned in research applications.

In this section, I present the results of the model, showing how to evaluate the syntactic and semantic similarity between words in the corpus. I then present a method for measuring the performance of the model, and compare model performance across several specifications.

Word Similarity Unlike the atomic (one-hot) representation for each word in the vocabulary, the distributed word representations exist in a continuous space. As such, we can conceive of the distances between words, with the expectation that words close to each other will share semantic and/or syntactic similarities. Two simple measures for distance in this space are Euclidean distance and the cosine similarity. The Euclidean distance between two words, say v, v', is the shortest line between the two words in the $D-$dimensional space: $\sqrt{\sum_{d=1}^{D}(z_{v,d} - z_{v',d})^2}$.

The cosine similarity measures the cosine of the angle (in radians) between two vectors[3]. A cosine similarity of 1 implies that the words are exactly the same, and higher scores imply that two words are close to each other in the $D-$space. The cosine similarity is computed

$$\frac{\sum_{d=1}^{D} z_{v,d} z_{v',d}}{\sqrt{\sum_{d=1}^{D} z_{v,d}^2}\sqrt{\sum_{d=1}^{D} z_{v',d}^2}} \tag{3.7}$$

Table 3.1 shows six words that appear in the corpus, along with the five words that the CBOW model finds to be most similar by cosine similarity. The results show how the lower-dimensional representation of words captures semantic and syntactic similarity. On the former count, we see that many of the words in the table are similar in meaning to their reference word: e.g., (awful, horrible), (terrific, fantastic), (political, ideological). In some cases, the terms are not synonyms but are syntactically similar. Though (good, bad) and (liberal, conservative) are opposites, they occur in similar contexts in natural language.

We can also consider the similarity of multiple vectors simultaneously. Consider three cases. The words most similar to 'conservative' are: liberal, conservatism, libertarian, conservatives, populist. The words most similar to 'religious' are: religion, christian, christianity, judeo, religiously. These are each computed by comparing the $D-$length vectors for the word 'conservative' and 'religious' and returning the vectors with the highest cosine similarity to each individually. If we combine the two vectors `conservative + religious` by averaging them together, we get another D-length vector. The words in the corpus closest to this joint vector are: evalgelicals, christian, traditional, tradition, fundamentalist. In other words, we are finding words that are close to *both* 'religious' and 'conservative.' This returns words not simply related to conservatism and religiosity, but specifically related to the religious right in the United States.

---

[3] If the angle between two words is 0 degrees (0 radians), it would imply that the words have the exact same location in the space, and thus have a similarity $\cos(0) = 1$. If the words were completely the opposite of each other in the space, they would have an angle of 180 degrees (π radians), and thus have a cosine similarity of $\cos(\pi) = -1$. If the words were completely unrelated, they would be orthogonal to each other in the space, meaning 90 degrees ($\pi/2$ radians), and thus have cosine similarity $\cos(\pi/2) = 0$. In other words, words very close to each other will have a cosine similarity near 1, words unrelated to each other will have cosine similarity near zero, and words opposite each other will have cosine similarities near -1.

Word similarity also reveals how domain-specificity might matter when applying these learned word vectors to research tasks. When I train the CBOW model on the IBC, the words estimated most similar to 'gridlock' are: partisanship, impass, polarization, partisan, polarized. For comparison, if we consider the pre-trained word vectors released by Google, which were trained on the roughly 100 billion word Google News corpus, the most similar unigrams to gridlock are: traffic, jams, snarls, congestion, stalemate. If we sought to use distributed word representations for a political research task, the IBC-trained model's 'understanding' of gridlock would match our intuition better than the more general Google News-trained model's understanding. We will consider this question again in the next section.

Evaluation Word embedding models are unsupervised. They seek to learn the underlying structure of language based on some version of word co-occurrence. It is thus difficult to evaluate model performance. We could expose the model to a test set of context-target pairs, like those used to train the model. But the size of the vocabulary—again, tens of thousands to million of words—means that to get a reasonable test set, we would need a massive testing corpus[4] (in addition to a large training corpus).

Bruni, Tran & Baroni (2014) propose an evaluation method using a hand-coded word similarity set. The authors used Amazon Mechanical Turk to obtain human coding for a set of 3000 word pairs, which workers rated on a seven-point Likert scale based on how similar the words were. The authors note that ratings were reasonably reliable, with a correlation of 0.84 between the average score of two experts with the scores by the Turkers. The MEN dataset (named for Marco, Elia and Nam, the study's authors) gives us a manner to integrate human evaluation with the model results, and thus compare various unsupervised model specifications based on how the perform on a supervised task. Unfortunately, the MEN data is generic—pairs include

---

[4] Why? Consider what such pairs would demand of the model: given a context, predict the *single* word that belongs in context in one sample. In the context 'The president has ___ the tax bill,' many words would make sense: debated, signed, vetoed, rejected, discussed, and so forth. If the single sample of this context actually says that the president 'considered' the tax bill, all of the former predictions would be incorrect, despite them being reasonable. We would need many testing samples to get a stable estimate of model performance.

(car, automobile) and (sunlight, sun)—and contains very few pairs specific to political topics. I thus composed a separate evaluation set that is domain specific. Following Bruni, Tran & Baroni (2014), I compose random pairs of words from the IBC, with selection proportional to word frequency in the corpus. I omit stopwords, however, since these convey little information, and manually trim common but book-specific terms, like the word 'chapter.' As a sample, pairs include (budget, fiscal) with similarity 6, (congress, court) with similarity 5, and (tomorrow, legislation) with similarity 1. I rated the pairs, along with a fellow researcher. The correlation between the raters was 0.72.

I compare several model specifications, based on how they perform on the MEN and domain-specific evaluation sets. As a sort of external baseline, I compute performance statistics using the Google News vectors, provided by Google, which contains three million unique words and 300-dimensional embeddings. For the CBOW model fit to the IBC, I vary both the number of dimensions in the embedding, and the context window-size. With limitless data, more embeddings should be capable of capturing more nuances of similarity, and typical window sizes are 5-15 words. Very small context windows tend to miss contextual information, while context windows too large introduce too much noise by, for example, trying to uncover semantic or syntactic similarity between two words 100 spaces apart.

Figures 3.3 and 3.4 show the model evaluation results. The x-axis shows the dimensionality of the word embedding, and the y-axis shows the correlation between the human-coded similarity between words and the model cosine similarity between words. Point shapes indicate the context window size (i.e., the number of words to the left and right of a target word used to predict it), and I present the results from four sizes: 5, 8, 11 and 15. The dashed line shows the result from the Google News corpus with context size of 10 and 300-dimensional embeddings. First, the Google News word embeddings yield better results than any of the IBC embeddings on the MEN test set, as evidenced by the wide gap between the various IBC-trained model results and the dashed line at 0.64, the correlation between the Google News-trained embedding similarities and those in the MEN evaluation data. This makes sense: the Google News corpus contains about 100 billion words in the corpus, meaning that the model is able to observe an incredible number of contexts, and thus can extract high-quality information about the semantic and syntactic meaning of the words. The IBC, by contrast, offers fewer data with which to train the model, and performs understandably worse on the generic test set.

The IBC-trained model performs fairly well on the domain-specific test, however. In some cases, the IBC-trained model performs nearly on-par with the Google News-trained model. The IBC-trained model seems to reach peak performance on both tasks with a smaller embedding size, with a peak at somewhere around 200-250 dimensional embeddings, which again makes sense given the smaller size of the IBC. The size of the context window has a rather smaller effect on performance. Both the five- and fifteen- word context perform well, while a seven-word context performs somewhat worse. Aside from very small and very large contexts performing poorly, there seems not to be much of a clear pattern between window-size and performance on either test set.

3.2 Document Representations

To this point, I have presented the details of neural network distributed word representations. The neural network models scale words into a lower-dimensional representation, which turn out to capture latent semantic and syntactic structure in language. Yet to what end? In many applied tasks, researchers have found that word embeddings augment performance on various tasks in natural language processing, such as sentiment analysis (Tang, Wei, Yang, Zhou, Liu & Qin 2014, Liu 2012, Dos Santos & Gatti 2014, Xue, Fu & Shaobin 2014), document classification (Kim 2014, Zhang, Zhao & LeCun 2015), and topic classification (Cao, Li, Liu, Li & Ji 2015). In the literature, however, there are many ideas about *how* to leverage the word embeddings. Imagine we seek to classify the ideological perspective of documents using a supervised method. We have training documents, labeled by their ideological perspective; and we have word embedding vectors for the words in the corpus. How do we aggregate the word-level embeddings into a document-level representation to use in the classification task?

3.2.1 Vector Aggregation

In this section, I consider two simple methods for aggregating the word-level embeddings to the document level. First, I simply average the word vectors together. This method employs a similar logic as the `conservative + religious` example above: by averaging together the word embedding vectors for each word in a document, we can position the document in the same space as the words themselves, which might convey information relevant to the target classification, e.g., the ideological perspective of a document. I consider three weighting schemes: uniform-weighting, frequency-weighting and tf-idf weighting. Uniform weighting simply averages together the embeddings for words that appear in a document, regardless of how frequently each word occurs. Frequency-weighting takes a weighted average of the word vectors, with the weight proportional to the number of time each word occurs in the document. Tf-idf weighting computes the weighted average, with weights given by the term frequency-inverse document frequency for each word in each document (see the introduction to this thesis for more).

Second, I employ a max-pooling method, which combines the embeddings for a document by taking the maximum value in each of the D dimensions across all of the words in the document. In other words, if we collect all word vectors for each word in a document, the first element in the resulting max-pooled vector will be the maximum value in the first dimension of all words in the document. This approach has proven useful in some deep learning approaches (Kim 2014, Hu, Lu, Li & Chen 2014, Zhang & LeCun 2015), though in this context it is certainly atheoretical and might prove ineffective.

In general, should we expect these methods to work? Perhaps not. There are no real interpretations of the latent dimensions into which we are projecting the words using the CBOW model. Thus there is no reason to expect that averaging the projections together, or taking maxima over the projections, would be themselves meaningful. Then again, models that ignore word order do not seem like they should work, since order is clearly important for language; yet they have proven useful enough to dominate text analysis to this point. Researchers, such as Iyyer et al. (2014), have indeed found that averaging word vectors can work. While methods that account for word order and linguistic structure improve model performance at predicting how liberal political language is, they find that baseline models using simple averages do not perform too poorly. If simple methods can produce decent results, we could reduce some of the costs to applied researchers of using distributed word representations in their scholarship.

3.2.2 Convolutional Networks

While appealingly simple, the above methods are also rather coarse. Distributed word representations clearly capture important latent structure in language, but it's also not entirely clear *why* these methods work (Levy & Goldberg 2014). As such, it's not obvious that averaging or max-pooling the word vectors would result in a meaningful representation of documents, and thus might not work well in text classification or prediction tasks. The simple aggregation methods also discard important information conveyed through word order. This is true in bag-of-words models, too, as discussed in the introduction to this thesis.

To avoid the problems that arise from both simplifications, I introduce an adaptation of *convolutional neural networks*. CNNs are deep learning models that filter inputs in local regions, and aggregate signals up to higher levels of abstraction (Le Cun, Denker, Henderson, Howard, Hubbard & Jackel 1990, Dos Santos & Gatti 2014, Severyn & Moschitti 2015). Their name derives from the concept of 'convolution' in mathematics, where we repeatedly apply a function to the output of another function, which corresponds to the CNN's application of filters to the output of other filters.

The first applications of convolutional layers in artificial neural networks emerged from image processing and classification research. Convolutions make a bit more intuitive sense in these cases, so we can begin with an image as an example. A traditional feed-forward ANN, like those discussed in Chapter 2, would attempt to classify an image based on the activation of pixels in an image. For instance, we could attempt to recognize a black-and-white hand-written character based on how dark each pixel in the image is. In a simple ANN, as with a regression model, this approach ignores spatial relationships between pixels. The convolutional network samples regions of the space, and then aggregates the information learned from those regions to use in prediction tasks. This can both improve prediction accuracy, and allows for recognition of, say, hand-written digits regardless of where they occur in a larger image.

When applied to text models, CNNs sample regions of the words in a document, and then aggregate the information extracted from those regions into a higher-level abstraction. Essentially, this provides a natural way to account for local dependence between words without having to $n-$gram the text beforehand. For example, very different messages are conveyed by the phrases 'Libertarian ideals are fundamental' and 'Libertarian ideals are dangerous.' In a bag-of-words approach, we would need to specify each as a trigram, `libertarian_ideals_fundamental`, `libertarian_ideals_dangerous`, which for this example works fine, but when applied to a large corpus will result in a huge vocabulary of mostly nonsensical n-grams. CNNs, by contrast, can account for the proximity of 'libertarian ideals' and 'fundamental' versus 'dangerous' naturally and quite efficiently (Zhang & Wallace 2015, Zhang, Roller & Wallace 2016).

Document representation here differs from that used in a straightforward ANN. Instead of representing documents via a document-term matrix, we represent *each document* as an $N_j \times D$ matrix, where $j = 1, 2, ...J$ indexes the documents in the corpus, and $i = 1, 2, ...N$ indexes the words in the document. So if the first document has 100 total words (i.e., not unique words), and we represent the words using $D = 300$ dimensional word embeddings, the document will be represented by a 100×300 matrix. The first row of the matrix would represent the embedding for the first word in the document, and so forth. In practice, we standardize document length so that all documents have the same matrix dimensions. Documents longer than the standard length are truncated, while shorter documents are padded with a filler term[5].. The CNN has three stages that we can consider in turn: filtering, pooling and feeding forward.

Filtering In image processing, where CNNs first gained traction, filters represent a transformation applied to an image. Filters take areas of, e.g., pixels and locally manipulate them. Averaging pixels with their neighbors blurs an image; differencing pixels with their neighbors tends to invert an image and produce bright outlines and dark interiors of shapes. Within the CNN framework, we can think of filters as a window of weights that pass over the input data—the document—and transform it via Hadamard (i.e., element-wise) multiplication and summation. For example, if we take the following matrix $\mathbf{X}$ and apply the following filter $\mathbf{F}$, the resulting filtered representation of $\mathbf{X}$ would be $\mathbf{X} \circ \mathbf{F}$:

$$\mathbf{X} = \begin{bmatrix} 0 & 0.1 & 0 \\ 0.5 & 0 & 0.5 \\ 0.1 & 0.5 & 0.1 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad (\mathbf{X} \circ \mathbf{F}) = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0.1 \end{bmatrix}$$

For images, the filter window has a height and a width, to be determined by the researcher. The window will begin in the top-left of the image and slide across, moving by a user-specified stride (typically one), until it reaches the right edge, after which it returns to the left edge and shifts down one pixel. At each stop, the filter is applied, and the resulting filtered representation is compressed (summed) into a *feature map*.

In text modeling, the window will have a fixed width, D, so that the filter is applied to all dimensions of the word embedding vectors. The filter is applied and the result compressed into a feature map, and then filter then slides down again. A document with $N = 10$ words and a filter window size of $K = 5$ will produce a feature map with $N - K + 1 = 6$ elements. As a simple example, assume we have the $\mathbf{X}$ matrix from above, and a filter $\mathbf{F}$ of size 1, i.e. the filter acts on one row of $\mathbf{X}$ at a time. The resulting feature map would have 3 elements, computed as such:

---

[5] For more information, see the introduction to this thesis

$$\mathbf{X} = \begin{bmatrix} 0 & 0.1 & 0 \\ 0.5 & 0 & 0.5 \\ 0.1 & 0.5 & 0.1 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \quad (\mathbf{X} \circ \mathbf{F}) = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0.1 \end{bmatrix},$$

$$\text{feature map} = \begin{bmatrix} 0 \\ 1 \\ 0.1 \end{bmatrix}$$

If instead we have a filter size 2, the resulting feature map would have 2 elements: one from the two-row filter applied to the first two rows of **X**, and one from the filter applied to the second and third rows of **X**. Each of these is summed up, which generates the two elements of the feature map:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad (\mathbf{X} \circ \mathbf{F})_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \end{bmatrix}, \quad (\mathbf{X} \circ \mathbf{F})_2 = \begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & 0.1 \end{bmatrix}$$

$$\text{feature map} = \begin{bmatrix} 1 \\ 1.1 \end{bmatrix}$$

In the examples above, we have applied a single filter of deterministic weights to a document, which yields a single feature map for a single document. This process differs in practice in three ways. First, the weights in the filter are estimated rather than fixed *a priori.* Second, we typically specify many (e.g., tens to hundreds of) filters, each of which is optimized during training. If we are performing document classification, this enables us to estimate or ‘learn’ the best filters to use for the current task. Third, we can specify filter windows of variable size, for example specifying a set of filters that scans over two-word windows, another set scanning over three-word windows, and so forth. Each filter produces a separate feature map for each document.

Visuals can better convey the process. In Figure 3.5, we can consider how a single filter behaves on a document. Again, let us consider the document 'The president is an executive, but the president is not a potato.' The document is represented by an $N = 12$ word by $D = 6$ dimensional embedding matrix. (Clearly in practice, embeddings will have more than six dimensions.) For the example, imagine that the colors are unique to the words, and that the color saturation reflects the magnitude of the embedding for a particular word.

We apply a three-word filter window, meaning we will apply our filter (the red rectangle) to words (1,2,3), words (2,3,4), and so on. By moving the filter down one word each time, we will obtain $12 - 3 + 1 = 10$ filter applications. For each filter application, we compress the resulting matrix by multiplying the filter weights by the each three-word window window, and by summing the product, thus creating a scalar representation of the filter. For each filter application, we concatenate the resulting compressions into a 10-length feature map.

Extending this logic, we turn to Figure 3.6. This shows The process for using both two- and three-word filter windows. The three-word filter window produces a 10-length feature map, as before; the 2-word filter window produces an 11-length feature map. Performing max pooling on each produces a scalar representation of both feature maps, which are then put together into the final concatenated feature map. This process allows us to combine representations built from filter windows of different sizes.

Pooling The result of filtering produces many feature maps, potentially even producing sets of feature maps with different lengths. Figure 3.6 shows such a case, where the three-word filter produces a 10-length feature map, while the two-word filter produces an 11-length feature map. To combine the feature maps into a single document representation, we perform pooling. The most common form of pooling is *max pooling*, which selects the maximum value in each feature map, creating a single representation of the document with as many elements as there are feature maps. (Nagi, Ducatelle, Di Caro, Cireşan, Meier, Giusti, Nagi, Schmidhuber & Gambardella 2011, Zeiler & Fergus 2014, Giusti, Cireşan, Masci, Gambardella & Schmidhuber

2013). Alternatives include mean pooling or fractional max pooling (Graham 2014, Boureau, Ponce & LeCun 2010, Lin, Chen & Yan 2013). In this chapter, I will exclusively use max pooling.

Feed Forward After pooling, we have a representation of each document: a vector with length equal to the number of filters. Each filter applied to the text contributes one element of that pooled layer. From this point on, we can fit a traditional feed-forward network, along with the concomitant decisions discussed in the first chapter of this thesis: numbers of hidden layers, numbers of nodes, activation functions, and so forth. The output of the model is the target, e.g., the document classifications, and we optimize parameters (the filters and the weights in the feed-forward portion of the network) to get the highest predictive performance. We can fit the parameters using stochastic gradient descent, as with simple ANNs.

Multi-Channels An extension of the framework above is the use of multiple 'channels' in the network. In the models discussed to this point, each document has a single input representation: each row is a word, in the order in which they appear in the document; and the row vectors are the word embeddings from, e.g., a continuous bag of words model as described in Section 2. We can extend this architecture to allow for multiple distributed word representations. For instance, we could fit a CBOW model to our training corpus and use the resulting embeddings as one channel in a CNN, and use embeddings from a massive (but perhaps not domain-specific) corpus, like those trained on the Google News corpus, as another channel.

3.3 Applications

The purpose of this paper is to consider to what extent, and under what modeling specifications, distributed word representations can aid researchers at measurement tasks. To this end, I assess the predictive accuracy of various models using word embeddings. For one set of models, I aggregate word embeddings using the two methods described above: averaging together the embeddings for all words in each document, and pooling the embeddings for all words in each document by choosing the largest value on each dimension. I also fit several convolutional neural network models, which considers documents as represented by a sequence of words and their associated embeddings.

I deploy these approaches toward two classification tasks. First, I use the data provided by ReactLabs to predict the tone of utterances by Mitt Romney and Barack Obama (as in Chapter 2). Second, I push the methods a bit further, toward serving as a supervised topic model. Using the hand-labeled topic codes provided by ReactLabs, I train models to correctly topic-classify statements by the two candidates. A third application, analyzing the structure of elite and electoral political ideology appears in the following chapter.

3.3.1 Debates

Following the task described in Chapter 2, I return to the task of measuring the tone, or positivity/negativity of expressions by Democrat Barack Obama and his Republican opponent Mitt Romney in the October 3, 2012 presidential debate. In this chapter, I attempt to leverage the information provided in distributed word representations to improve predictive accuracy. Each utterance (sentence or stand-alone clause) comes with a hand-coded tone classification.

I extend the application by also considering a another problem: topic coding. Each utterance in the data has been topic coded by area experts (Boydstun et al. 2014). The topic codes replicate those used in the Policy Agendas Codebook[6]. Table 3.2 shows the policy area codes and the proportion of utterances falling into each category. Because certain classes contain too few utterances for reliable training and test-set generation, I consider five topics: macroeconomics, health, labor/jobs, education, and defense.

