Embedding Concepts and Documents in One Space

A System for Valid and Replicable Text-as-Data Measurement

Haohan Chen*

December 14, 2020

Abstract

In this paper, I develop a system for generation of valid and replicable measures from text data by integrating distributed semantics with unsupervised and semi-supervised clustering models and researchers' prior information. The system for text-as-data measurement has two design features: informative representation and stepwise guidance. With informative representation, it learns distributed semantics to represent words, concepts, and documents in a low-dimensional space. With stepwise guidance, it integrates distributed semantics with clustering algorithms, seed dictionaries, and researchers' manual coding into a workflow that reliably links concepts of substantive interest with messy text data and produces codebooks for convenient replication and extension.

"Trust me. I've checked. See these examples." This is the typical response to inquiries on text-as-data measurement validity and replicability in a political science conference room. No more questions.

In text-as-data studies, measurement validity and replicability puzzles researchers and their audience. Researchers who are willing to put in lots of effort improving the reliability of their text-based measures, either by fine-tuning machine reading models or by investing hours to read documents, lack tools to improve their efficiency. They also lack tools to help systematically document their efforts and results in human-interpretable ways to communicate with their audience. On the other end, audiences of text-as-data studies have a hard time asking concrete questions and making specific suggestions regarding measurement. Presented with

*Postdoctoral Fellow, Center for Social Media and Politics, New York University, 60 5th Avenue, New York, NY 10011 (Email: haohan.chen@nyu.edu)

only a general description of concepts of interest and methods applied, probably accompanied by a few example documents, it is almost impossible to concretely criticize a text-as-data measurement scheme.

Existing text-as-data methods do not equip researchers with enough tools to systematically improve validity of text-based measurement in the process of data analysis. Nor is there a standardized way to output replicable schemes to create measurement from text data.

In this paper, I develop a text-as-data system for valid and replicable measurement. The system is built on on distributed semantics, an important family of natural language processing techniques that have just started to receive attention in the political science community. I provide the first comprehensive introduction of the method to the political science community, including its intuition and engineering details. I also develop an original system helping political scientists create text-based measures using distributed semantics.

The rest of this paper is organized as follows: Section 1 identifies the problem of measurement validity and replicability in text-as-data methods with reference to the literature and outlines my solution. Section 2 introduces the intuition and engineering details of distributed semantics. Section 3 presents the distributed codebook approach, a workflow helping researchers interpreting and using distributed semantics for document coding. Section 4 introduces All-Text-In-One-Space (ATIOS), a new system that creates valid and replicable measurement from text data. Section 5 concludes and discusses contributions.

1 Validity and Replicability of Text-as-Data Measurement

Political scientists applying text-as-data methods in their research are often asked two questions: How valid a measure is a text-based variable? How replicable is the study? A text-as-data study producing valid measurement convincingly shows its audience how raw text data, usually large and messy, are mapped to simplified and quantified indicators of concepts and ideas. A replicable text-as-data study transparently reports its analysis process and data that enables reproduction, replication, and extension. Many existing text-as-data methods face limitations in both interpretability and replicability. In this section, I identify the research question with reference to the existing literature. The first part reviews the development of text-as-data methods in political science. The second part discusses how state-of-the-art

methods face limitations in measurement validity and replicability. The third part outlines my approach to tackling the challenges.

1.1 Text-as-Data in Political Science: A Brief Review

Text-as-data methods emerged in political science studies and other social sciences fields, empowered by recent development of machine learning and natural language processing techniques. Text-as-data methods assist political scientists in studies of political activities using language as the medium. Initial applications include using text of treaties to study peacemaking and using text of laws and regulations to study legislation and ideology (see Grimmer and Stewart, 2013, for a review). Other fields of social sciences embrace the methods. For example, economists use text data to study stock market volatility, impact of central bank policies, and economic policy uncertainty (see Gentzkow et al., 2017, for a review). The application of text-as-data methods is made possible by the recent rapid development of natural language processing techniques in computer science, featuring crowdsourced large training data, increased computational power, and integration with machine learning (especially deep neural network) algorithms (see Goldberg, 2016, for a review of deep neural network for natural language processing) .

I classify text-as-data methods into two types, distinguished by the relative role of machines and humans in making sense of text data. In the first type, machines "read" and categorize text into groups with minimal human input (except for some model parameters and random seeds). Humans interpret and validate the groups produced by machines, linking them to concepts and ideas of interest. The most widely used family of models in this category is topic modeling with Latent Dirichlet Allocation (LDA). A hierarchical Bayesian model, the LDA model represents documents as mixtures over latent topics, each of which is characterized by a distribution over words (Blei et al., 2003). Extending upon LDA, Structural Topic Modeling (STM) allows correlations among topics and inclusion of covariates. It has become an important tool of this type of text-as-data studies in political science (Roberts et al., 2014, 2016). For example, Pan and Chen (2018) apply STM to leaked email text data from communication within a Chinese propaganda department to analyze selective reporting of citizens' complaints; Kim (2018) uses STM to code text of Chinese news reports on automobiles to

detect and explain media bias against foreign automakers.