The topic task is potentially easier insofar as topics *can* be quite different in terms of the language used when discussing them. There are very different sets of words and phrases used in discussing Social Security and immigration, for instance. Classifying topics can be more difficult in other ways, however, since it requires classification into one of multiple classes. In

---

[6] The coding scheme was originally devised by Frank R. Baumgartner and Bryan D. Jones, with the support of National Science Foundation grant numbers SBR 9320922 and 0111611, and are distributed through the Department of Government at the University of Texas at Austin. Neither NSF nor the original collectors of the data bear any responsibility for the analysis reported here.

this example, we have the added difficulty of topics that conceptually overlap (e.g., macroeconomics and jobs are similar topics). In a bag-of-words approach, the best performing classification model considered reached roughly 60 percent accuracy. From one perspective, this result is disappointing, since it means that of all new utterances considered, a bag-of-words model might only get a bit over half of the topic classifications to match expert assessments. On the other hand, if we uniformly assign new utterances to the modal topic class, we would only expect to get about 40 percent correct. A bag-of-words model, such as those presented in Chapter 2, can beat these baselines considerably.

Unfortunately, as Table 3.3 shows, aggregating the word embedding vectors does not produce reliable results on either the tone or topic classification task. The table shows the predictive accuracy of various models, using three different averaging methods on the word embeddings, as well as max-pooling, to combine word embeddings into a single document representation When attempting to map language onto tone, vector averaging only yields 51 to 55 percent accuracy, which is barely above what random assignment could obtain. Max pooling fails similarly, with the best performing ANN model (which has two layers of 50 hidden nodes) only achieving 59 percent accuracy, while the other baseline models perform slightly worse. This compares quite poorly to the results shown in Chapter 2, where bag-of-words models routinely achieved north of 65 percent accuracy. On the topic classification task, we again find both vector aggregation methods producing poor results. Both vector averaging and max pooling produce classification accuracy between 25 and 40 percent. Simply choosing the modal topic category would give accuracy of 40 percent. In short: this has not worked.

The failure of the vector aggregation models is disappointing, but perhaps should not be unexpected. As aforementioned, averaging or max pooling the word representations may make sense for short phrases, such as averaging the vectors ‘religious’ and ‘conservative’ in Section 2. For longer documents, though, it is not entirely clear why aggregating the vectors would be useful for classification. Is a sentence—much less a longer document—really semantically equivalent to the sum, or average, of its component words? Some researchers have found that averaging may help to partition documents into very different topics—e.g., food, politics, geography, and physics—but is far too coarse for finer-grained distinctions on domain-specific tasks (Clinchant & Perronnin 2013).

On the other hand, the covolutional neural network models seem to leverage the word embeddings to good effect. The test data accuracy results from various specifications appear in Table 3.4. The model names give the filter window sizes for each CNN; i.e., $\text{CNN}_{2,3,4}$ indicates a convolutional neural network model with filter window sizes of two, three and four. Each CNN was fit with 100 hidden nodes in a single hidden layer, and the activation function is soft ReLU. To minimize overfitting, I specify a dropout parameter $p = 0.5$. I fit each model using 50, 100 and 200 filters.

We can first focus on the left panel of the table, where we see the predictive accuracy on the tone classification task. On average, the varied CNN models perform well, though not uniformly. The $\text{CNN}_{(2)}$ model, which uses only one filter window size of two words, achieves only about 58 accuracy, or several points worse than the bag-of-words models discussed in Chapter 2. By expanding the number of filter window sizes, however, predictive accuracy jumps considerably. The $\text{CNN}_{(3,4,5)}$ model with 100 filters—which uses 100 filters over each of three, four and five word windows—achieves 82 percent accuracy on the test set. This not only performs better than any other model on this task, but also outperforms the best bag-of-words models considered in Chapter 2. The $\text{CNN}_{(3,5,7)}$ model, both with 50 and 100 filters, also performs reasonably well, beating the predictive accuracy of the most robust models in Chapter 2.

As with the vector aggregation models, performance drops when we switch to examining the topic classification task. The worst performing CNN, with only a two-word filter window, gives better but still modest results compared to the vector aggregation methods, with performance slightly less than 50 percent. The $\text{CNN}_{(2,3,4)}$ and $\text{CNN}_{(3,4,5)}$ models, both with with 100 filters, achieves a considerably better 61 percent accuracy.

Analyzing the heat map in Figure 3.7 shows the topic-specific prediction precision, as well as showing where the model makes the greatest prediction errors. I present the results from the $\text{CNN}_{(2,3,4)}$ model, but the results are similar for most CNN specifications. The model correctly predicts more than 60 percent of the utterances in the test set belonging to the health care, education and defense topics. The worst performance occurs with the macroeconomics and jobs/labor topics and, as the figure shows, this happens largely because the model tends to mis-classify each as the other. Take for an example Mitt Romney's statement "And over the last four years small-business people have decided that America may not be the place to open a new business." The model incorrectly classifies this statement as having an 84 percent chance of belonging to the 'macroeconomics' topic, when in fact the hand-coded label is labor/jobs. This type of error is disappointing, but reflects the logical conceptual overlap between the two topics.

Table 3.4 also provides an important warning regarding overfitting with CNNs. Based on the reported results, specifying somewhere around 100 filters seems to give optimal performance on these two tasks. Using only 50 filters tends to perform several points worse, on average, than using 100 filters. But we pay a price for using too many filters. Including more filters can improve performance on training data, but makes it easier to overfit the model to the training data. On both the tone and topic classification tasks, the predictive accuracy of the model on the test data drops—sometimes by as much as 23 points—when we double from 100 to 200 filters.

The takeaway from these applications can be summarized thus: (1) word embeddings convey useful information; (2) this information is not straightforward to use for other tasks, like sentiment analysis or topic classification; (3) accounting for word order creates a natural way for documents to be represented as collections of potentially correlated word representations; and (4) when correctly leveraged, word embeddings can improve model performance and result in more reliable measurements of concepts that we care about as applied researchers.

3.3.2 Note on Multi-Channels

To this point, I have represented words using embeddings provided by Google, trained on the Google News corpus. I chose these embeddings because they were trained on a large corpus and have been used in many other applied research projects. As I mention above, however, we can employ multiple word embeddings within a single model. In such a setup, every document in the corpus is represented as a matrix for each word embedding used. If we wished to train a CNN on both the distributed representations provided by Google, as well as those we trained on the Ideological Books Corpus, each document would then have two matrices: one with the Google News embeddings stacked in the order in which they appear in the document; and one with the IBC embeddings, stacked similarly. The model then proceeds as it would with a single channel: we filter and compress into feature maps, and then pool the feature maps from both channels into a single representation for each document.

The logic to this approach is compelling. For some words—particularly for common words—the Google News embeddings may have higher quality, since they were trained on an inordinate amount of data. For domain-specific language, however, word embeddings trained on political language may generate better representations. By filtering and pooling, a CNN can automatically select the best representation on a word-by-word basis.

To assess whether employing multiple distributed representations improves model performance, I consider the best performing CNN model presented in this chapter, the $\text{CNN}_{(3,4,5)}$, fit to two channels of embeddings: the Google News embeddings, and a 300-dimensional CBOW model fit to the Ideological Books Corpus. I cross-validate the model using 100 random partitions of the ReactLabs debate data: 600 utterances for training and 170 for testing for each partition. On each partition, I fit the $\text{CNN}_{(3,4,5)}$ model once using the Google News word embeddings, once using the IBC word embeddings, and once on the two-channel union of both.

The advantage of using multiple channels seems to depend on the application. Referring to Figure 3.8, using the IBC word embeddings alone performs worse than using those provided by

Google, and the empirical 95 percent interval obtained from the 100 cross-validation partitions are also markedly wider for the IBC-trained embeddings. This shows that the word embeddings trained on an exponentially larger corpus outperform those trained on the IBC, and also produce much more stable results. This makes sense, especially for a task like predicting tone. Though the utterances of the two presidential candidates are clearly about political topics, there is a certain universality to the language of optimism and positivity, and that of pessimism and negativity. Returning to an earlier example, consider the statement that "legislative functions ... are in a state of paralysis as a result of partisan gridlock." Whether the model places 'gridlock' in a similar space as polarization or traffic snarls probably matters little for predicting that the statement is negative. Embeddings trained on a much smaller corpus produces lower quality results when assessed on generic linguistic tasks (see again Figures 3.3 and 3.4), and this loss of information results in lower-quality predictions overall.

The story is somewhat different on the more domain-dependent task of classifying topics. Figure 3.9 repeats the cross-validation exercise using the topic classification task. It shows that we gain a slight—though truthfully not significant at any conventional level—improvement in average model performance by employing a two-channel CNN specification. As with the tone task, the GN embeddings produce better results than the IBC embeddings, and show more stability (lower variation across partitions) than using the IBC alone. Using both performs best, which indicates that combining general-purpose embeddings trained on a very large corpus, along with embeddings trained on a domain-specific corpus, can be worth it when the researcher is attempting a task where domain specificity is key.

3.4 Discussion

Taken together, these results suggest that distributed word embeddings can convey useful information about language, and that they can serve a purpose beyond parlor tricks like returning similar words or solving analogies. By capturing the semantic and syntactic features of words, distributed representations can account for linguistic patterns, which in turn can aid us in text classification or other modeling tasks. In bag-of-words models, we would typically estimate the effect of ‘health care reform,’ ‘affordable care act’ and ‘obamacare’ on the probability that a document belongs to a health care topic. Word embeddings can begin to account for the similarity between ‘obamacare’ and ‘health,’ and ‘obamacare’ and ‘reform,’ which makes our modeling more robust to the diversity of language.

Yet we also need to note that using distributed word representations in prediction tasks requires special care. Simple methods for aggregating word embeddings to represent documents largely fail. Averaging or max pooling embedding vectors together might work for constructing representations of two or three word phrases, like ‘religious conservative,’ but these methods are too coarse to give clean representations for longer documents. Bag-of-words models, like regularized logistic regression, support vector machines, or feed-forward artificial neural networks, fit to document-term matrices, perform better on average than models fit to aggregated vector representations of words.

When we represent documents as sequences of words, each of which is represented by a word embedding, performance improves. Convolutional neural network models give us one manner by which to accomplish this. CNNs were originally developed for image recognition, and achieved impressive results by constructing representations of whole images as the aggregate of smaller, regional images. Applied to texts, this means representing entire documents as the aggregation of smaller multi-word phrases. A single document can thus be represented as the composite of many smaller sub-documents, which are aggregated and used in prediction, classification, or scaling tasks.

The CNN models, combined with distributed representations, provide an additional advantage: relative simplicity. Unlike more traditional methods, the convolutional approach naturally learns multi-word representations that are useful in predictive tasks. In other words, we do not need to n-gram a corpus before proceeding to model specification. This point can be easy to understate, but it amounts to a considerable advantage in applied research. Processing a corpus into multi-word phrases can be both computationally expensive and can introduce enormous bias into the modeling process. Generating n-grams results in a very large vocabulary of unique words/multi-word phrases, but most of these phrases are meaningless and occur infrequently in the text. The phrase 'president is an' in the sentence 'The president is an executive' conveys, by itself, almost no meaning whatsoever. This requires us to filter phrases from the vocabulary, but these decisions can have large effects on the predictive abilities of the models we specify later. By naturally learning the temporal dependencies between words in a document, CNNs provide a compelling alternative.

The message on domain specificity is a bit more mixed. When we assess the performance of word embeddings on general tasks—like achieving a high correlation between common words that the models believe to be similar, and what human coders believe to be similar—embeddings trained on large, general-purpose corpora will tend to perform better. If we assess model performance on a domain-specific set of tasks, training corpora on domain-specific tasks can work better.

Similarly, if a text analytic project addresses a question that largely hinges on general linguistic differences—like the positivity or negativity of tweets or speeches, whether messages contain angry, violent, conciliatory or cooperative language, and so forth—using embeddings pre-trained on large corpora can improve performance on classification, prediction or scaling tasks. If the project hinges on leveraging differences in a domain-specific lexicon, where words can have very different implications than they do outside of the research domain, combining large-corpora embeddings with those the user trains on a domain-specific corpus may work better. As the results show, pre-trained embeddings produce good results on predicting the tone of utterances during a presidential debate, but are augmented on a finer-grained task like predicting the topics of such statements.

Future research could progress considerably, and along two different routes. (1) The next phase of research could explore better ways to incorporate distributed word representations into standard text analytic frameworks. Some researchers (e.g., Iyyer et al. (2014)) have found simpler models like logistic regression, combined with averaging word representation vectors, to perform reasonably well at classification tasks. Progress in this area would make it easier for researchers to incorporate continuous word representations into their research without necessarily needing to invest in building convolutional or other deep-learning models. (2) More research should be done on how to use convolutional neural networks, as well as recursive or recurrent neural networks, to use word ordering in an efficient way (Iyyer et al. 2014). This would potentially free us from relying so heavily on n-gramming texts, and thereby having to make the consequential and sometimes problematic choices thereby associated.

Using distributed representations of words has proven useful in many natural language processing tasks. To date, we have not brought these tools to bear in answering political science questions. Part of the reason, I suspect, is that it is not immediately apparent how to leverage these tools. Distributed representations are useful, but care must be taken to incorporate them with measurement tasks appropriately. This chapter attempts to convey the usefulness of distributed representations, to assess how they can best be integrated into political research, and ultimately to provide applied scholars with a useful tool for analyzing textual data. In the following chapter, I point these tools toward my own motivating substantive research program: analyzing the structure of elite political ideology.

3.5 Tables

Table 3.1: WORD EMBEDDINGS CAPTURE MEANING

awful	**good**	**terrific**	**blessed**	**political**	**judiciary**	**liberal**
horrible	great	fantastic	fortunate	politics	courts	conservative
terrible	bad	great	thankful	ideological	judicial	liberals
dreadful	decent	marvelous	bless	electoral	oversight	reformist
horrendous	nice	superb	wonderful	democratic	jurists	progressive
horrid	better	wonderful	grateful	partisan	justice	left

The table shows six unigrams (the column headers) and the five unigrams most similar to each, according to the cosine similarity metric. The results show both semantic (awful → horrible) and syntactic (good → bad) similarity. The results were computed by using results from the D_{300} continuous bag of words model.

Table 3.2: TOPIC DISTRIBUTION IN A 2012 PRESIDENTIAL DEBATE

Code	**Topic**	**Proportion**
1	Macroeconomics	0.29
2	Civil Liberties	< 0.01
3	Health	0.22
4	Agriculture	0
5	Labor/Jobs	0.05
6	Education	0.07
7	Environment	< 0.01
8	Energy	0.03
9	Sundry/Non-political	0.14
10	Transportation	< 0.01
12	Law/Crime	0
13	Social Welfare	0.03
14	Community/Housing	0.01
15	Banking/Finance	0.01
16	Defense	0.06
17	Science/Technology	< 0.01
18	Trade	< 0.01
19	Foreign Affairs	< 0.01
20	Gov't Operations	0.07

The distribution of topics, as coded by researchers with ReactLabs. Each utterance is coded into one topic category, as defined by the Policy Agendas Project. The 'sundry' topic, code 9, captures mostly statements that are not policy-oriented. In the context of the debate, this comprises mainly statements by candidates about the structure of the debate, or interactions with the moderator.

Table 3.3: TEST SET ACCURACY FOR TOPIC AND TONE

	Uniform Avg		Freq Avg		tf-idf Avg		Max Pool	
Model	**Tone**	**Topic**	**Tone**	**Topic**	**Tone**	**Topic**	**Tone**	**Topic**
Naïve Bayes	0.51	0.25	0.52	0.30	0.54	0.22	0.50	0.30
Logistic regression	0.52	0.33	0.52	0.22	0.48	0.23	0.53	0.28
Support vector machine	0.55	0.31	0.49	0.38	0.50	0.23	0.50	0.37
$\text{ANN}_{2\times 50}$	0.51	0.39	0.59	0.26	0.58	0.20	0.49	0.40

The testing accuracy for tone and topic, based on the hand-coded presidential debate transcript data from ReactLabs. The training set is a random sample of 600 documents used to train the models; the testing set is the remaining 170 documents not used for training. Column blocks give test set predictive accuracy from four models: Naïve Bayes, regularized logistic regression, support vector machine, and a feed forward neural network with 2 layers of 50 hidden nodes ($\text{ANN}_{2\times 50}$). Models fit after aggregating the word embedding vectors for each document. Uniform averaging simply averages the embeddings for all words in a document; frequency-weighted averaging (Freq Avg) is a weighted average based on how frequently each word appears in a document; tf-idf averaging (tf-idf Avg) is a weighted average based on the tf-idf score for each word in each document; and max pooling represents each document as the maximum value in each of the D dimensions for the words in the document.6 No model performs well. For tone, there are two categories which are roughly evenly distributed, and no model performs much above 50 percent. For topic, there are five categories, and the models perform worse than uniformly predicting the modal topic.

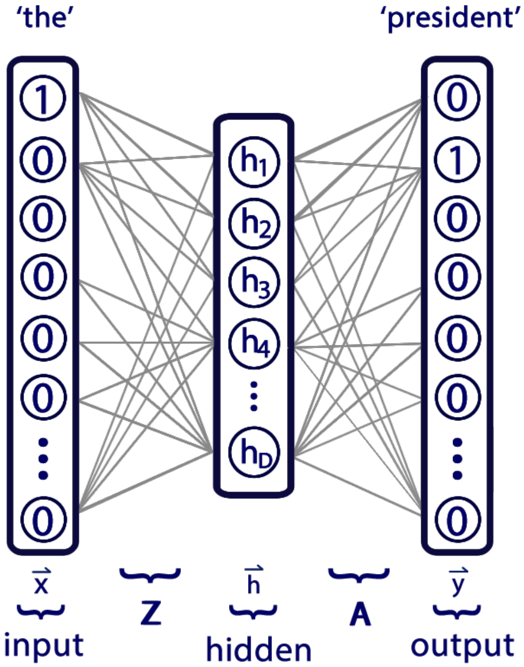
Table 3.4: CNN TEST SET ACCURACY FOR TONE AND TOPICS

	Tone			**Topic**		
Num. Filters:	50	100	200	50	100	200
$\text{CNN}_{(2)}$	0.58	0.57	0.58	0.46	0.48	0.43
$\text{CNN}_{(2,3)}$	0.62	0.63	0.59	0.49	0.52	0.54
$\text{CNN}_{(2,3,4)}$	0.72	0.75	0.60	0.59	**0.61**	0.57
$\text{CNN}_{(3,4,5)}$	0.74	**0.82**	0.59	0.53	**0.61**	0.58
$\text{CNN}_{(3,5,7)}$	0.68	0.79	0.61	0.49	0.49	0.51
$\text{CNN}_{(3,4,5,6,7)}$	0.72	0.77	0.58	0.53	0.55	0.53

Test set predictive accuracy for presidential debate tone and topics. The table shows testing accuracy (percent correctly predicted) for classifying tone (left) and topic (right), based on the hand-coded presidential debate transcript data from ReactLabs. The training set is a random sample of 600 documents used to train the models; the testing set is the remaining 170 documents not used for training. Bolded figures show the best overall performance on each task. The model names give the filter window sizes for each CNN; i.e., $\text{CNN}_{2,3,4}$ indicates a convolutional neural network model with filter window sizes of two, three and four. All CNN models presented use the same feed-forward specification: one hidden layer with 100 hidden nodes, *Soft ReLU* activation function, and $p = 0.5$ dropout on the hidden layer. As a baseline, randomly assigning tone labels would yield roughly 50 percent accuracy, and randomly assigning topic labels would yield roughly 12 percent accuracy.

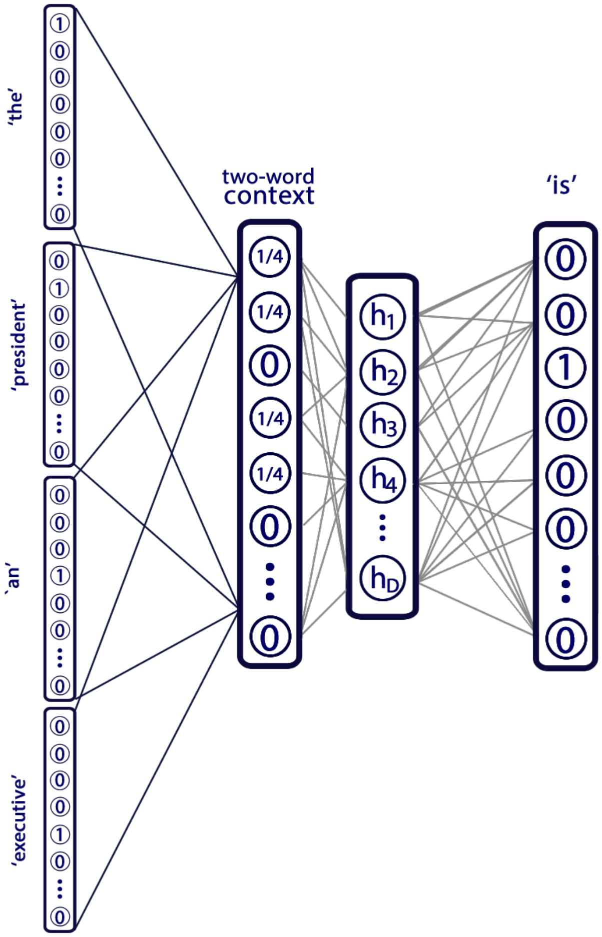
3.6 Figures

Figure 3.1: Graphical Representation of the Continuous Bag of Words Model



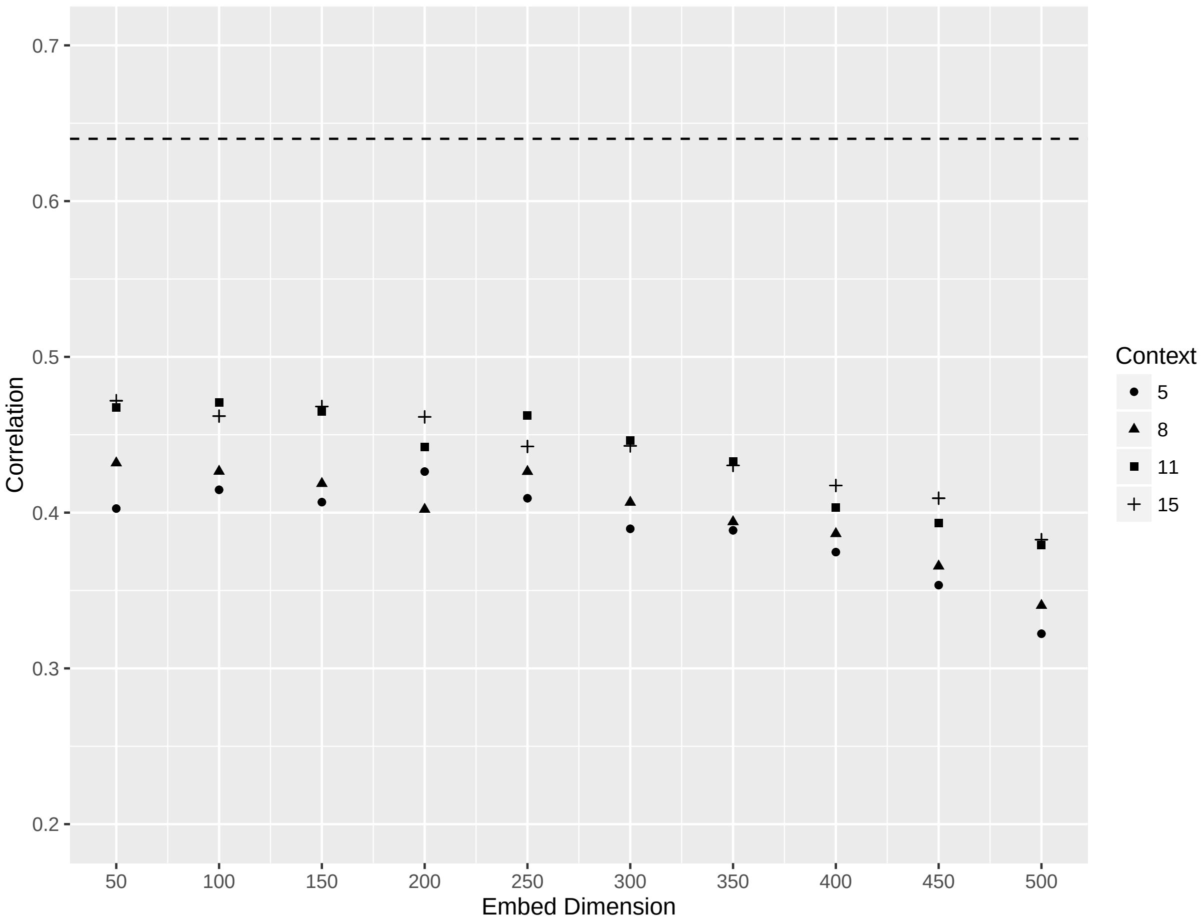
Graphical representation of the continuous bag of words model, simplified to represent the estimation problem for a single word in the corpus. Given a one-hot encoding of the first word in the corpus, 'the', we seek to optimize the $D-$dimensional representation that can still predict the following word to be 'president,' which is also one-hot encoded. The layer of hidden nodes, of which there are D, represents a bottleneck in the network, which forces the model to learn a lower-dimensional representation of the input word that is still capable of predicting the next word. For clarity, not all edges in the graph are shown, but the layers are fully connected.

Figure 3.2: GRAPHICAL REPRESENTATION OF THE CONTINUOUS BAG OF WORDS MODEL WITH TWO-WORD CONTEXT



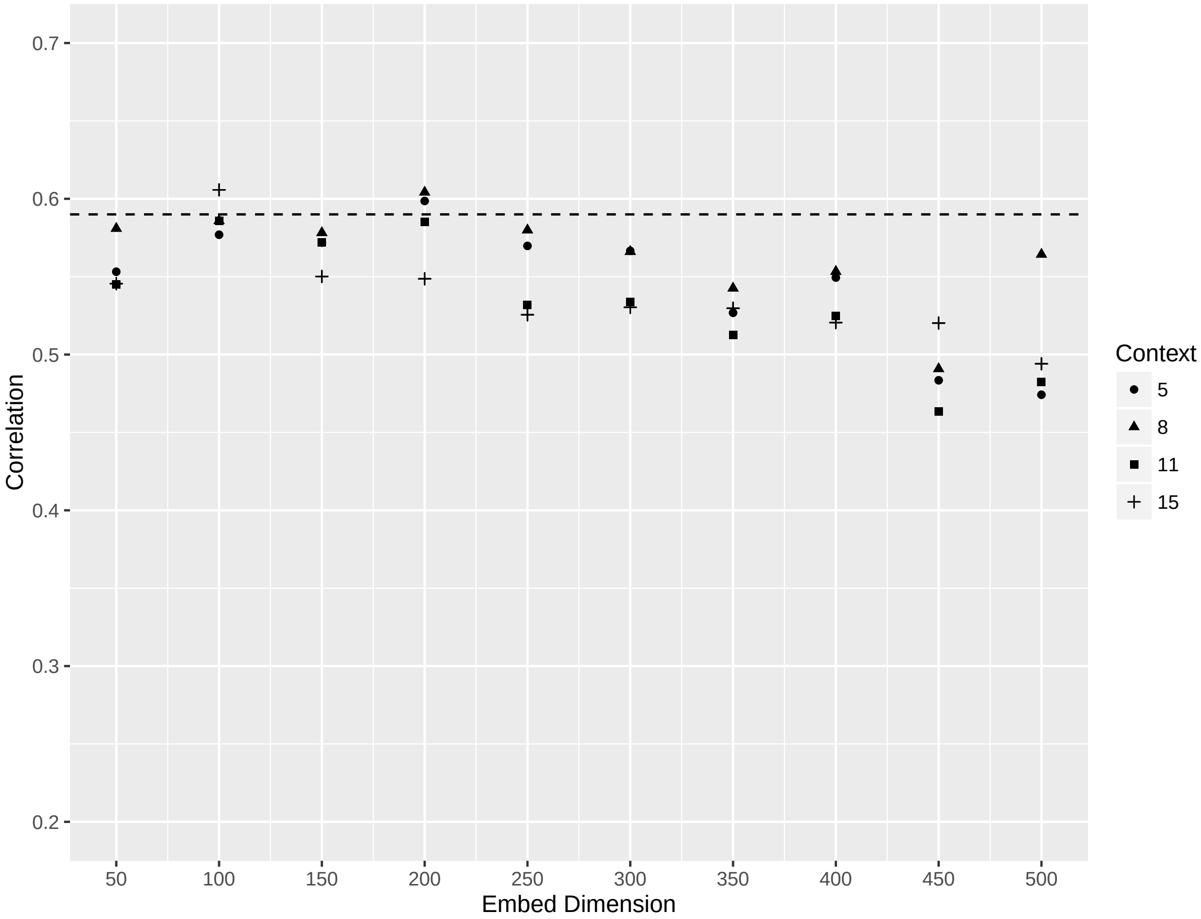
Graphical representation of the continuous bag of words model. The figure shows the use of two-word context window (two words to each side) around the target word. The expression under consideration is 'The president is an executive,' and the target word is 'is.' Each context word is one-hot encoded, and are combined into a single context vector. The rest of the model is the same as in Figure 3.1.

Figure 3.3: MEN EVALUATION RESULTS FOR THE CONTINUOUS BAG OF WORDS MODEL.



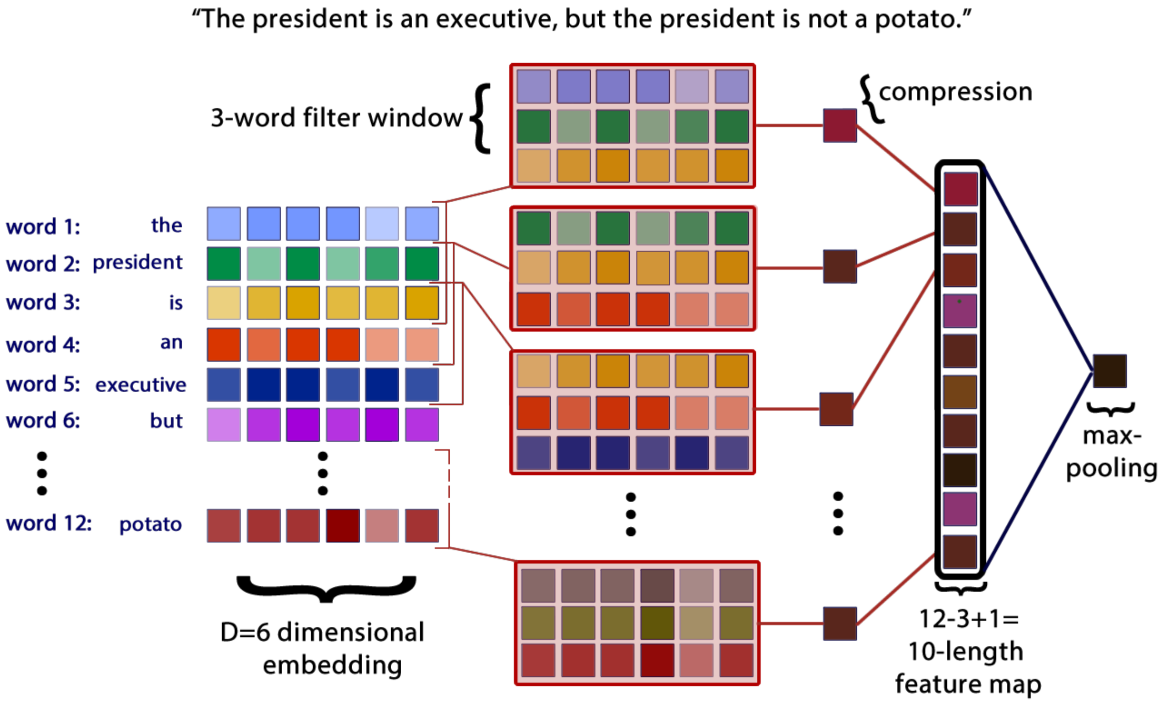
The table gives results from evaluating model performance on the MEN data. The x-axis shows the dimensionality of the word embedding, and the y-axis shows the correlation between the human-coded similarity between words and the model cosine similarity between words. Point shape indicate the context window size (i.e., the number of words to the left and right of a target word used to predict it). The dashed line shows the result from the Google News corpus with context size of 10 and 300-dimensional embeddings.

Figure 3.4: DOMAIN-SPECIFIC EVALUATION RESULTS FOR THE CONTINUOUS BAG OF WORDS MODEL



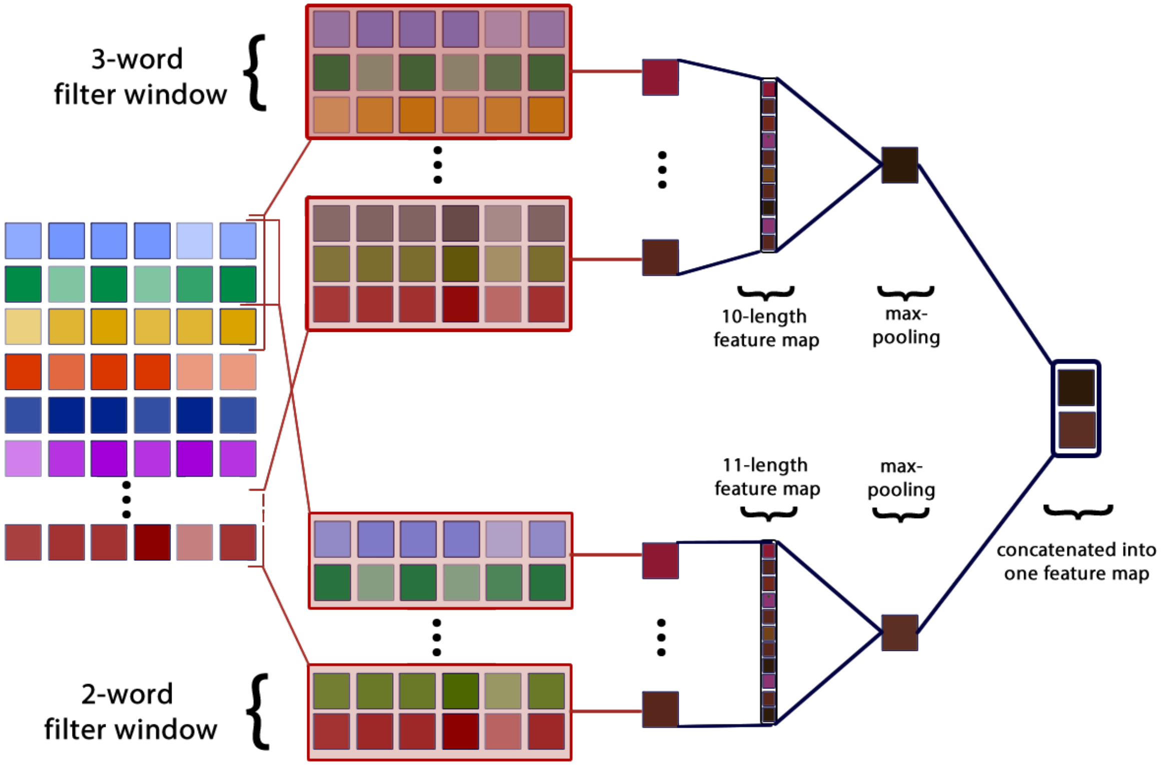
Domain-specific evaluation results for the continuous bag of words (CBOW) model. The table gives the results from evaluating model performance on the domain-specific evaluation data. The x-axis shows the dimensionality of the word embedding, and the y-axis shows the correlation between the human-coded similarity between words and the model cosine similarity between words. Point shape indicate the context window size (i.e., the number of words to the left and right of a target word used to predict it). The dashed line shows the result from the Google News corpus with context size of 10 and 300-dimensional embeddings.

Figure 3.5: A GRAPHICAL REPRESENTATION OF THE CONVOLUTION PROCESS



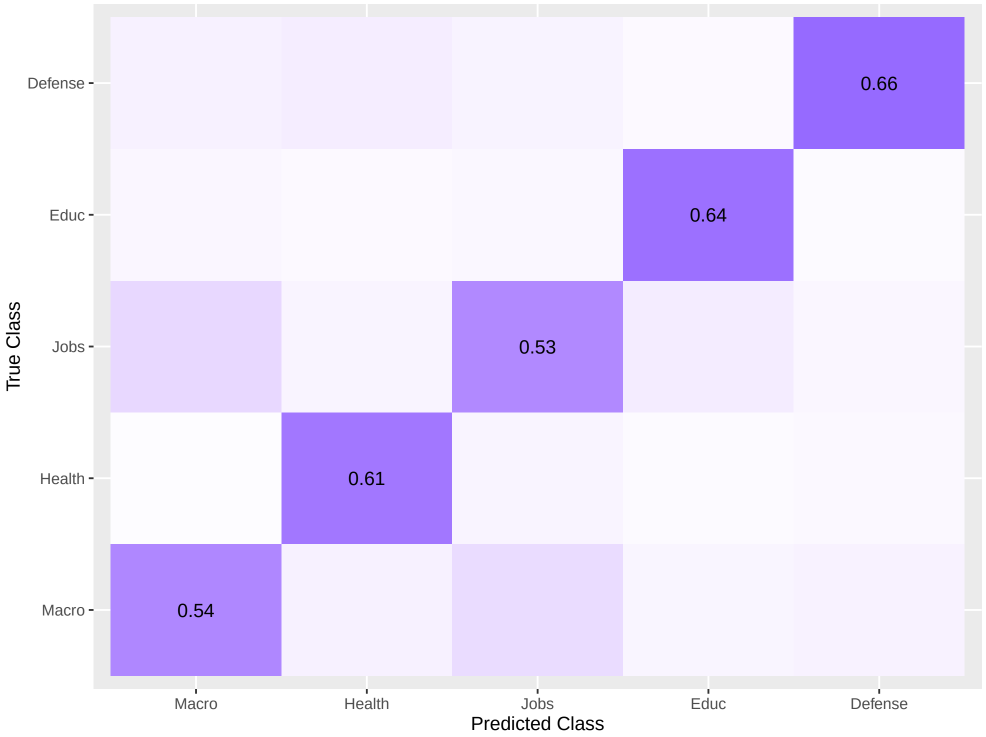
A graphical representation of the convolution process in a convolutional neural network. For this example, imagine that the document is 'The president is an executive, but the president is not a potato.' This is represented by an $N = 12$ word by $D = 6$ dimensional embedding matrix. Colors are unique to the words, and saturation reflects the magnitude of the embedding for a particular word. Figure shows a $K = 3$-word filter window, i.e. applying the filter (the red rectangle) to words (1,2,3), words (2,3,4), et cetera. We move the filter down one word each time, obtaining $N - K + 1 = 10$ filter applications. For each filter application, we compress the resulting matrix by multiplying the filter weights by the each three-word window window, and by summing the product, thus creating a scalar representation of the filter. We then concatenate the resulting compressions into a 10-length feature map.

Figure 3.6: A GRAPHICAL REPRESENTATION OF THE CONVOLUTION PROCESS WITH MULTIPLE WINDOW SIZES



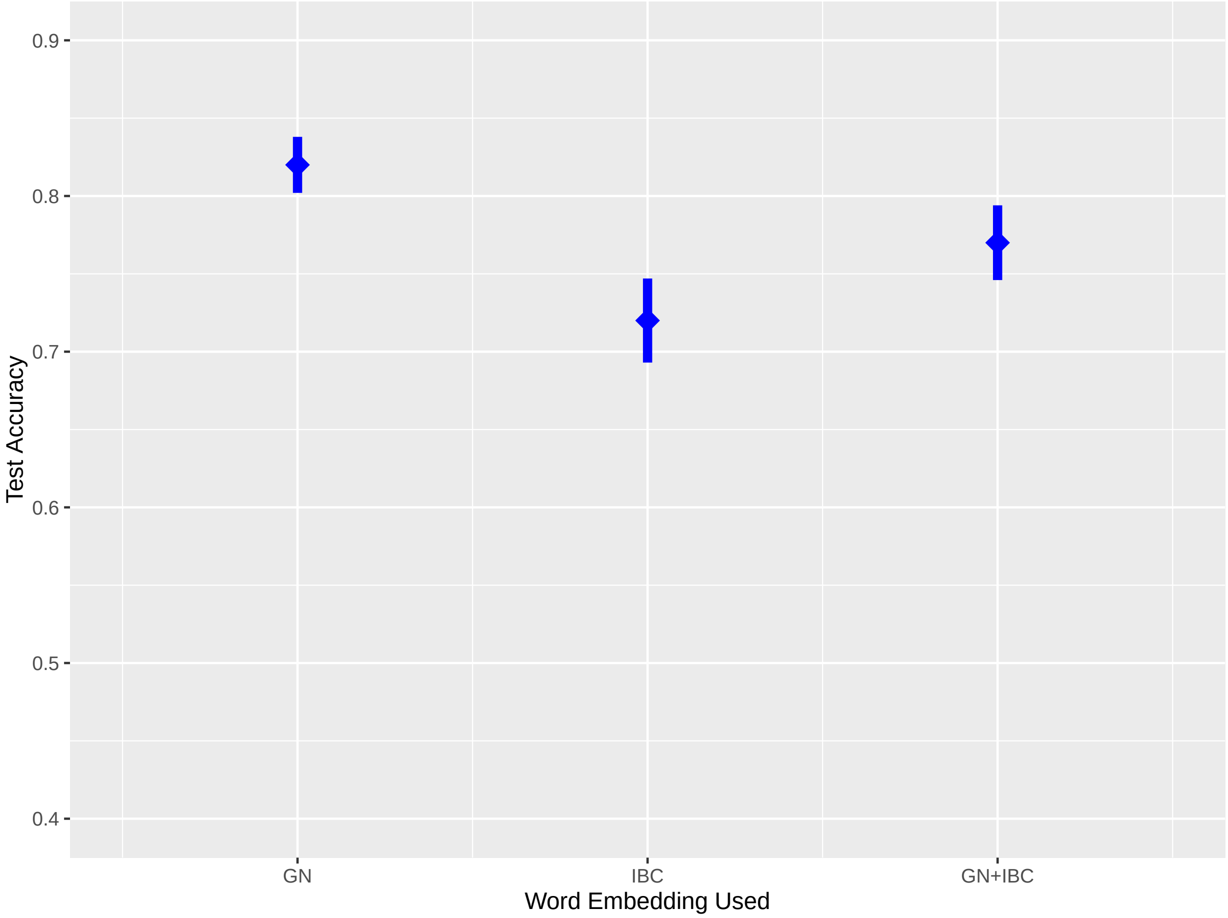
A graphical representation of the convolution process in a convolutional neural network. Building on the intuition in Figure 3.5, this shows the process for using multiple filter widths. As before, we fit a $K = 3$-word filter window, which again produces a 10-length feature map. We also fit a $K = 2$-word filter window, which produces an 11-length feature map. Performing max pooling on each produces a scalar representation of each feature map, which are then concatenated into the final feature map. This process allows us to combine representations built from filter windows of different sizes.

Figure 3.7: TOPIC CLASSIFICATION PRECISION



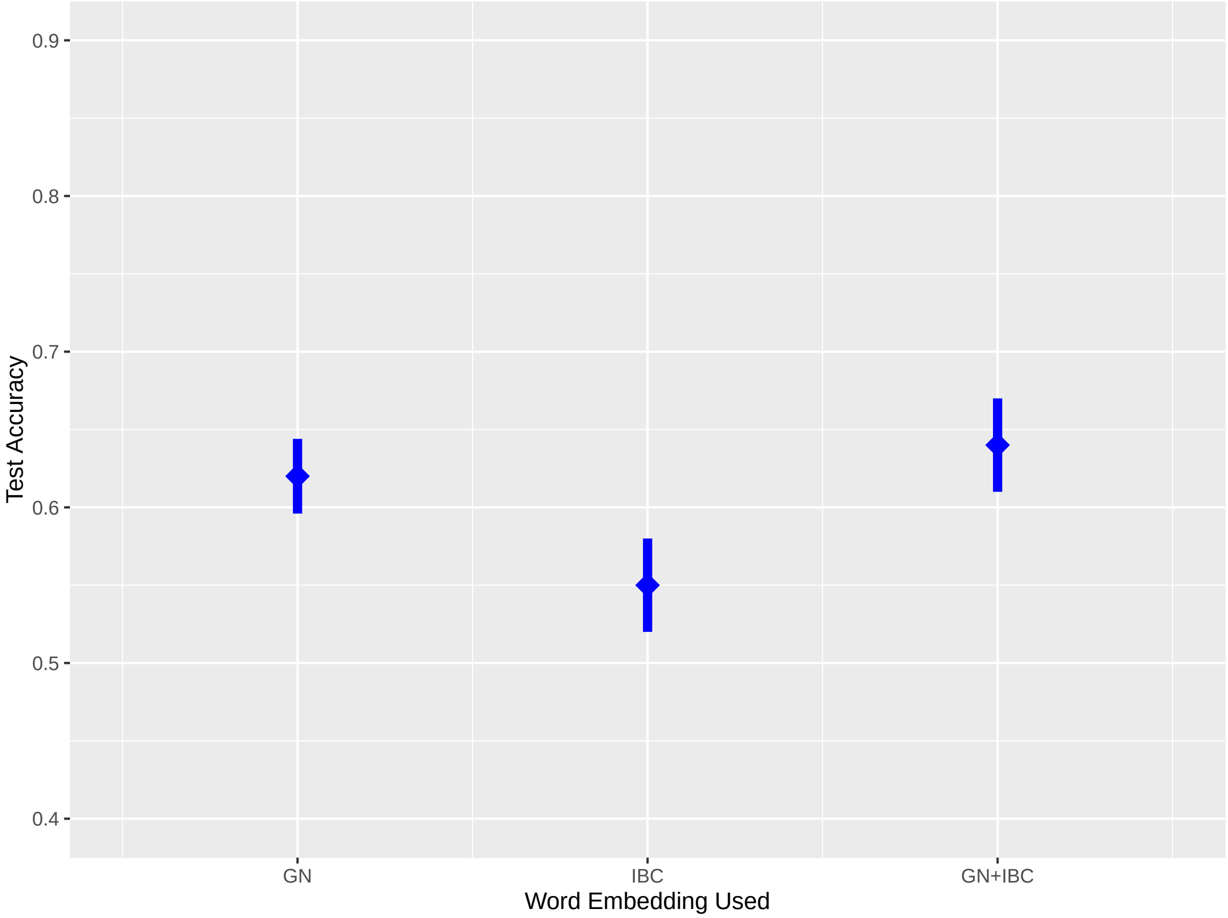
Topic classification precision using the $\mathrm{CNN}_{(2,3,4)}$ model. The topic labels are: macroeconomics, health care policy, labor/jobs, education, and defense/security. Darker colors represent a higher proportion of classifications belonging to a certain cell. The darker colors along the diagonal reflect that most test documents are correctly labeled by the model. The model performs worst on 'macroeconomics' and 'jobs,' which while disappointing, reflects a logical pattern given how similar the two topics are.

Figure 3.8: TONE CLASSIFICATION PRECISION WITH MULTIPLE EMBEDDINGS



Using domain-specific word embeddings decreases tone classification accuracy. Comparing the test set predictive performance of the $\text{CNN}_{(3,4,5)}$ model using various word embeddings, based on 100 cross-validation samples from the ReactLabs debate data. GN indicates that the model uses the Google News-trained word embeddings; IBC indicates Ideological Books Corpus-trained word embeddings; and GN+IBC reflects that the model was fit with two channels, with one channel populated by the GN embeddings and one channel populated by the IBC embeddings. Point estimates are the mean accuracy on the task, and bars represent 95 percent empirical intervals.

Figure 3.9: TOPIC CLASSIFICATION PRECISION WITH MULTIPLE EMBEDDINGS



Using domain-specific word embeddings can improve topic classification. Comparing the test set predictive performance of the $\text{CNN}_{(3,4,5)}$ model using various word embeddings, based on 100 cross-validation samples from the ReactLabs debate data. GN indicates that the model uses the Google News-trained word embeddings; IBC indicates Ideological Books Corpus-trained word embeddings; and GN+IBC reflects that the model was fit with two channels, with one channel populated by the GN embeddings and one channel populated by the IBC embeddings. Point estimates are the mean accuracy on the task, and bars represent 95 percent empirical intervals.

4 ON THE STRUCURE OF ELITE IDEOLOGY

Ideology anchors politics. Nearly every political conflict centers on a set of fundamental political ideas, and the manner in which individuals apply those ideas to their social context. In political science, we have dedicated much ink to the study of ideology. Implicit in this research is an understanding that ideology is foundational to politics, that viewing policy conflict as simply partisan misses the broader point. Ideology is the structure that produces coherent sets of political preferences. Without an understanding of ideology, we are unmoored, drifting from one political conflict to another with little perspective on how these preferences emerged, or what serves to organize individuals' many attitudes.

As a discipline, we seem to have arrived at a basic understanding of how ideology in the United States is structured. Ample evidence from legislative, judicial, state, and public opinion studies shows that our attempts to measure ideology finds that ideology simplifies to a left-right divide. From presidents to individual voters, our measurements of ideology are best described as a scale from very conservative to very liberal. This type of measurement can explain and predict much political behavior. It is both powerful and simple.

Herein lies the problem, to which I contend we have paid far too little attention. Our understanding of ideology stands opposed to our measurement. We follow Converse (1964) in viewing ideology as the framework that organizes and informs our political preferences. Ideology is not the preferences themselves, but the fundamental ideas about politics that organize preferences.

Yet our measurements tend to rely on outward behavior like voting on bills or answering survey questions. These behaviors do not represent ideology alone, but take place in a context where partisan conflict prevails. As a result, the low-dimensional representations of ideology with which we are all so familiar may not actually imply that ideology itself is low-dimensional, but rather that partisan conflict forces political behavior into a one-dimensional space.

In this chapter, I attempt to address this important question. I argue that political ideology is multifaceted, comprising many distinct schools of thought that are not easily summarized in a left-right manner. I conceive of ideology as discrete and hierarchical, comprising the familiar left-right dichotomy, but also comprising subgroups that share some beliefs, while differing on others. Compressing ideological thought to only the coarse left-right distinction omits much of what motivates ideological groups.

Further, just because observable political behaviors by elected leaders are readily summarized in a simple left-right space does not necessarily imply that this is the only way to conceptualize the structure of American political ideology. Political actors may indeed hold more nuanced ideological beliefs, but their behavior is *censored* by a system dominated by two-party politics. Parties control the terms of debate in legislatures, in elections, and are even influential in judicial and bureaucratic decision making. So clearly a left-right divide, which underlies the modern American party system, is a sizable component of political ideology. But the party system may also obscure the finer ideological structure among political elites.

To analyze the structure of elite ideology, I employ automated methods for large-scale text analysis. I analyze the writings of political authors, commentators, and pundits, and I analyze the campaign speeches of presidential candidates from 2008, 2012 and 2016. I show that a discrete and nuanced view of ideology fits the political writings well, meaning that there is more to these expressions of political belief than a simple left-right worldview. When we transition to political speeches, however, that nuanced structure all but disappears. Candidates do not express much in the way of nuanced ideological beliefs, and their political language collapses into a more traditional left-right space.

4.1 Theory

For a concept that political observers understand so intuitively, scholars have myriad, often quite specific, definitions of ideology. Over the past six decades, researchers have posited that ideology comprises all of “abstraction, internal consistency, external contrast, endurance through time, rationality, sophistication, a hierarchical ordering of idea-elements, parsimony—or some combination of these characteristics” (Gerring 1997). This is what Gerring refers to as the ‘problem of definition’ in ideology studies. Yet as Gerring also argues, despite the wide array of definitions in the literature, we gain more theoretical ground by looking for commonalities rather than differences between them.

In this section, I outline how scholars have tended to define ideology, and offer a definition that draws on the dominant themes in the literature. I review how we have measured ideology, the apparent unidimensional structure of ideology, and why I believe these inferences may be misleading. To preview the core of the theory, I ague the following. Many of the sources we use to measure ideology are not unadulterated ideological behaviors. Instead, data such as roll-call votes, campaign or Congressional floor speeches, and the like are partisan behaviors, filtered through and shaped by the political parties. Parties favor conflicts the unite their side against their opponents, which in turn makes the behavior seem one dimensional. This does not imply that ideology itself, at least among elites with highly informed and nuanced belief systems, is unidimensional.

4.1.1 What is ideology?

There are two consistent themes spanning the many definitions of ideology. The first, and most commonly cited, is that of constraint. According to nearly all political scholarship these past sixty years, ideology comprises a coherent set of foundational beliefs about proper societal goals and how best to achieve them (Gerring 1997). We imagine ideology as an “attitude structure, with its parts organized in a coherent fashion” (Campbell, Converse, Miller & Stokes 1960); as “an organization of opinions, attitudes and values” (Adorno, Frenkel-Brunswik, Levinson & Sanford 1950); “a consistent and integrated pattern of thoughts and beliefs explaining man’s attitude toward...his existence in society” (Loewenstein 1953); or “a system of collectively held normative and reputedly factual ideas and beliefs about and attitudes advocating a particular pattern of social relationships and arrangements" (Hamilton 1987).

We could continue in this vein for some time. Even at this point, though, it should be clear that scholarly understandings of ideology center on a common theme. Ideology represents, at least in part, the core beliefs that organize political attitudes and opinions. This 'belief system' anchors political preferences, partisanship and behavior (Campbell et al. 1960, Converse 1964, Adorno et al. 1950, Apter 1964, Rokeach 1968, Billig 1984, Tedin 1987). Ideology is not constraint itself, but the pattern of consistency in beliefs that motivates a similar pattern of consistency in how individuals position themselves on political issues of the day. Or, as Herbert Kritzer put it more simply (1978,487), ideology is the "general principles from which constraint flows."

We generally measure ideology as the sum of many political preferences, which may have desensitized us to the common thread in the academic literature. Ideology is about *ideas* and not preferences. Ideology speaks to *why* we possess the views we do. Ideology is about motivation. Libertarians and religious conservatives will agree on plenty of policy questions. What separates these groups is why they hold those views. Socialists, progressives, religious liberals—they agree on plenty, but secular liberals do not draw the motivation for those preferences from their religious convictions.

Yet another feature remains to ideology, both common in scholarly definitions yet too easily unremarked,. Ideology comprises not simply the psychological beliefs that organize attitudes, but also the psychological attachment to an identifiable group of other individuals. In a 1968 article, Edward A. Shils described ideology using nine criteria, most of which centered on the structure of beliefs, but two of which were "consensus of those that accept [the beliefs]" and an "association with a corporate body intended to realize the patterns of belief" (Shils 1968, 66). Ideology is not just the belief system of the individual, but the groups of citizens who share those beliefs. Indeed, as Noel (2014) points out, the entire purpose behind, and method of, studying ideology is finding patterns of beliefs among many individuals. Ideology in the isolation of an individual mind might be of some psychological interest, but an ideology's ability to influence the political system requires nontrivial popular support (Noel 2014).

Let us then piece together these definitions. On the one hand, ideology comprises the foundational beliefs about politics that define how we view political debates. And on the other, ideology defines how we associate with other citizens—with whom we find common ground, how we personally identify and sort ourselves, and how we comprehend modern political debates. Clearly this definition leaves many questions unanswered. Whence come ideologies? Why do individuals claim certain mantles? For this exercise, these questions remain interesting but unanswered, left for other scholars to agonize over. For this paper, we focus simply on what ideology *is* and why we care about it. Ideology is the core beliefs that organize preferences, and that ties us to groups of like-minded citizens who share our foundational beliefs. Or, as Jost (2006, 653) puts it succinctly, ideology is a "belief system of the individual that is typically shared with an identifiable group, and that organizes, motivates, and gives meaning to political behavior, broadly construed." Ideology comprises both the psychological beliefs that organize our preferences, and the beliefs that tie us to others in society. Neither alone really suffices. We require both consistency in political preferences, and an identifiable group of individuals who share those preferences, to believe that we have encountered an ideology.

4.1.2 Who has an ideology?

Scholars of public opinion have noticed the concomitant problem: if we require consistency of beliefs, and identification with a recognized political group, can we really consider most citizens as being ideological? Most research says no. Foundational studies in political attitudes, for example Campbell et al. (1960) and Converse (1964), argue that most citizens reflect little consistency in beliefs across issue areas or over time. This is not terribly surprising, since Americans show little interest in, or knowledge of, American politics (Delli Carpini & Keeter 1997). Political elites—elected leaders, party elders, opinion leaders, and power brokers—may not only possess more knowledge about politics, but also more ideological thinking generally (Converse 1964, Delli Carpini & Keeter 1997, Kritzer 1978, Jost 2006).

As Jost (2006) argues, however, this may arise from our tendency to 'define away' political ideology. According to Jost, we have constructed such high bars to definitional ideology that we cannot find it in the mass public. Elites may possess constrained attitudes and identify with a clear ideological group, but most of the public does not (Kritzer 1978). That said, plenty of research has found that citizens evince reasonably high levels of stability in their ideological self-identification; that is, they can consistently place themselves on a liberal-to-conservative scale (Conover & Feldman 1981, Kerlinger 1984, Knight 1990, Jacoby 1991). Ellis & Stimson (2012) remark on a consistent pattern in survey responses, namely the 'conflicted conservatives' who hold mostly liberal policy preferences but self-identify as conservative. Yet even this group, whose self-identification is at odds with their preferences, is only remarkable because most respondents show no such inconsistency. The average citizen likely knows little of Rawls and Rousseau, but nevertheless possesses some political attitudes and can identify the opposing camps of American political battlefield (Bishop 2004).

In short, we should not view as exclusive the arguments that ideology requires sophistication, and that average citizens can behave ideologically. Ideology is not a binary state. Political elites should evince more constraint and a clearer identification with groups of like-minded thinkers. Less politically-motivated citizens should evince somewhat less constraint, a fuzzier understanding of ideological groups, and thus a looser identification with those groups. This leaves us with an exciting possibility, namely that the structure of ideology varies: richly nuanced for elites, and simpler for most of the public. This possibility has important implications for how we understand the structure of American political ideology.

4.1.3 What is the structure of ideology?

This leads us to the central question of this paper: what do ideologies look like? Traditional conceptualizations of ideology in political science describe a unidimensional, often defined as a left-right, ideological space (Downs 1957, Abramowitz 1978, Alesina, Roubini & Cohen 1997, Aldrich 1995, Converse 1964, *inter alia*). Indeed, "left-right has been," according to Jost (2006, 654), "the single most useful and parsimonious way to classify political attitudes for more than 200 years."

Researchers have found the parsimony of the left-right ideological scale compelling for decades, and show time and again that such a definition has predictive power in analyzing elite behavior (Hinich & Munger 1996, Aldrich 1995, Downs 1957). The famous Nominate procedure, and its cousins, use latent variable models to project thousands of roll-call votes into a low dimensional space. The first dimension (and at various points in history, the second as well) ably predict future votes in Congress (Poole & Rosenthal 1991, Poole & Rosenthal 1999, McCarty, Poole & Rosenthal 2006, Jackman 2001, Clinton, Jackman & Rivers 2004). Members estimated to be on the right end of the scale match those we would assess to be the most conservative; those on the left tend to be the most liberal; and those in the middle—to the extent that there are any today—match those we would believe to be moderates. The growing gap between parties on the latent dimension reflects our understanding that parties are becoming not only more consistent by partisanship, but also further apart in the foundational beliefs that motivate the members' preferences (Poole & Rosenthal 1984, Poole & Rosenthal 2011).

Similar methods applied to voting in state legislatures, the supreme court, and lower courts recover the same basic pattern (Shor & McCarty 2011, Berry, Fording, Ringquist, Hanson & Klarner 2010, Shor, Berry & McCarty 2010, Martin & Quinn 2002). Across institutions, the behavior of elite political actors is well summarized, and predicted, from a one-dimensional understanding of ideology. If political ideology exists in some more complex form, political leaders seem impervious to ideology's nuance. Leaders act as if the nature of conflict is one-dimensional, and their behavior follows.

And what of the voters? Do citizens hold beliefs that show more diversity than a left-right dichotomy? It does not seem so. In fact, most citizens evince little constrained ideological beliefs at all. Scholars of public opinion have shown that most citizens have little knowledge of, or interest in, politics (Delli Carpini & Keeter 1997). As a result, most citizens evince little consistency—across issue areas or across time—in their expressed opinions, what Converse (1964) called a lack of constraint in political preferences. Rarefied political ideology all but eludes these citizens, but a fairly simple 'liberal to conservative' scale, such as the seven-point ideology scale used in the American National Election Study, does a reasonable job of capturing the general nature of their political beliefs and party preferences (Holm & Robinson 1978).

Even more convincingly, if we examine the aggregate of citizen preferences, the left-right representation is more powerful. Despite the vicissitudes of individual political preferences, if we average over the views of citizens on many political issues, public attitudes are both well-summarized with, and respond to events in, a one-dimensional space (Stimson, MacKuen & Erikson 1995, Erikson, MacKuen & Stimson 2002, Stimson 2004). When national leaders move public policy, or a party wins the White House, the public responds in a one-dimensional space is a way both theoretically and empirically explicable. This result is stunning. Even if individuals do not show much evidence of ideological thinking, the aggregate of individuals shows predictable reaction to public events and, most important for this chapter, the movement occurs in a one-dimensional space.

Seen from this perspective, the low-dimensional representation of ideology appears as *fait accompli*. Citizens, from voters to legislators, judges to presidents, seem to behave as if ideology is effectively a left-right conflict. Perhaps counter-intuitively, then, I maintain that this conceptualization of ideology misses quite a lot. This follows directly from the our working definition of ideology: ideology is about ideas, and there's no compelling reason, *a priori*, to assume that the full scope of political thought should align along a single axis (Marcus, Tabb & Sullivan 1974). Political elites—those who think deeply and often about politics—are not limited in their beliefs to liberalism and conservatism.

We know this intuitively to be true. Religious liberals and secular liberals both share plenty of policy preferences, but the foundational beliefs that inspire those preferences differ markedly. Both groups tend to oppose legally-imposed social traditionalism, both support a strong safety net, and both generally favor an active government role in creating a fair and level economic playing field. For the former, however, these views derive from deep religious conviction, a belief in social justice, and a commitment to Christian charity. For the latter, these preferences arise from non-religious conceptions of fairness and justice, and may well accompany deep skepticism of religious belief. So which, of these two is more liberal—more 'to the left'—in a unidimensional space? That there is no reasonable answer to that question tells us that, at least for elites with highly developed ideological belief systems, one dimension does not suffice.

4.1.4 Parties, elections and ideology.

A *prima facie* appeal does not qualify as an affirmative theory, especially when faced with such evidence to the contrary. If ideology among political elites is multidimensional and group-based, why do we consistently find the "useful and parsimonious" left-right continuum (Jost 2006, 654) when analyzing political preferences and behaviors, across institutions and contexts? To answer that, I rely on two assumptions: (1) the left-right distinction remains the primary orientation of ideological conflict; and (2) ideological conflict takes place in a partisan, institutional, and electoral context.

The first of these assumptions is the easiest to handle. Despite the fine-grained differences between ideological groups—e.g., what differentiates religious and secular liberals—the clearest marker for any ideological group is whether they're fundamentally liberal or conservative in orientation, which Jost (2006) refers to as the 'core' fault line in political conflict. As McClosky & Zaller (1984, 189) explicate, "Politicians and the policies they espouse . . . are usually described as liberal if they seek to advance such ideas as equality, aid to the disadvantaged, tolerance of dissenters, and social reform; and as conservative if they place particular emphasis on order, stability, the needs of business, differential economic rewards, and defense of the status quo." Put more simply, even for conservative ideologies with ample differences in both motivations and preferences, it is far harder to tell the difference between a libertarian and a religious conservative than between a libertarian and a progressive.

The logical conclusion, then, is that ideology is still organized around a left-right orientation. We can think of these as high-level clusters of ideological thought, or as the centroids of belief systems shared by liberal and conservative ideologues. Within these clusters we find the more nuanced distinctions: the socialists, the communists, the progressives, the religious liberals, the libertarians, the populist conservatives, the fascists. This implies a tree-like structure to ideology, with ideological groups nested in broader classes. This view was clearly articulated by Michael Freeden, a political theorist at Oxford University. Per Freeden (2003), ideologies are discrete groups with diffuse borders, clusters of individuals who share fundamental political ideas. Ideological groups may overlap or share certain ideas with other groups, but the groups have unique interpretations of how their ideas relate to political debates (Freeden 2003, Farmer 2006). Freeden also asserts that ideology is hierarchical. There are major (what Freeden calls 'macro') ideological branches with tenets to which large numbers of people subscribe, and shooting off from these are smaller branches of more refined ('micro') clusterings of ideas. Libertarians and far-right reactionaries, for instance, share many 'conservative' principles, but will differ dramatically on others. As a result, the two groups may cooperate on certain issues put before the public, while they will oppose each other on others.

The ample and fundamental differences between liberals and conservatives explains why so much of our measurement uncovers a unidimensional nature to ideological conflict. In short: these measurements do not uncover ideological conflict *per se*, but ideological conflict between two major ideological camps—which align with the two major political parties—rather than between smaller ideological groups. What we observe and use to measure ideology rarely centers on ideology itself, but rather on the conflict between political parties in their quest to obtain and exercise power (Aldrich, Montgomery & Sparks 2014). What is the common thread between legislative roll call votes, presidential elections, Supreme Court decision making, even Congressional floor debates? They all entail conflict between the two fundamental political camps in American politics: Democrats (or their appointees) on the left and Republicans (and their appointees) on the right.

My argument is as follows. Political parties build coalitions, comprising many ideological schools of thought, and points them toward agreeable, collective action. In so doing, parties compress the observable scope of political conflict, from richly nuanced and multidimensional, into a simpler, one-dimensional space. Not all ideologues choose to seek elected office or enter public service. Those that do not may behave—of interest for this paper, they may express their views—in ways that allow us to observe a multidimensional structure to ideology. Those that seek public office, by contrast, need to achieve pragmatic political goals, which incentivizes cooperation over ideological purity. In other words, even if candidates and public officials possess nuanced and multidimensional ideological views, they censor themselves in order to appeal to, and cooperate with, big-tent political parties. Using data that has been filtered through this process will be unable to recover the original multidimensional ideological structure.

Parties facilitate collective political action Politics is, at heart, about getting things done. Individuals have many motivations for engaging in public affairs. They may wish to advance their career, or achieve policy outcomes that benefit them. Or perhaps they are motivated by deeply held beliefs, or moral convictions, or informed assessments about the consequences of policy proposals. Whatever the case, the political system is the arena for achieving political ends.

In a democratic system, one cannot rule by fiat. For a democratic government to function, individuals with diverse beliefs, preferences and aspirations must devise some method for cooperation. Narrow, purist and uncompromising attachment to a set of ideological beliefs stands directly opposed to such cooperation. A government controlled by a relatively small group who share a set of core beliefs and seek to impose them on the public is not a democracy, it's an oligarchy. A government constantly pulled apart by many groups of uncompromising ideologues is an anarchy. Neither works very well.

Political parties provide a solution. Parties bring together individuals, orient them toward shared goals, and help them to overcome the challenges of collective action. Individuals, who may still possess and passionately believe in their ideological creeds, gain an infrastructure to achieve some, if not many, of their political goals. In exchange, they agree to cooperate with individuals representing other ideological predispositions and, if necessary, to abandon some of those preferences that cannot reach broad agreement (Schattschneider 1960). Parties offer a bargain to ideological groups: compromise but see some of your ideas come to fruition, or else starve in the wilderness of ideological purity.

In other words, and more in the parlance of the discipline, political parties serve as aggregators of political interests (Aldrich 1995). Unlike ideological groups, we can assume that parties care more about winning majorities—in elections and in policy-making alike—since power must first be obtained to be exercised (Mayhew 1974, Aldrich 1995, *inter alia*). Particularly in the American context, where our electoral rules favor a two-party system, parties thus become big tents, required to represent a broad range of ideologies and interests[1].

Theoretically, these two parties could orient along any axis. At times in American history, the parties aligned more along class and economic interests, even though this created coalitions that held deeply divergent ideological views on issues such as racial equality and segregation (Aldrich 1995, Poole & Rosenthal 1984, Aldrich, Montgomery & Sparks 2014). In the modern era, the parties have sorted themselves largely along ideological lines. Recall that most voters neither know nor care much about politics, and do not evince a highly structured ideological identity; but many understand the broad distinctions between left and right, and can align themselves thereto. Further, citizens have followed the patterns of ideological polarization and partisan sorting seen among political elites over the past three decades (Abramowitz 2010, Bartels 2000, Levendusky 2009). As Alan Abramowitz (2010) notes, the popular assumption that most citizens are moderates may be a myth. Instead of the unimodal distribution of citizens along a left-right axis, which we have tended to assume for decades, a rather bimodal

---

[1] We note that ideology is not the *only* motivator of political action. Plenty of citizens, and presumably plenty of political elites, do not espouse a particular ideology. Plenty behave in self-interest, or else are motivated by singular policy goals neither constrained nor related to broader, fundamental political beliefs. Parties build coalitions from ideological groups, interest groups and—well, any group that can serve the electoral goals of the party.

distribution may be more appropriate.

This has three important implications for political parties. First, it makes ‘left’ and ‘right’ the most natural orientation for the two major American political parties. After all, it is easier to build a coalition among groups that agree on many, or most, issues, rather than those that disagree on most. Further, as I noted earlier, ideology goes deeper than the preferences of individuals. Ideology is about motivation and beliefs that inspire preferences. For individuals focused on accomplishing political goals, the motivation matters little. It is easier to build a coalition among those who agree on an outcome, even if they disagree on *why* that policy outcome is best. Put heuristically, religious liberals and conservatives will agree plenty that faith serves as the foundation for their preferences. But it’s easier to get religious liberals and socialists to agree on welfare policy than to get religious liberals and religious conservatives to agree.

Second, parties have reasons to eschew ideological purity in favor of broader political appeal. To a certain extent, running in a party primary—which selects the party’s nominee to stand in the general election—ideological purity could work in the favor of a candidate, if the partisan electorate in a particular consistency mostly comprised a single ideological group, In other words, running purely as a libertarian could propel a candidate toward winning a Republican Party nomination, if libertarians comprised the majority of the voting constituency for that office. But even in the case where an ideological purist wins a party nomination, he or she will face a broader and more ideologically-diverse electorate in a general election. A voting public with a coarse orientation toward liberalism and conservatism could find plenty to like in a ideologically-pure candidate, but it is hard to imagine that a ideological purity would be a winning strategy generally. The better approach would, in fact, be for parties to field candidates who—regardless of their internal beliefs—can represent a broader coalition of political perspectives.

Finally—and of particular relevance to ideological measures that rely on legislative roll-call voting—parties tend to structure debate in a way that *looks* one dimensional, regardless of the preferences of the individuals who comprise those parties. The votes taken by legislators, in Congress or state legislatures, does not represent an uncensored expression of individual ideology (Aldrich, Montgomery & Sparks 2014). In an excellent article in *Political Analysis*, Aldrich, Montgomery & Sparks (2014) show that scaling roll call votes nearly always leads researchers to conclude that individuals possess unidimensional policy preferences, especially when individuals act as members of polarized parties. In addition, institutional rules—committee structure, filibusters and cloture votes, and so forth—favor legislation acceptable to (super-) majorities of legislators. Since parties provide the most natural avenue to majority support in a legislature, most of political conflict in legislatures plays out between the two major political parties. This, in turn, gives the appearance of unidimensionality in policy preferences.

As Aldrich, Montgomery & Sparks (2014) argue, though, this is an illusion. In a set of simulation studies, the authors show that even imbuing simulated legislators with high-dimensional preferences leads to unidimensional summaries of roll-call voting behavior, particularly when members have strong partisan attachments. What is more, plenty of research has shown that the various institutional peculiarities in legislatures can affect the structure of policy debate in Congress (Dougherty, Lynch & Madonna 2013, Hurwitz, Moiles & Rohde 2001, Welch & Carlson 1973, Wright & Schaffner 2002, Lee 2009). The particulars of these institutional rules lays beyond the scope of this theory. The broad thrust, however, is this: Parties favor debates that unify their side, and either outright defeat or else divide the opposition. Roll call votes, or even votes taken by the Supreme Court, are not random selections from the pool of possible votes. They are strategic choices by legislators—or again, even justices—that will favor issues around which majoritarian support can be built.

So why do we consistently find unidimensional ideology to be so compelling, so predictive, and so powerful? In all likelihood, it is because of where we have been looking. Ideology itself may well be multidimensional, nuanced and complex. Ideologues, however, face a choice: remain true to those core beliefs, or else compromise and cooperate to achieve political ends. Our measures focus on the behavior of the latter group exclusively. Those who seek elected or institutional power should fare better by being ideologically pluralistic, embracing some compromises to those who share many ideological proclivities but may differ on others. These broad-based coalitions fare better—in essence, they obtain more of their objectives—if they unify along ideological grounds, which in turn makes political conflict appear as a polarized debate between liberals and conservatives. When we observe behavior that has been heavily influenced by partisan considerations, we are not *truly* observing ideology. We are seeing ideology filtered through a partisan lens. Parties give individuals an infrastructure in which to pursue political goals, but they also disincentivize ideological purity. We simply cannot conclude that American political ideology must be unidimensional based on roll-calls, campaign behavior, or citizen preferences.

Why rhetoric? We face another conundrum. If political behavior is tainted by partisan conflict, how can we possibly observe the supposed rich nuance of ideological thinking? It is well and good to contend that ideology *can* be fine-grained, but another problem entirely to observe the differences between ideological groups. I contend—unsurprising to the reader of this thesis—that the best method for analyzing ideological thought is through language. A theory that argues—as I do— that ideology is truly about *ideas* should begin from that premise. Rhetoric gives us a window into the ideas of ideological groups (Gerring 1997, Kritzer 1978, Loewenstein 1953, Thompson 1984). Elite rhetoric tends to be free of corrupting outside influence. Party platforms face a high degree of scrutiny from politicians intent on winning elected office. Speeches by legislators are crafted to help the speaker maintain his or her seat, and the agenda for debate is usually dictated by party leaders. Ideological elites, by contrast, are free to express their core views, and to tie these to issues of the day.

I contend that language gives us three important clues into the ideological perspective of the author: As a human reading a political text, we would estimate the ideology of the author by searching for indicators of (1) The author's fundamental beliefs and how these relate to current political debate; (2) The issues the author finds most compelling and the solutions the author proposes to solve them; and (3) The framing and other rhetorical flourishes the author employs. The first two clearly follow from our working definition of ideology. A person's core beliefs will, at least partially, determine which issues that person finds most urgent to solve, and will also drive them to finds certain solutions more preferable to others. The third comprises the manner in which—specifically the language with which—the author explains his or her beliefs and issue preferences. This can include predominantly focusing on certain frames, like focusing on the law enforcement and crime aspect of immigration. It can also include rhetorical flourishes, such as labeling immigrants as 'aliens' versus 'workers.'

To measure ideology using language, we need to assume that these three facets of political language can be captured by measurable linguistic differences. That is, we must rely on the assumption that political rhetoric contains ideological *signals*, in addition to the noisy, non-ideological content that language contains (Sim et al. 2013). Using the automated methods I describe in the following section, I exploit differences in word frequencies between documents to account for the ideology of the author. This is clearly imperfect. As humans reading political writings, we will look for the clues I mention above, but we would also look for other linguistic features, such as negation, quotation and refutation, or sarcasm. These come naturally to us, but can prove challenging to account for in statistical models. That said, we should bear in mind that the methods do not need to account for *all* signals in the text, so long as it does not miss signals systematically. There is no indication in the results that follow that this is the case.

4.2 Data & Design

The theory I present requires an assessment of ideology in two contexts: ideological language in elite discourse, and in the context of partisan conflict. I analyze two data sources. For elite discourse, I use the Ideological Books Corpus (Sim et al. 2013). The corpus was compiled by Sim et al. (2013) and has been used by Acree et al. (2014) and Iyyer et al. (2014) to study ideological rhetoric in American politics and elections. The corpus contains text from more than 200 books and magazines from political thinkers and commentators. The documents represent a diverse cross-section of ideological thought in America: populist conservatives like Pat Buchanan and Lou Dobbs, religious liberals like Jim Wallis and Rose Agonito, and libertarians like John Stossel and Andrew Napolitano. In total, the corpus contains documents representing the two traditional ideological groups, left and right, an ideological center category, and nine subclassifications. The authors divide conservative authors into: moderates, populists, libertarians, far-right conservatives, and religious conservatives. The authors divide liberal authors into: moderates, progressives, socialists, and religious liberals. The authors' descriptions of the categories, as well as popular exemplars of each category, can be found in Table 4.1.

I pause here to note that the subdivision of ideologies and subideologies is not, and does not need to be, comprehensive. The corpus was constructed from available books and magazines, which means that the ideologies and subideologies it comprises must be sufficiently large to command a commercial audience[2]. As Noel (2014) argues, to merit consideration, an ideology needs to command enough support to influence the political system. The ideologies and subideologies in the Ideological Books Corpus represent ideas espoused by reasonably large and influential groups, and those ideas are sufficiently mainstream that they influence modern political debate. Many voters may find anti-capitalist or anti-immigrant rhetoric to be extreme, but both ideas find their ways into current political discussions.

To assess the ideological expressions in the face of more partisan considerations, I use campaign speech transcripts from the 2008, 2012 and 2016 presidential campaigns. The collection includes campaign speeches delivered by all major candidates for office, in the primary and

---

[2] Some largeer ideological groups, like environmentalists, communists or anarchists, are not represented in the corpus. The authors undoubtedly either did not find sufficient writings to represent these groups, or else decided that they were readily incorporated into other groups, like progressivism or socialism. In future, the corpus can be extended to accommodate other ideological groups.

general elections, from both major political parties in 2008 and 2012, as well as primary campaign speeches from candidates Ted Cruz, Donald Trump, Bernie Sanders and Hillary Clinton in the 2016 primary.

Presidential campaign speeches offer two advantages. Clearly these speeches qualify as political expressions in an electoral and partisan context, providing necessary contrast with the books and magazines in the Ideological Books Corpus. These speeches also offer an opportunity for ideological nuance. Most speeches were delivered during primary campaigns, not during the general election contest between the two major party nominees. Primaries do not center on inter-party conflict, as we might expect from Congressional floor speeches or general election campaign speeches, but between candidates from the same political party. If there were a time in American elections where we would expect to consistently see professional, well-articulated expressions of nuanced political beliefs, it would be during a contested primary for high office.

4.2.1 Methods

The analysis unfolds in two parts, each of which addresses a question posed by this chapter. First, is there a multiclass structure to elite political ideology? Second, does this structure persist in campaign rhetoric, or does it rather reduce to a simpler left-right dimension? For both questions, we essentially face a predictive task. Just as with roll-call, judicial vote, or public attitude scaling procedures, we seek out the lowest-dimensional representation that still has predictive power. In the famous Nominate procedure, for example, we conclude that one or two latent dimensions suffice to summarize roll-call voting behavior because that low-dimensional representation of votes can be used to reliable predict new votes. Put another way, the variance in votes can be accounted for by a low-dimensional representation. For this chapter, our task is the same: using the language from political ideologues, can we reliably predict the ideological group of an author? And can that prediction be easily summarized in a low-dimensional space?

To address elite ideology, I employ several supervised machine learning methods for extracting patterns in language from a corpus. The Ideological Books Corpus contains documents, each of which comes with a hand-labeled ideological class (left, right, center) and ideological subclass (e.g., religious left, libertarian). The basic strategy, then, is to find the language that best partitions the authors based on their assigned ideological class and subclass.

I first use results from Sim et al. (2013), who employ a modified version of the Sparse Additive Generative model described by Eisenstein, Ahmed & Xing (2011). The sparse additive model regresses the frequency with which words appear in the documents on the ideological classification for the documents, using sparsity-inducing priors to select the words or multi-word phrases useful in the classification task. To ensure that the discovered structure is not simply a relic of the sparse additive model, I also employ two deep-learning methods for modeling the ideological content of the texts: a deep feed-forward artificial neural network model, as described in Chapter 1, and a convolutional neural network, as described in Chapter 2. Despite the different structure and assumptions of the three models, the results from these methods are quite similar. Since the sparse additive model has been used with this data in other published studies, I use the sparse additive model for my primary analysis unless otherwise noted. The sparse additive model also brings the advantage of explicit weights on the words in the vocabulary (unlike the deep-learning methods which contain many weights which makes it difficult to ascertain the relationship between particular words and particular ideological classes and subclasses), which will make it straightforward to assess the substantive differences in language between ideological groups.

To model the ideological content of campaign speeches, I use the patterns learned in modeling the Ideological Books Corpus to predict the ideological subclassifications of the presidential campaign speeches. This step follows naturally from the first: for the sparse additive model, the feed-forward model, and the convolutional model, I simply estimate the conditional probability that each speech belongs to one of the ideological classes and subclasses, based on the words or multi-word phrases contained in each. By using the language from the Ideological Books Corpus to model the campaign speeches, I also make it as likely as possible to discover a multiclass ideological structure in an electoral context, thereby falsifying the theory that electoral pressure compresses ideology to primarily a left-right space.

For these models, the text needs to be initially processed. I follow Sim et al. (2013), and use their method for processing the text for use in the sparse additive and feed-forward models. This includes stemming, normalizing numbers (i.e., replacing all numbers with a single placeholder) and retaining only two- and three-word phrases. For more information, refer to the Sim et al. (2013) article. For the convolutional model, I normalize numbers, but do not stem or *n*-gram the corpus, for the reasons described in Chapter 2. I take as the unit of analysis the books and magazines parsed at the paragraph-level for the Ideological Books Corpus, and at the speech-level the campaign speeches. For cross-validation purposes, I partition the Ideological Books Corpus into an 80 percent training set and 20 percent test set. The partition is random, but I ensure that the training corpus is balanced between the subclasses, which speeds convergence of the deep learning models.

4.2.2 Expectations

I hypothesize that ideology among elites is not readily captured by a left-right dimension, but rather that this simplification arises from the nature of partisan conflict. The most natural fault line in American politics divides the two major parties, and the typical voter does not take particular interest in, nor have much knowledge of, politics. Accordingly, electoral rhetoric will be simpler, and lower-dimensional, than other elite language.

If the structure of ideology among political elites does not reduce to a simple left-right dimension, we should be able to detect meaningful differences in how ideologues express their views. We can attempt to study these differences qualitatively, but the methods I propose bring the benefits of quantitative rigor. If there are real differences in language to be found, statistical models should be able to exploit those differences, and reliably predict the ideological class and subclass of documents, based solely on the language those documents contain. Further, the document-level predictions should reflect substantively-meaningful relationships between the classes and subclasses. Documents belonging to the left subclasses should be more closely related to each other than those on the right, meaning we should see fewer mis-classifications among documents within the same broad class than between the classes.

We should expect a different pattern from the speeches. If partisan and electoral pressure tends to collapse debate into a left-right space, the speeches should reflect this in two ways. First, we should find it harder to predict subideological labels for political speeches. In other words, we should find that major candidates for the presidency do not represent niche ideologies in their campaign speeches, but rather seek to represent broader themes of liberalism and conservatism. Second, if we look for underlying patterns among candidates, we should find that their political language reduces to a straightforward left-right space. If we evaluate the language and it is easily, and substantivaly, summarized in a low-dimensional space, we might conclude that electoral competition has reduced a complex ideological playing field into a partisan, one-dimensional battle ground.

4.3 Results

Let us begin with the simpler of the two questions. Recall that we have a random partition of the corpus, which contains more than 170,000 paragraphs from the Ideological Books Corpus, into a training set and a testing set. A reasonable test for whether real differences in language exist between the ideological subclasses is whether the hand-labeled ideological subclass is *learnable* from language alone. Can we correctly classify withheld documents into the classes and subclasses, as assigned by Sim et al. (2013)? The answer to this question is, broadly speaking, yes.

Table 4.2 summarizes the cross-validation results for the three models described above: the sparse additive model due to Eisenstein, Ahmed & Xing (2011), the feed-forward artificial neural network, and the convolutional neural network. The top panel shows the test set predictive accuracy using only the coarser ideological classes—i.e., can we classify the documents correctly into left, right, and center, ignoring the ideological subclasses?—while the bottom panel shows the accuracy when predicting ideological subclasses.

We note two features from the table. First, the subclass predictive accuracy, across three models with very different assumptions, are all fairly high. Because the sample was balanced before fitting the model, the best, modal-subclass prediction would yield accuracy 0.10. Clearly these models have found differences between the subclasses that aid in prediction. I will address momentarily whether those differences seem to carry substantive meaning.

Second, we do notice that all three models find it easier to predict left-right classifications than to predict the subclassifications. We could interpret this as undercutting the theory I have presented: namely, that ideology is mostly about left and right, and the other distinctions are less theoretically meaningful. After all, we would prefer simpler explanations where possible—there is always nuance in the political world, but our scientific goal is to describe the forest and not the trees. In this case, however, I would argue that the subideologies are sufficiently distinct as to be theoretically meaningful. We should expect that left and right are easier to classify than these more refined subclasses. It is also easier to tell the difference between and dog and a horse than it is to tell the difference between a poodle and a pitbull, but that doesn't make the taxonomy irrelevant.

Whether the subideological taxonomy is theoretically meaningful rests in large part on whether there are substantively important differences between the subclasses that cannot be easily summarized in a lower-dimensional space. To address the first part, let us refer to Tables 4.3 and 4.4. The sparse additive model attempts to find the features—i.e., the words or multi-word phrases—that are most useful in predicting that a document belongs to each ideological subclass. The tables show the top phases associated with each subclass. Clearly the model is finding, by and large, meaningful phrases that match our intuitive understanding of these ideological subclasses. The phrases that distinguish libertarian authors include property rights, the founding fathers, and other phrases associated with constitutionalism and individual liberty. The phrases associated with religious conservatism include Jesus Christ, Christian nation, and references to social concerns like abortion.

To address the second question, we can analyze the classification predictions themselves; or more precisely, the variance of the predictions. If the structure of the elite ideology is much more easily summarized in a left-right dimension, then the variance in the classification predictions should be mostly accounted for by differences between left and right. Recall that each document in the test set has a predicted probability of belonging to each of the ten ideological subclasses. I assess the top principal components of the predictions to see if a single dimension captures most of the predicted probabilities. Consider the solid line in Figure 4.1. The figure shows the cumulative proportion of variance captured by the first ten Thomponents, and as we can clearly see, no single component seems to capture the predictions well. In fact, we require seven components to capture 90 percent of the variance in the predictions. This is rather inconsistent with a theory that the underlying structure of ideology, in these elite writings, is low-dimensional.

Do these principal components represent the ideological subclasses? Generally speaking, yes. I present a sample of two principal components for illustration, though the pattern is largely the same for all components. Figures 4.2 and 4.3 show how documents load on two of the principal components (the third and sixth respectively). Along the x-xis, the test set documents are ranked by their loading on the component, and the y-axis shows the actual loadings. The principal components are of course inductively learned, but we can tell them apart by analyzing which documents receive the highest loadings. Panel (a) differentiates libertarian documents from all others by representing them as dark circles. Clearly, most of the documents receiving the highest loadings on the third principal components are those by libertarian authors. For the sixth principal component, most of the high-loading documents (again, represented as dark circles) were written by progressive authors.

Figure 4.4 shows another interesting pattern in the predictions. If we plot the loadings for each document on the fourth and eighth principal components, we see three distinct clouds of documents. The dark "x" points mark documents hand-coded as religious conservative, the "+" points mark documents hand-coded as religious liberal, and the lightly-shaded points represent all other documents in the test set. What we see is a clear relationship between religious language—both religious liberal and religious conservative documents receive higher scores on the eighth principal component. The classes are, however, still distinct, with religious liberal documents tending to have lower loadings on the fourth component, while religious conservative documents have higher loadings.

The pattern in campaign rhetoric is quite different. Each campaign speech has been classified into one of the ideological subclasses, using the estimated models fit to the Ideological Books Corpus. As with the books test set, this yields a predicted probability that each of the speeches belongs to each of the ideological subclasses, which I then aggregate up to the candidate-level. The appendix has tables which show the proportion of speeches, by each candidate, classified into each of the subclasses, but this makes for a somewhat overwhelming summary, and it turns out to be unnecessary.

Figure 4.5 shows the candidates' language by summing up the proportion of candidate language classified as being from one of the liberal subclasses (x-axis) and the proportion classified as being from one of the conservative subclasses (y-axis). The candidates do not show much of a pattern of using language from particular ideological subclasses, but they do evince a clear distribution along a left-right dimension. Candidates like Ted Cruz, Michele Bachmann and Ron Paul are predicted to be at one extreme, while Joe Biden, Bernie Sanders and John Edwards make up the other. It appears that the subclassifications can be readily summarized in a left-right way. Referring back to the dashed line in Figure 4.1, this intuition is born out. By using Principal Components Analysis on the speech classifications, we see that a single component accounts for 60 percent of the variance in predictions, and two components account for nearly all of it.

Taken together, we see evidence for both propositions in this chapter. There are real, substantive differences between the ideological groupings included in the Ideological Books Corpus. These ideological groups use distinct language to represent their core political beliefs, and to address the issues most central to those beliefs. These differences are not simply qualitative, but measurable, and they do not collapse into a single left-right dimension.

When we switch attention away from political tracts and toward language in an electoral context, however, we return to a one-dimensional world. Presidential candidates from the 2008-2016 campaign cycles use ideological language, but do not seem to represent particular ideological subclasses. Even candidates like Ron Paul, who self-describes as a libertarian, does not seem to use that much language similar to libertarian authors (see Table 4.8 in the Appendix). Instead, candidates tend to sample their language from across the ideological subclasses. This makes sense: even in a primary campaign, candidates face an incentive to be broadly appealing. This electoral pressure incentivizes candidates to present themselves, not as niche ideologues, but in simpler—and for measurement purposes, lower-dimensional—way.

No single principal component well summarizes the use of language by ideologues, yet a single dimension does well capture patterns in candidate language. Ideology is thus both multifaceted and low-dimensional, nuanced and simplified, all at once. Outside of electoral and partisan pressure, ideologues can express the nuanced views that distinguish ideological groups from one another. In the face of partisan conflict, however, ideological nuance falls away and the familiar left-right summary reappears.

4.4 Discussion

I hope to have presented evidence for a rather simple theory. As scholars and close observers of politics, we know that ideology is richly nuanced, and that a single- or even two-dimensional representation of ideology does not adequately capture what is ultimately a group-based identity (Noel 2014). Yet in so many applications, ideology reduces to a left-right, continuous dimension. From roll call votes to Supreme Court decisions to the public mood, ideology comprises attitudes of elites and average citizens, arrayed so neatly from left to right.

As this chapter argues, however, the truth is more complicated. We have not found evidence for rich and nuanced ideological thought because we have been searching in the wrong places. Or, to be more precise, we have found ideology in its electoral and partisan form, because we have searched primarily in places heavily influenced by electoral and partisan concerns. Most voters care little about politics, and know only the broad partisan strokes of current debate. Elected officials, bureaucrats, and judges, who concern themselves with public opinion and institutional legitimacy, and who derive much of their political viability from the support of political parties, thus engage in what appears to be a one-dimensional conflict. They censor their expressions in ways that appeal to their party bases and leaders, and in ways that can gain support from their fellow partisans.

When we seek out the ideas of ideological groups, expressed on their own terms and outside of immediate electoral pressure, what we find is stark differences between even similar ideological groups. To be sure, ideology is still well-described in terms of left and right. But ideology also appears to be hierarchical, and the subdivisions between classes are sufficiently distinct that we can find both meaningful and quantitatively-predictable boundaries between ideological groups. When we attempt to find a clean linear subspace with these groups, such a summary is elusive.

Yet not all language shows rich nuance of ideological thought. When we return to partisan political conflict, represented in this chapter by the campaign speeches by major party candidates for the presidency, language reduces back to a left-right dimension that fairly cleanly maps onto our substantive understanding of ideology. This result as been replicated on Congressional speeches, and reached the same conclusion (Diermeier et al. 2012). Where the search for a low-dimensional representation of ideology fails with political books and magazine articles, it succeeds with speeches.

We arrive, then, at the most important question for any scientific enterprise, and in a sense at the conviction of this thesis: to what avail? Does this show anything that is useful, relevant and important to how we understand politics? My belief, and my hope, is that the answer is yes. For far too long, we have thought that political ideology reduces to primarily a left-right dichotomy. This representation is powerful, and is true in many contexts. By focusing on votes or elections, though, we deprive ourselves of an theoretical—and measurable—understanding of the political ideologies that seek to influence modern political debate.

4.5 Tables

Table 4.1: DESCRIPTIONS OF THE SUBCLASSES IN THE IBC

Subclass	**Description**	**Exemplars**
Left-Progressive	Sometimes critical of Democratic party from left, but often still vote Democratic. Highly concerned about crony capitalism, growing inequality, money in politics, labor unions, racial discrimination, gay rights.	Ed Schultz, Rachel Maddow; Paul Krugman, Arianna Huffington, the *Nation* and *American Prospect* magazines.
Left-Moderate Liberal	Liberal on some issues, but more committed to compromise and supportive of financial/business interests, and military intervention.	Bill Bradley, Evan Bayh, Jim Webb, *The New Republic* magazine.
Left-Religious Liberal	Religious, but with liberal political leanings and a commitment to social justice, egalitarianism, inclusiveness and altruism.	*Soujourners Magazine*, authors like Jim Wallis and Ronald Sider.
Left-Socialist	Favors government intervention in economic matters, often advocating state-run industries, universal health care, and an extensive safety net. Also characterized by deep skepticism of capitalism, banking and wealth disparities.	Noam Chomsky, Richard Wolff, the *International Socialist Review*
Centrist	Heavily critical of political polarization and hyper-partisanship, often complain about the dysfunction in the political system, typically critical of money in politics, and politically independent.	Thomas Mann, Norman Ornstein, Charlie Christ, Charles Wheelan

Table 4.1 (continued)

Subclass	**Description**	**Exemplars**
Right-Moderate Conservative	Typically critical of talk radio and tea party influence, unhappy with perceived radicalism among Republican elected officials and leadership and worried that taking a hard-line on social issues, and immigration These ideologues tend to take environmental issues seriously and support stem-cell and other scientific research.	Christine Todd Whitman, David Frum, Megan McCain, Andrew Sullivan, Joe Scarborough, Ross Douthat, Reihan Salam.
Right-Populist	Appeal to working class values and includes cultural conservatives concerned about immigration and the perceived decline of Western and White-Anglo culture, suspicious of political correctness, not completely convinced of the benefits of free trade or globalism. These ideologues also tend to highly value nationalism and patriotism.	Lou Dobbs, Bill O'Reilly, Pat Buchanan
Right-Libertarian	Favor limited government, individual freedom and personal liberty; critical of government overreach. Often economically conservative while socially liberal.	Ron Paul, Rand Paul, John Stossel, Judge Anthony Napolitano, *Reason* magazine
Right-Religious Conservative	Religious, and politically conservative and typically literalist in Biblical interpretation. Often draw on scripture to justify socially conservative positions.	Pat Robertson, Rick Santorum, Ralph Reed, Family Research Council

Table 4.1 (continued)

Subclass	**Description**	**Exemplars**
Right-Far Right	Talk radio conservatives, who tend to mix elements of libertarianism, religious conservatism, national populism, and cultural traditionalism. These ideologues also tend ot be enamored of patriotic symbols and suspicious of leaders' insufficient patriotism or tendency to say anything critical of U.S. Of all on in the conservative classes, these ideologues tend to be the firmest believers in conspiracy theories, particularly as relate to liberal political figures.	Rush Limbaugh, Sean Hannity, Laura Ingraham, Glenn Beck, Michael Savage

The ideological subclasses in the Ideological Books Corpus, along with the authors' description of each ideological subclass and well-known representatives of each subclass.

Table 4.2: IDEOLOGICAL CLASSIFICATION ACCURACY ACROSS THREE MODELS

Model	**Accuracy**
Three Ideological Classes	
Sparse Additive Model	0.75
Feed-forward Model	0.72
Convolutional Model	0.80
Ten Ideological Subclasses	
Sparse Additive Model	0.64
Feed-forward Model	0.65
Convolutional Model	0.71

Test set predictive accuracy for three classification models: the sparse additive model described in Eisenstein et al. (2011) and Sim et al. (2013), a feed-forward artificial neural network with 2 layers and 50 hidden nodes in each layer, and a convolutional neural network with filter sizes 3, 4, and 5, and 100 hidden nodes.

Table 4.3: THE TOP WORDS & PHRASES ASSOCIATED WITH LIBERAL IDEOLOGIES

Progressive	**Socialist**	**Moderate**	**Religious**	**Centrist**
United States	United States	United States	biological family	Long Beach
American Prospect	monopoly capitalism	modern art	progressive religion	debt limit
Abu Ghraib	class struggle	young woman	nuclear family	stock option
executive director	occupy movement	eighteenth century	bad theology	country music
public info	political economy	al Jazeera	religious issue	average American
State Department	capitalist system	Mitt Romney	early church	corporate America
public policy	trade union	twentieth century	religious community	original intent
Vice President	labor movement	great deal	American creed	tax increase
interest rate	ruling class	free market	Willow Creek	Mark Levin
Head Start	developing country	tax code	strict father	American dream
United Nations	public option	free trade	early Christians	Wal Mart
mental illness	working people	common law	Mother Theresa	super PAC
John Kerry	Dr. King	supply siders	family values	George Washington
San Francisco	American worker	South Africa	God's love	loan office
Iraq war	economic crisis	living conditions	tax collector	Republican Party
child care	social justice	K Street	NUM_Matthew	Glenn Beck
civil war	working class	interest rate	NUM_Mark	Washington, DC
political power	fossil fuel	Grover Norquist	NUM_Luke	Alexander Hamilton
community college	corporate state	private sector	NUM_John	debt ceiling

Phrases that most clearly distinguish each ideological subclass from the others, based on the class- and subclass-effects estimated using sparse additive model. Many of the terms match the expectations of area experts, with the exception the rather anemic centrist category. The difficulty in extracting meaningful 'centrist' words largely stems from the relatively few truly centrist authors of political tracts.

Table 4.4: The Top Words & Phrases Associated with Conservative Ideologies

Modertae	**Populist**	**Religious**	**Far-right**	**Libertarian**
Governor Bush	illegal aliens	Christian nation	illegal immig.	raw milk
Ronald Reagan	illegal immigrants	human beings	North Korea	property rights
human being	border patrol	Jesus Christ	flat tax	chief justice
foreign policy	immig. reform	anti-Christian	political correct	natural right
Middle East	birth rate	New York	big gov't	founding fathers
John McCain	national media	God's word	affirmative action	eminent domain
radio program	civil war	Sarah Palin	Jewish state	Supreme Court
Saddam Hussein	Los Angeles	fed. gov't	human life	constitutional right
Repub. Party	white America	San Franscisco	Saudi Arabia	TSA agent
mass destruction	African American	private property	Arab Spring	Ninth Circuit
Karl Rove	open border	elementary school	radio show	bear arms
Cold War	trade deficit	Holy Spirit	American exception	free speech
Sam's Club	elected officials	year old	left wing	Habeas Corpus
look back	corporate America	pro life	mainstream media	Fourth Amendment
Social Security	free trade	Joseph Smith	popular culture	Fourteenth Amendment
Miss America	culture war	public schools	global warming	judicial activism
Iraq war	special interest	Planned Parenthood	hard work	economic freedom
mind set	national interest	Judeo-Christian	Muslim Brotherhood	Federal Reserve
class voter	working men	daily saint	Ron Paul	medical marijuana
health care	border security	Jim Walis	foreign aid	defense attorney

Phrases that most clearly distinguish each ideological subclass from the others, based on the class- and subclass-effects estimated using the first-stage SAGE model. Many of the terms match the expectations of area experts, with some confusion between Far Right and Populist classes.

Table 4.5: IDEOLOGICAL CLASSIFICATIONS & SUBCLASSIFICATIONS CRUZ, TRUMP, CLINTON, AND SANDERS IN 2016

	Cruz	Trump	Clinton	Sanders
All Liberal	0.20	0.44	0.55	0.55
All Conservative	0.72	0.50	0.42	0.35
Center	0.08	0.06	0.02	0.07
Socialist	0.05	0.08	0.19	0.13
Moderate	0.06	0.15	0.07	0.11
Progressive	0.00	0.18	0.22	0.24
Religious Liberal	0.03	0.00	0.07	0.07
Far-Right	0.21	0.10	0.06	0.08
Libertarian	0.11	0.10	0.09	0.09
Moderate Conservative	0.07	0.17	0.15	0.11
Populist	0.08	0.16	0.07	0.04
Religious Conservative	0.19	0.07	0.05	0.03

The ideological classifications (top panel) and subclassifications (bottom) for the top four candidates in the 2016 presidential campaign. Entries represent the proportion of speeches classified into each ideological class and subclass, according to the sparse additive model (the results from the neural network models look similar). Rounding may result in columns that do not sum exactly to 1.

Table 4.6: IDEOLOGICAL CLASSIFICATIONS & SUBCLASSIFICATIONS FOR DEMOCRATIC PRESIDENTIAL CANDIDATES IN 2008

	Obama (P)	Obam (G)	Biden	Clinton	Edwards	Richardson
All Liberal	0.59	0.42	0.73	0.68	0.68	0.58
All Conservative	0.41	0.53	0.24	0.19	0.28	0.37
Center	0.00	0.05	0.03	0.13	0.04	0.05
Socialist	0.06	0.07	0.03	0.01	0.02	0.10
Moderate Liberal	0.19	0.09	0.02	0.01	0.14	0.01
Progressive	0.05	0.08	0.08	0.15	0.11	0.13
Religious Liberal	0.12	0.03	0.03	0.02	0.01	0.05
Far-Right	0.05	0.01	0.02	0.00	0.01	0.01
Libertarian	0.05	0.19	0.01	0.02	0.01	0.01
Moderate Conservative	0.09	0.12	0.01	0.02	0.03	0.03
Populist	0.12	0.06	0.01	0.01	0.01	0.06
Religious Conservative	0.05	0.09	0.01	0.01	0.02	0.04
Republican	0.00	0.00	0.00	0.00	0.00	0.00

The ideological classifications (top panel) and subclassifications (bottom) for Democratic candidates in the 2008 presidential campaign. Proportions were estimated using the the sparse additive model (the results from the neural network models look similar). "G" indicates the candidate's speeches in the general election, and "P" indicates the primary campaign, where that difference is necessary. Entries represent the proportion of speeches classified into each ideological class and subclass. Rounding may result in columns that do not sum exactly to 1.

Table 4.7: IDEOLOGICAL CLASSIFICATIONS & SUBCLASSIFICATIONS FOR REPUBLICAN PRESIDENTIAL CANDIDATES IN 2008

	Romney	Giuliani	Huckabee	McCain(P)	McCain (G)
All Liberal	0.49	0.31	0.29	0.20	0.31
All Conservative	0.51	0.58	0.65	0.70	0.47
Center	0.00	0.10	0.06	0.10	0.22
Socialist	0.01	0.02	0.00	0.04	0.04
Moderate Liberal	0.00	0.04	0.02	0.02	0.02
Progressive	0.02	0.03	0.01	0.02	0.04
Religious Liberal	0.20	0.03	0.02	0.04	0.01
Far-Right	0.01	0.05	0.03	0.02	0.02
Libertarian	0.06	0.05	0.07	0.04	0.04
Moderate Conservative	0.17	0.06	0.06	0.39	0.12
Populist	0.04	0.09	0.04	0.06	0.05
Religious Conservative	0.13	0.03	0.12	0.01	0.02

The ideological classifications (top panel) and subclassifications (bottom) for Republican candidates in the 2008 presidential campaign. Proportions were estimated using the the sparse additive model (the results from the neural network models look similar). "G" indicates the candidate's speeches in the general election, and "P" indicates the primary campaign, where that difference is necessary. Entries represent the proportion of speeches classified into each ideological class and subclass. Rounding may result in columns that do not sum exactly to 1.

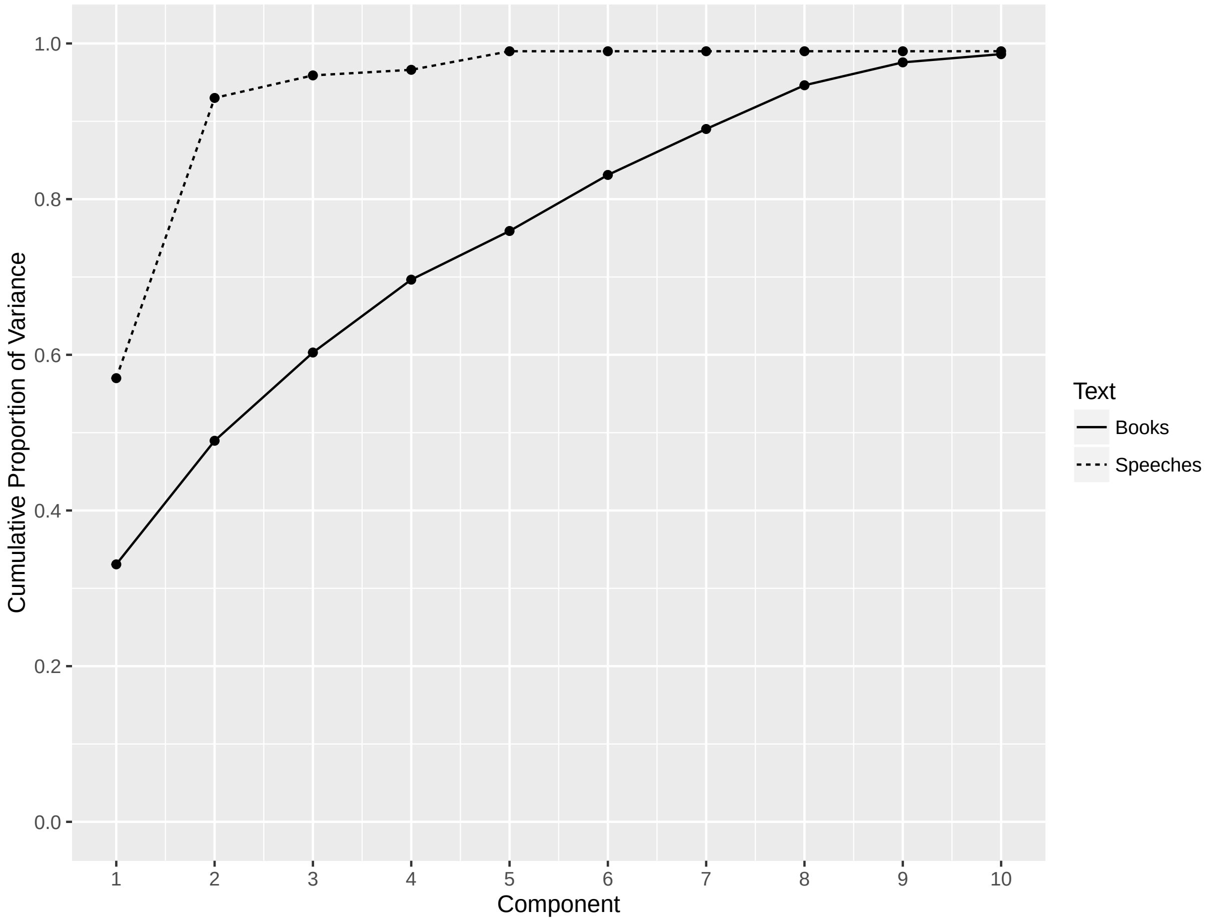
Table 4.8: IDEOLOGICAL CLASSIFICATIONS & SUBCLASSIFICATIONS FOR REPUBLICAN PRESIDENTIAL CANDIDATES IN 2012

	Romney (P)	Romney (G)	Bachmann	Cain	Huntsman	Paul	Pawlenty	Perry	Santorum
All Liberal	0.33	0.48	0.24	0.22	0.46	0.26	0.53	0.30	0.30
All Conservative	0.59	0.49	0.69	0.56	0.46	0.63	0.39	0.60	0.54
Center	0.08	0.03	0.07	0.22	0.07	0.11	0.07	0.10	0.16
Socialist	0.00	0.13	0.02	0.01	0.01	0.01	0.05	0.02	0.01
Moderate Liberal	0.01	0.02	0.02	0.04	0.02	0.02	0.04	0.02	0.01
Progressive	0.01	0.01	0.03	0.02	0.01	0.02	0.03	0.03	0.02
Religious Liberal	0.02	0.04	0.03	0.04	0.10	0.01	0.08	0.02	0.01
Far-Right	0.08	0.11	0.08	0.05	0.04	0.03	0.03	0.05	0.03
Libertarian	0.14	0.05	0.12	0.02	0.02	0.07	0.06	0.07	0.04
Moderate Conservative	0.00	0.01	0.04	0.05	0.05	0.03	0.07	0.09	0.05
Populist	0.01	0.02	0.02	0.02	0.02	0.01	0.01	0.02	0.02
Religious Conservative	0.02	0.00	0.05	0.08	0.02	0.07	0.07	0.04	0.03

The ideological classifications (top panel) and subclassifications (bottom) for Republican candidates in the 2012 presidential campaign. Proportions were estimated using the the sparse additive model (the results from the neural network models look similar). 'G' indicates the candidate's speeches in the general election, and 'P' indicates the primary campaign, where that difference is necessary. Entries represent the proportion of speeches classified into each ideological class and subclass. Rounding may result in columns that do not sum exactly to 1.

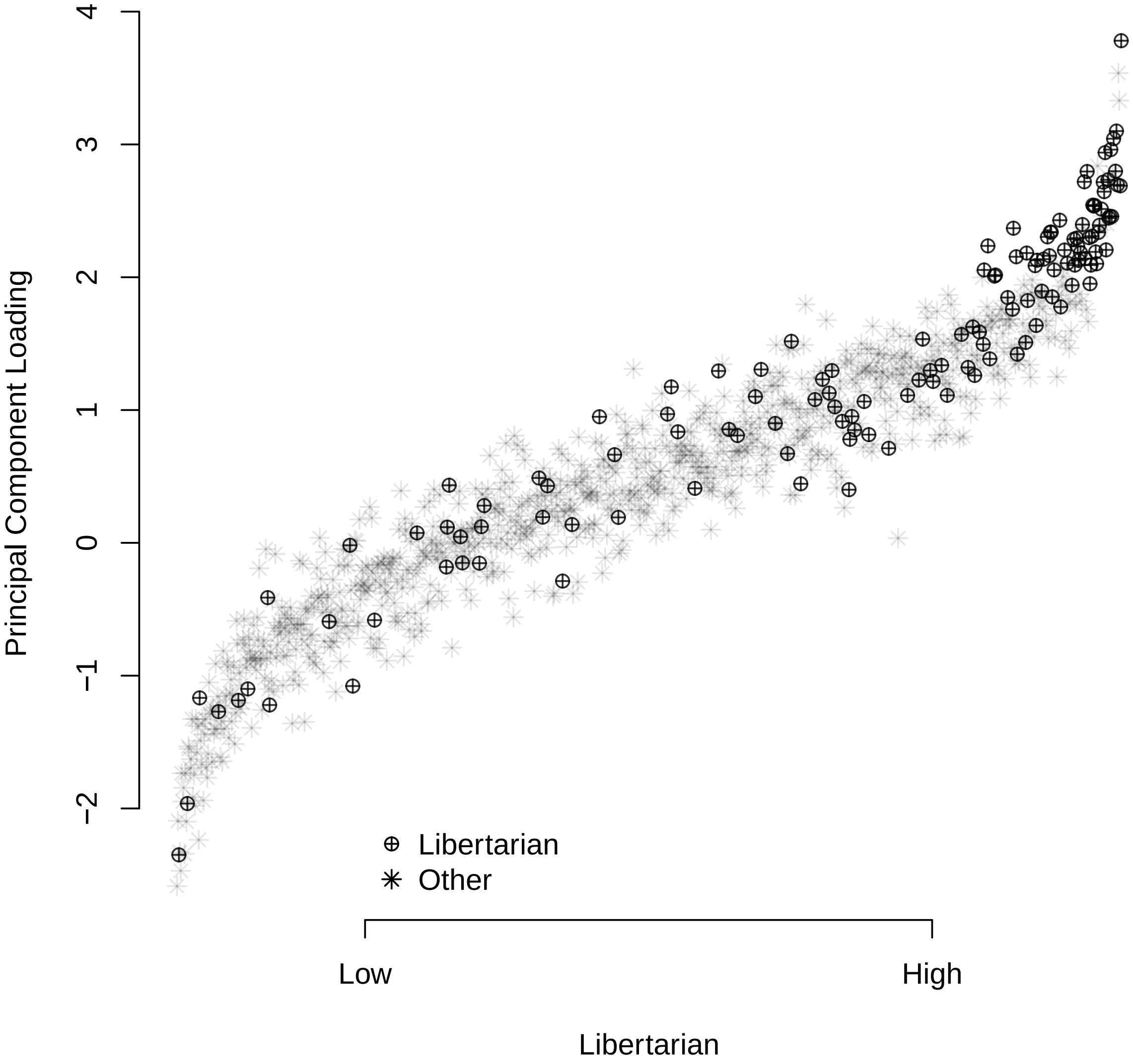
4.6 Figures

Figure 4.1: POLITICAL BOOKS EVINCE HIGHER DIMENSIONALITY THAN CAMPAIGN SPEECHES



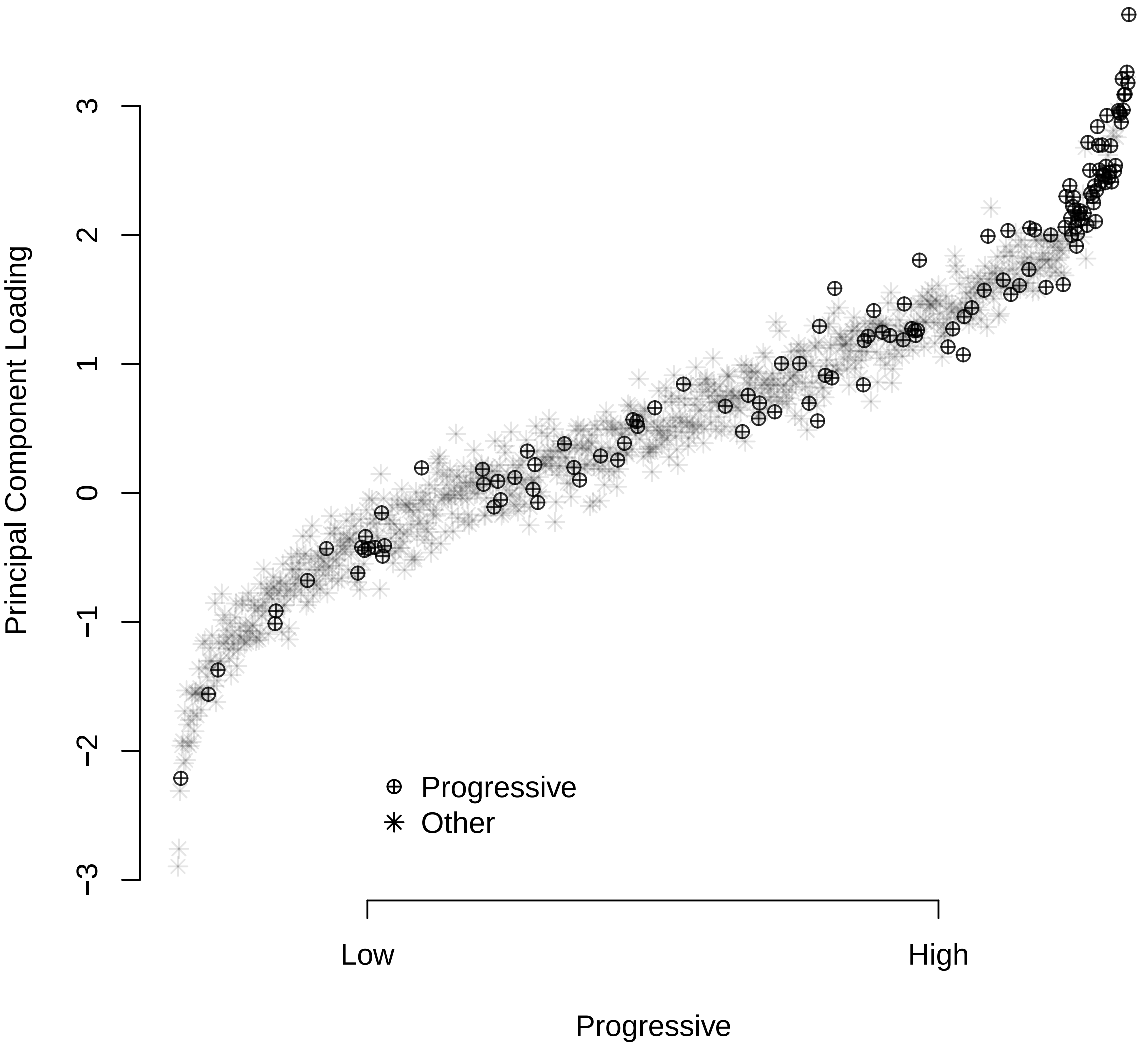
The y-axis shows the proportion of the cumulative proportion of variance explained by the number of principal components along the x-axis. The solid line represents the the classifications of the Ideological Books Corpus test set, and the dashed line represents the classifications of the presidential campaign speeches into the nine ideological subclasses. To account for variation in the ideological subclassifications in campaign speeches, one component generally suffices.For the ideological subclassifications on test documents in the Ideological Books Corpus, we require nine principal components to account for as much variation as two components in political speeches.

Figure 4.2: THE THIRD COMPONENT CAPTURES THE UNDERLYING LIBERTARIAN IDEOLOGICAL DIMENSION



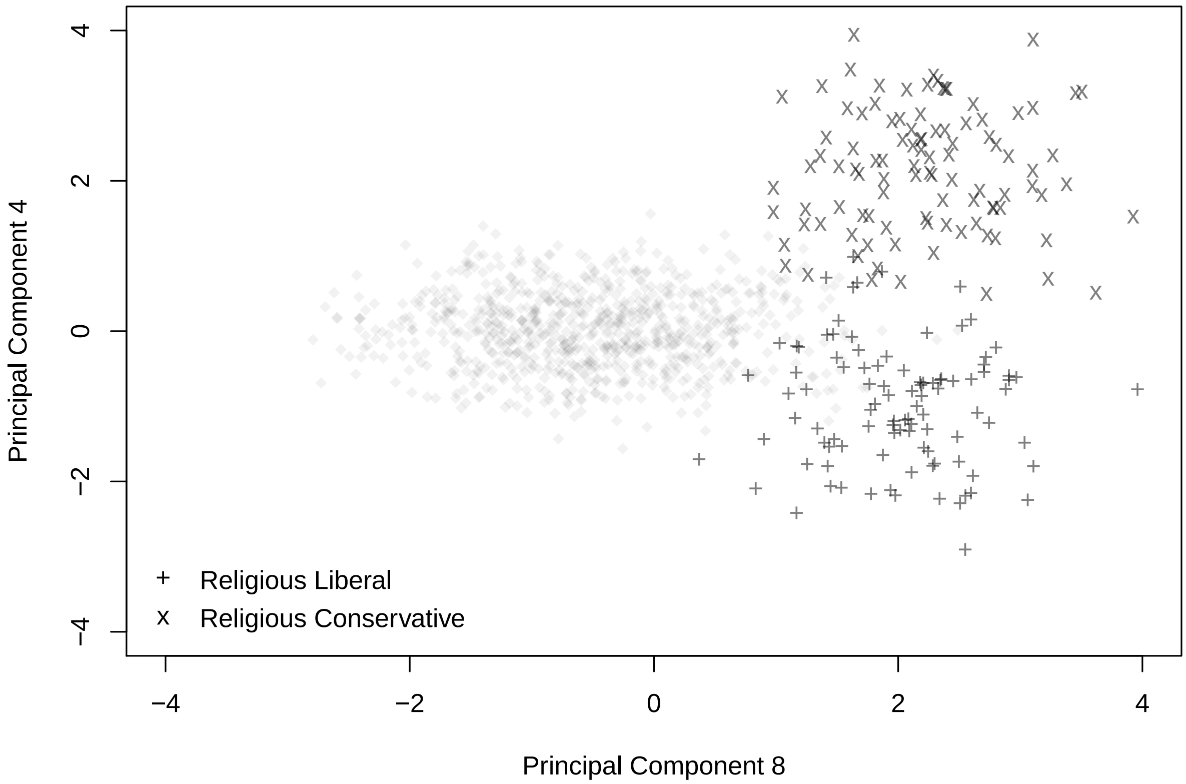
The figure shows the principal component loadings on the third principal component for the Ideological Books Corpus test set predictions. Each point represents a document, and the x-axis indexes the documents ordered by their loadings onto the third principal component. Libertarian documents are represented by the dark circles, and we see that most documents with high loadings are from libertarian authors.

Figure 4.3: THE SIXTH COMPONENT CAPTURES THE UNDERLYING PROGRESSIVE IDEOLOGICAL DIMENSION



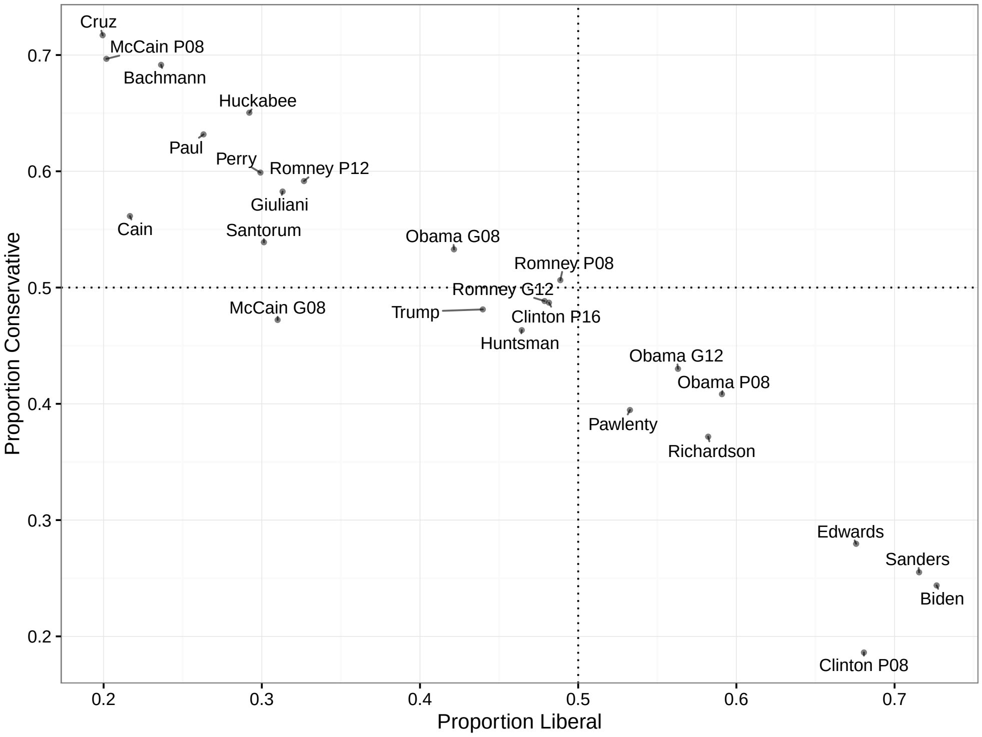
The figure shows the principal component loadings on the sixth principal component for the Ideological Books Corpus test set predictions. Each point represents a document, and the x-axis indexes the documents ordered by their loadings onto the sixth principal component. Progressive documents are represented by the dark circles, and we see that most documents with high loadings are from libertarian authors.

Figure 4.4: TWO LATENT DIMENSIONS CAPTURE RELIGIOSITY IN LANGUAGE



The figure shows the principal component loadings on the eighth (x-axis) and fourth (y-axis) components. Each point represents a document in the Ideological Books Corpus test set. The 'x' marks religious conservative texts, and the '+' marks religious liberal texts. The lighly-shaded points represent all other documents. The figure shows the relationship between religious language—both that there is an association between religious language, whether conservative or liberal, and that there are clearly differences between the two subclasses.

Figure 4.5: THE AGGREGATED LEFT-RIGHT CLASSIFICATIONS OF PRESIDENTIAL SPEECHES



The proportion left is the sum the percentages of speech sentences classified as one of the liberal ideological subclasses, and the same for the proportion right. While the candidates do not sample from particular ideological subclasses (i.e., there is no candidate who samples their language predominately from a particular subclass), there is a coherent left-right orientation of the candidates. Democrats tend to use mostly liberal language, Republicans mostly conservative, and as noted in Acree et al. (2014) note, McCain, Romney and Obama move ‘toward the center’ when moving from the primaries (denoted as ‘P’) to the general election (denoted as ‘G’).

REFERENCES

Abramowitz, A.I. 1978. “The Impact of a Presidential Debate on Voter Rationality.” *American Journal of Political Science* pp. 680–690.

Abramowitz, Alan I. 2010. *The Disappearing Center: Engaged Citizens, Polarization, and American Democracy.* Yale University Press.

Acree, Brice, Justin Gross, Noah Smith, Yanchuan Sim & Amber Boydstun. 2014. Testing the Post-Primary Moderation Hypothesis. In *American Political Science Association Annual Meeting. August 28-31*.

Adorno, Theodor W, Else Frenkel-Brunswik, Daniel J Levinson & R Nevitt Sanford. 1950. “The Authoritarian Personality.”.

Aldrich, John H. 1995. *Why Parties? the Origin and Transformation of Party Politics in America.* Vol. 15.

Aldrich, John H, Jacob M Montgomery & David B Sparks. 2014. “Polarization and Ideology: Partisan Sources of Low Dimensionality in Scaled Roll Call Analyses.” *Political Analysis* 22(4): 435–456.

Alesina, Alberto, Nouriel Roubini & Gerald Cohen. 1997. *Political Cycles and the Macroeconomy.* MIT Press.

Ansolabehere, Stephen, Shanto Iyengar, Ann N Crigler, Thomas M Holbrook, Robert Huckfeldt & John Sprague. 1999. “Going Negative. How Political Advertisements Shrink & Polarize the Electorate.”.

Apter, David Ernest. 1964. *Ideology and Discontent.* Free Press New York.

Bartels, Larry M. 2000. “Partisanship and Voting Behavior, 1952-1996.” *American Journal of Political Science* 44(1): 35–50.

Beauchamp, Nicholas. 2011. How to Scale Legislatures With Text: A Comparison of Methods, With Applications to the US Congress and UK House of Commons. Technical report Working Paper.

Beauchamp, Nicholas. 2015. “Predicting and Interpolating State-Level Polls Using Twitter Textual Data.”.

Belue, Lisa M & Kenneth W Bauer. 1995. “Determining Input Features for Multilayer Perceptrons.” *Neurocomputing* 7(2): 111–121.

Bengio, Yoshua. 2012. “Practical Recommendations for Gradient-Based Training of Deep Architectures.” In *Neural Networks: Tricks of the Trade.* Springer. pp. 437–478.

Bengio, Yoshua, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin & Jean-Luc Gauvain. 2006. “Neural Probabilistic Language Models.” In *Innovations in Machine Learning.* Springer. pp. 137–186.

Bermingham, Adam & Alan F Smeaton. 2011. On Using Twitter to Monitor Political Sentiment and Predict Election Results. In *Workshop on Sentiment Analysis Where AI Meets Psychology*. IJCNLP.

Berry, William D, Richard C Fording, Evan J Ringquist, Russell L Hanson & Carl E Klarner. 2010. "Measuring Citizen and Government Ideology in the US States: A Re-Appraisal." *State Politics & Policy Quarterly* 10(2): 117–135.

Billig, Michael. 1984. "Political Ideology: Social Psychological Aspects." *The Social Dimension* 2: 446–470.

Bishop, George F. 2004. *The Illusion of Public Opinion: Fact and Artifact in American Public Opinion Polls*. Rowman & Littlefield Publishers.

Blei, David M. 2012. "Probabilistic Topic Models." *Communications of the ACM* 55(4): 77–84.

Blei, David M, Andrew Y Ng & Michael I Jordan. 2003. "Latent Dirichlet Allocation." *The Journal of Machine Learning Research* 3: 993–1022.

Boureau, Y-Lan, Jean Ponce & Yann LeCun. 2010. A Theoretical Analysis of Feature Pooling in Visual Recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. pp. 111–118.

Box, George EP. 1979. "Robustness in the Strategy of Scientific Model Building." *Robustness in Statistics* 1: 201–236.

Boydstun, Amber E, Justin H Gross, Philip Resnik & Noah A Smith. 2013. "Identifying Media Frames and Frame Dynamics Within and Across Policy Issues.".

Boydstun, Amber E, Rebecca a Glazier, Matthew T Pietryka & Philip Resnik. 2014. "Real-Time Reactions to a 2012 Presidential Debate a Method for Understanding Which Messages Matter." *Public Opinion Quarterly* 78(S1): 330–343.

Bruni, Elia, Nam-Khanh Tran & Marco Baroni. 2014. "Multimodal Distributional Semantics." *J. Artif. Intell. Res.(JAIR)* 49(1-47).

Bullinaria, John a & Joseph P Levy. 2007. "Extracting Semantic Representations From Word Co-Occurrence Statistics: A Computational Study." *Behavior Research Methods* 39(3): 510–526.

Bullinaria, John a & Joseph P Levy. 2012. "Extracting Semantic Representations From Word Co-Occurrence Statistics: Stop-Lists, Stemming, and SVD." *Behavior Research Methods* 44(3): 890–907.

Campbell, A., Phillip E. Converse, W.E. Miller & D. Stokes. 1960. *The American Voter*. New York: Wiley.

Cao, Ziqiang, Sujian Li, Yang Liu, Wenjie Li & Heng Ji. 2015. A Novel Neural Topic Model and Its Supervised Extension. In *AAAI*. pp. 2210–2216.

Chauvin, Yves & David E Rumelhart. 1995. *Backpropagation: Theory, Architectures, and Applications.* Psychology Press.

Chen, Tianqi, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang & Zheng Zhang. 2015. "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems." *arXiv Preprint arXiv:1512.01274* .

Clinchant, Stéphane & Florent Perronnin. 2013. Aggregating Continuous Word Embeddings for Information Retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and Their Compositionality.* pp. 100–109.

Clinton, Joshua, Simon Jackman & Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." *American Political Science Review* 98(02): 355–370.

Collobert, Ronan & Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks With Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning.* ACM pp. 160–167.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu & Pavel Kuksa. 2011. "Natural Language Processing (Almost) From Scratch." *The Journal of Machine Learning Research* 12: 2493–2537.

Collobert, Ronan & Samy Bengio. 2004. Links Between Perceptrons, MLPs and SVMs. In *Proceedings of the Twenty-First International Conference on Machine Learning.* ACM p. 23.

Conover, Pamela Johnston & Stanley Feldman. 1981. "The Origins and Meaning of Liberal/conservative Self-Identifications." *American Journal of Political Science* pp. 617–645.

Converse, Philip E. 1964. "The Nature of Belief Systems in Mass Publics. in Ideology and Discontent, Ed. David Apter. New York: Free Press.".

Delli Carpini, Michael. & S. Keeter. 1997. *What Americans Know About Politics and Why It Matters.* Yale University Press.

Diermeier, Daniel, Jean-François Godbout, Bei Yu & Stefan Kaufmann. 2012. "Language and Ideology in Congress." *British Journal of Political Science* 42(01): 31–55.

Dos Santos, Cícero Nogueira & Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *COLING.* pp. 69–78.

Dougherty, Keith L, Michael S Lynch & Anthony J Madonna. 2013. "Partisan Agenda Control and the Dimensionality of Congress." *American Politics Research* p. 1532673X13511109.

Downs, A. 1957. *An Economic Theory of Democracy.* New York: Harper and Row.

Druckman, James N, Martin J Kifer & Michael Parkin. 2010. "Timeless Strategy Meets New Medium: Going Negative on Congressional Campaign Web Sites, 2002–2006." *Political Communication* 27(1): 88–103.

Eisenstein, Jacob, Amr Ahmed & Eric P Xing. 2011. Sparse Additive Generative Models of Text. In *Proceedings of ICML*. pp. 1041–1048.

Ellis, Christopher & James A Stimson. 2012. *Ideology in America*. Cambridge University Press.

Erikson, Robert S, Michael B MacKuen & James A Stimson. 2002. *The Macro Polity*. Cambridge University Press.

Eshbaugh-Soha, Matthew. 2010. "The Tone of Local Presidential News Coverage." *Political Communication* 27(2): 121–140.

Farmer, Brian. 2006. *American Political Ideologies*. McFarland.

Firth, John R. 1957. *A Synopsis of Linguistic Theory, 1930-1955*. Blackwell.

Freeden, Michael. 2003. *Ideology: A Very Short Introduction*. Oxford University Press.

Geer, John G. 2008. *In Defense of Negativity: Attack Ads in Presidential Campaigns*. University of Chicago Press.

Gerring, John. 1997. "Ideology: A Definitional Analysis." *Political Research Quarterly* 50(4): 957–994.

Giusti, Alessandro, Dan C Cireşan, Jonathan Masci, Luca M Gambardella & Jürgen Schmidhuber. 2013. "Fast Image Scanning With Deep Max-Pooling Convolutional Neural Networks." *arXiv Preprint arXiv:1302.1700* .

Glorot, Xavier & Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *International Conference on Artificial Intelligence and Statistics*. pp. 249–256.

Goh, ATC. 1995. "Back-Propagation Neural Networks for Modeling Complex Systems." *Artificial Intelligence in Engineering* 9(3): 143–151.

Goldberg, Yoav & Omer Levy. 2014. "Word2vec Explained: Deriving Mikolov Et al.'s Negative-Sampling Word-Embedding Method." *arXiv Preprint arXiv:1402.3722* .

Gori, Marco & Alberto Tesi. 1992. "On the Problem of Local Minima in Backpropagation." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 14(1): 76–86.

Graham, Benjamin. 2014. "Fractional Max-Pooling." *arXiv Preprint arXiv:1412.6071* .

Grimmer, Justin & Brandon M Stewart. 2013. "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* 21(3): 267–297.

Hamilton, Malcolm B. 1987. "The Elements of the Concept of Ideology." *Political Studies* 35(1): 18–38.

Harris, Zellig S. 1954. "Distributional Structure." *Word* 10(2-3): 146–162.

Hart, Roderick P, Jay P Childers & Colene J Lind. 2013. *Political Tone: How Leaders Talk and Why.* University of Chicago Press.

Hearst, Marti A., Susan T Dumais, Edgar Osman, John Platt & Bernhard Scholkopf. 1998. "Support Vector Machines." *Intelligent Systems and Their Applications, IEEE* 13(4): 18–28.

Hecht-Nielsen, Robert. 1989. Theory of the Backpropagation Neural Network. In *Neural Networks, 1989. IJCNN., International Joint Conference On.* IEEE pp. 593–605.

Hinich, Melvin J & Michael C Munger. 1996. *Ideology and the Theory of Political Choice.* University of Michigan Press.

Hinton, Geoffrey E, Simon Osindero & Yee-Whye Teh. 2006. "A Fast Learning Algorithm for Deep Belief Nets." *Neural Computation* 18(7): 1527–1554.

Holm, John D & John P Robinson. 1978. "Ideological Identification and the American Voter." *Public Opinion Quarterly* 42(2): 235–246.

Hopkins, Daniel & Gary King. 2007. "Extracting Systematic Social Science Meaning From Text." *Manuscript Available at Http://gking. Harvard. Edu/files/words. Pdf* .

Hopkins, Daniel J & Gary King. 2010. "A Method of Automated Nonparametric Content Analysis for Social Science." *American Journal of Political Science* 54(1): 229–247.

Hu, Baotian, Zhengdong Lu, Hang Li & Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems.* pp. 2042–2050.

Hurwitz, Mark S, Roger J Moiles & David W Rohde. 2001. "Distributive and Partisan Issues in Agriculture Policy in the 104th House." *American Political Science Review* pp. 911–922.

Iyyer, Mohit, Peter Enns, Jordan L Boyd-Graber & Philip Resnik. 2014. Political Ideology Detection Using Recursive Neural Networks. In *ACL (1).* pp. 1113–1122.

Jackman, Simon. 2001. "Multidimensional Analysis of Roll Call Data via Bayesian Simulation: Identification, Estimation, Inference, and Model Checking." *Political Analysis* 9(3): 227–241.

Jacoby, William G. 1991. "Ideological Identification and Issue Attitudes." *American Journal of Political Science* pp. 178–205.

Jost, John T. 2006. "The End of the End of ideology." *American Psychologist* 61(7): 651.

Julien, Isaac & Philip Resnik. N.d. "Aligning Real-Time Opinion Poll Responses With an Expectation-Maximization Algorithm.".

Kaid, Lynda Lee. 2004. "Political Advertising." *Handbook of Political Communication Research* pp. 155–202.

Kerlinger, Fred Nichols. 1984. *Liberalism and Conservatism: The Nature and Structure of Social Attitudes.* Vol. 1 Lawrence Erlbaum Assoc Incorporated.

Kim, Yoon. 2014. "Convolutional Neural Networks for Sentence Classification." *arXiv Preprint arXiv:1408.5882* .

Knight, Kathleen. 1990. "Ideology and Public Opinion." *Research in Micropolitics* 3: 59–82.

Krippendorff, Klaus. 2004. "Reliability in Content Analysis." *Human Communication Research* 30(3): 411–433.

Kritzer, Herbert M. 1978. "Ideology and American Political Elites." *Public Opinion Quarterly* 42(4): 484–502.

Lai, Siwei, Kang Liu, Liheng Xu & Jun Zhao. 2015. "How to Generate a Good Word Embedding?" *arXiv Preprint arXiv:1507.05523* .

Laver, Michael, Kenneth Benoit & John Garry. 2003. "Extracting Policy Positions From Political Texts Using Words as Data." *American Political Science Review* 97(02): 311–331.

Le Cun, B Boser, John S Denker, D Henderson, Richard E Howard, W Hubbard & Lawrence D Jackel. 1990. Handwritten Digit Recognition With a Back-Propagation Network. In *Advances in Neural Information Processing Systems.* Citeseer.

LeCun, Yann, Yoshua Bengio & Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521(7553): 436–444.

Lee, Frances E. 2009. *Beyond Ideology: Politics, Principles, and Partisanship in the US Senate.* University of Chicago Press.

Lengauer, Günther, Frank Esser & Rosa Berganza. 2011. "Negativity in Political News: A Review of Concepts, Operationalizations and Key Findings." *Journalism* p. 1464884911427800.

Leopold, Edda & Jörg Kindermann. 2002. "Text Categorization With Support Vector Machines. How to Represent Texts in Input Space?" *Machine Learning* 46(1-3): 423–444.

Levendusky, Matthew. 2009. *The Partisan Sort: How Liberals Became Democrats and Conservatives Became Republicans.* University of Chicago Press.

Levy, Omer & Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems.* pp. 2177–2185.

Lin, Min, Qiang Chen & Shuicheng Yan. 2013. "Network in Network." *arXiv Preprint arXiv:1312.4400* .

Liu, Bing. 2012. "Sentiment Analysis and Opinion Mining." *Synthesis Lectures on Human Language Technologies* 5(1): 1–167.

Loewenstein, Karl. 1953. "Political Systems, Ideologies, and Institutions: The Problem of Their Circulation." *The Western Political Quarterly* 6(4): 689–706.

Manning, Christopher D, Prabhakar Raghavan, Hinrich Schütze & Others. 2008. *Introduction to Information Retrieval.* Number 1 Cambridge University Press Cambridge.

Marcus, George E, David Tabb & John L Sullivan. 1974. "The Application of Individual Differences Scaling to the Measurement of Political Ideologies." *American Journal of Political Science* pp. 405–420.

Martin, Andrew D & Kevin M Quinn. 2002. "Dynamic Ideal Point Estimation via Markov Chain Monte Carlo for the US Supreme Court, 1953–1999." *Political Analysis* 10(2): 134–153.

Mayhew, David R. 1974. *Congress: The Electoral Connection.* Yale University Press.

McCarty, N.M., K.T. Poole & H. Rosenthal. 2006. *Polarized America: The Dance of Ideology and Unequal Riches.* MIT Press Cambridge, MA.

McClosky, Herbert & John Zaller. 1984. *The American Ethos: Public Attitudes Toward Capitalism and Democracy.* Harvard University Press.

Mikolov, Tomas, Kai Chen, Greg Corrado & Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space." *arXiv Preprint arXiv:1301.3781* .

Nagi, Jawad, Frederick Ducatelle, Gianni a Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber & Luca Maria Gambardella. 2011. Max-Pooling Convolutional Neural Networks for Vision-Based Hand Gesture Recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference On.* IEEE pp. 342–347.

Nigam, Kamal, John Lafferty & Andrew McCallum. 1999. Using Maximum Entropy for Text Classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering.* Vol. 1 pp. 61–67.

Noel, Hans. 2014. *Political Ideologies and Political Parties in America.* Cambridge University Press.

Pal, Sankar K & Sushmita Mitra. 1992. "Multilayer Perceptron, Fuzzy Sets, and Classification." *Neural Networks, IEEE Transactions On* 3(5): 683–697.

Pang, Bo, Lillian Lee & Shivakumar Vaithyanathan. 2002. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10.* Association for Computational Linguistics pp. 79–86.

Poole, Keith T & Howard L Rosenthal. 2011. *Ideology and Congress.* Vol. 1 Transaction Publishers.

Poole, Keith T & Howard Rosenthal. 1991. "Patterns of Congressional Voting." *American Journal of Political Science* pp. 228–278.

Poole, K.T. & H. Rosenthal. 1984. “The Polarization of American Politics.” *The Journal of Politics* 46(04): 1061–1079.

Poole, K.T. & H. Rosenthal. 1999. “D-Nominate After 10-Years: A Comparative Update to Congress: A Political Economic History of Roll Call Voting.” *Unpublished Manuscript. Carnegie Mellon University* .

Porter, Martin F. 2001. “Snowball: A Language for Stemming Algorithms.”.

Proksch, Sven-Oliver & Jonathan B Slapin. 2008. “Position Taking in European Parliament Speeches.” *American Journal of Political Science* 52(3): 705–722.

Putnam, Robert D. 1971. “Studying Elite Political Culture: The Case of ŞIdeologyŤ.” *American Political Science Review* 65(03): 651–681.

Ridout, Travis N & Michael Franz. 2008. “Evaluating Measures of Campaign Tone.” *Political Communication* 25(2): 158–179.

Roberts, Margaret E, Brandon M Stewart, Dustin Tingley, Edoardo M Airoldi & Others. 2013. The Structural Topic Model and Applied Social Science. In *Advances in Neural Information Processing Systems Workshop on Topic Models: Computation, Application, and Evaluation.*

Rokeach, Milton. 1968. “Beliefs, Attitudes and Values: A Theory of Organization and Change.”.

Rong, Xin. 2014. “Word2vec Parameter Learning Explained.” *arXiv Preprint arXiv:1411.2738* .

Schattschneider, E.E. 1960. *The Semisovereign People.* New York: Holt, Rinehart and Winston.

Schmidhuber, Jürgen. 2015. “Deep Learning in Neural Networks: An Overview.” *Neural Networks* 61: 85–117.

Severyn, Aliaksei & Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado.* pp. 464–469.

Sheafer, Tamir. 2007. “How to Evaluate It: The Role of Story-Evaluative Tone in Agenda Setting and Priming.” *Journal of Communication* 57(1): 21–39.

Shils, Edward Albert. 1968. *The Concept and Function of Ideology.* Crowell Collier and Macmillan.

Shor, Boris, Christopher Berry & Nolan McCarty. 2010. “A Bridge to Somewhere: Mapping State and Congressional Ideology on a Cross-Institutional Common Space.” *Legislative Studies Quarterly* 35(3): 417–448.

Shor, Boris & Nolan McCarty. 2011. “The Ideological Mapping of American Legislatures.” *American Political Science Review* 105(03): 530–551.

Sim, Yanchuan, Brice Acree, Justin H Gross & Noah A Smith. 2013. Measuring Ideological Proportions in Political Speeches. In *Empirical Methods in Natural Language Processing.*

Skoric, Marko, Nathaniel Poor, Palakorn Achananuparp, Ee-Peng Lim & Jing Jiang. 2012. Tweets and Votes: A Study of the 2011 Singapore General Election. In *System Science (HICSS), 2012 45th Hawaii International Conference On.* IEEE pp. 2583–2591.

Slapin, Jonathan B & Sven-Oliver Proksch. 2008. "A Scaling Model for Estimating Time-Series Party Positions From Texts." *American Journal of Political Science* 52(3): 705–722.

Soelistio, Yustinus Eko & Martinus Raditia Sigit Surendra. 2015. "Simple Text Mining for Sentiment Analysis of Political Figure Using Naïve Bayes Classifier Method." *arXiv Preprint arXiv:1508.05163* .

Somasundaran, Swapna & Janyce Wiebe. 2010. Recognizing Stances in Ideological On-Line Debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text.* Association for Computational Linguistics pp. 116–124.

Spirling, Arthur. 2011. "US Treaty-Making With American Indians." *American Journal of Political Science* .

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever & Ruslan Salakhutdinov. 2014. "Dropout: A Simple Way to Prevent Neural Networks From Overfitting." *The Journal of Machine Learning Research* 15(1): 1929–1958.

Steinwart, Ingo & Andreas Christmann. 2008. *Support Vector Machines.* Springer Science & Business Media.

Stimson, James A. 2004. *Tides of Consent: How Public Opinion Shapes American Politics.* Cambridge University Press.

Stimson, James a, Michael B MacKuen & Robert S Erikson. 1995. "Dynamic Representation." *American Political Science Review* 89(03): 543–565.

Tang, Duyu, Furu Wei, Nan Yang, Ming Zhou, Ting Liu & Bing Qin. 2014. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *ACL (1).* pp. 1555–1565.

Tedin, Kent L. 1987. "Political Ideology and the Vote." *Research in Micropolitics* 2(1): 63–94.

Thomas, Matt, Bo Pang & Lillian Lee. 2006. Get Out the Vote: Determining Support or Opposition From Congressional Floor-Debate Transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics pp. 327–335.

Thompson, John B. 1984. *Studies in the Theory of Ideology.* University of California Press.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 267–288.

Tu, Jack V. 1996. "Advantages and Disadvantages of Using Artificial Neural Networks Versus Logistic Regression for Predicting Medical Outcomes." *Journal of Clinical Epidemiology* 49(11): 1225–1231.

Tumasjan, Andranik, Timm Oliver Sprenger, Philipp G Sandner & Isabell M Welpe. 2010. "Predicting Elections With Twitter: What 140 Characters Reveal About Political Sentiment." *ICWSM* 10: 178–185.

Wang, Hongning, Duo Zhang & ChengXiang Zhai. 2011. Structural Topic Model for Latent Topical Structure Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.* Association for Computational Linguistics pp. 1526–1535.

Welch, Susan & Eric H Carlson. 1973. "The Impact of Party on Voting Behavior in a Nonpartisan Legislature." *American Political Science Review* 67(03): 854–867.

Werbos, Paul John. 1994. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting.* Vol. 1 John Wiley & Sons.

Wright, Gerald C & Brian F Schaffner. 2002. "The Influence of Party: Evidence From the State Legislatures." *American Political Science Review* 96(02): 367–379.

Xue, Bai, Chen Fu & Zhan Shaobin. 2014. A Study on Sentiment Computing and Classification of Sina Weibo With Word2vec. In *2014 IEEE International Congress on Big Data.* IEEE pp. 358–363.

Yu, Dong & Li Deng. 2011. "Deep Learning and Its Applications to Signal and Information Processing [Exploratory Dsp]." *Signal Processing Magazine, IEEE* 28(1): 145–154.

Zeiler, Matthew D & Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision.* Springer pp. 818–833.

Zhang, Guoqiang Peter. 2000. "Neural Networks for Classification: A Survey." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions On* 30(4): 451–462.

Zhang, Xiang, Junbo Zhao & Yann LeCun. 2015. Character-Level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems.* pp. 649–657.

Zhang, Xiang & Yann LeCun. 2015. "Text Understanding From Scratch." *arXiv Preprint arXiv:1502.01710* .

Zhang, Ye & Byron Wallace. 2015. "A Sensitivity Analysis of (And Practitioners' Guide To) Convolutional Neural Networks for Sentence Classification." *arXiv Preprint arXiv:1510.03820* .

Zhang, Ye, Stephen Roller & Byron Wallace. 2016. "Mgnc-Cnn: A Simple Approach to Exploiting Multiple Word Embeddings for Sentence Classification." *arXiv Preprint arXiv:1603.00968* .