In the second type of text-as-data methods, humans "teach" machines how to make sense of text data. They generally take the following steps: humans code a subset of text data in light of the ideas and concepts of interest; supervised machine learning algorithms are trained to " imitate" human coding; the best algorithms are selected using a standard set of evaluation metrics; in the end, the best algorithms are used to code the rest of the text data, and researchers validate and interpret the results. This type of text-as-data methods, broadly defined, has a long history in political science research. The dictionary approach, where researchers construct a dictionary mapping words and phrases to concepts and task machines to look them up in text data, is the simplest algorithm in this type of methods. Among many applications, political scientists use the approach to classify documents by their tone and mention of specific topics of interest (see Grimmer and Stewart, 2013, for a review). Simple as it appears, the dictionary approach can be powerful, even when the data are large and messy. For example, a recent application in studying censorship in authoritarian China, King et al. (2013) use a dictionary approach to identify and scrape millions of politically relevant social media posts in China's social media sites. Moving beyond the dictionary approach, researchers employ more complex machine learning algorithms for natural language processing to detect more complex features from text. Pioneering in the methodological innovation in this type of methods, King and Hopkins (2010) develop a system to estimate proportions of documents in categories of interest based on a subset of human-coded documents. King et al. (2017) apply a combination of methods to code social media posts written by government-sponsored commentators in China. Recent efforts improve the efficiency of "teaching." Miller et al. (2018) introduces an active learning approach which enables machines to learn a satisfactory coding algorithm with less human labelling effort by iteratively prompting humans to code "borderline" cases where the current algorithm is indecisive about coding.

Though categorization of methods is not the focus of this paper and this section, I would like to add a note about the rationale of my categorization scheme, which is different from the existing literature. I refrain from using the supervised/unsupervised learning dichotomy to classify text-as-data methods in social sciences. For me, a text-as-data method is a workflow. The type of machine learning model applied only constitutes a part of a workflow. Equally crucial and worth explicit characterization is the role of humans in the workflow, because

interpreting the results and making inferences is the goal of all the political science studies reviewed. This differs from the literature of machine learning and natural language processing: their tasks are standardized, goals are mainly prediction, and the human's role is usually peripheral.

I highlight the above methods and their application as this paper directly builds on them with a focus on measurement (detailed in the following sections). However, the research frontier of text-as-data includes a separate stream of work, to which this paper does not directly speak but has potential contribution: text-as-data for causal inference. Egami et al. (2018) develop a workflow where researchers split data into training and test set to avoid analyst induced SUTVA violation caused by tuning and interpreting text-as-data models. Roberts et al. (2018) develop Topical Inverse Regression Matching that uses results of STM to match documents for causal inference with observational text data. Mozer et al. (2019) develop an experimental study where human respondents evaluate uses of a variety of text feature representation and text distance metrics in terms of their quality serving as confounder for matching. In general, these studies treat text as a kind of high-dimensional data, use text-as-data methods primarily for dimensional reduction (as oppose to measurement), and focus on discovery of a dimensional reduction scheme that produces unbiased causal estimates.

1.2 Challenges: Valid and Replicable Measurement with Text Data

Political scientists applying text-as-data methods face two challenging tasks: to justify that variables generated from text correctly map to concepts and ideas of interest; and to discuss to what extent the findings hold in new data. The former is an inquiry about measurement validity, and the latter is about replicability. In this section, I discuss how existing works have addressed the challenges and what remains to be done.

To justify that variables generated from text correctly map to concepts and ideas of interest, researchers should be able to control and describe the intermediate steps of the "trip" from raw text to concepts. Existing text-as-data methods constrain researchers' capacity to do so. Existing literature has identified the challenge and partly addressed it by proposing methods for validation. A comprehensive review of text-as-data methods, Grimmer and Stewart (2013) suggest four principles for text-as-data application, emphasizing that ma-

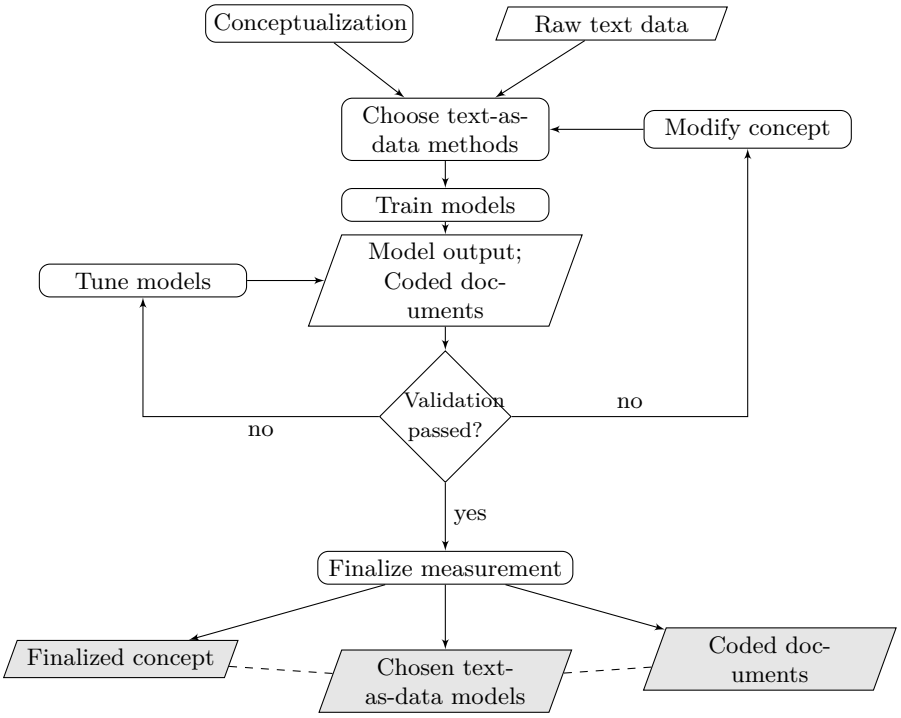
chines cannot replace researchers' careful reading and that researchers' extensive validation of machine coding is crucial ("validate, validate, validate").[1] Methods for validation have been developed in the computer science community, which political scientists have applied and extended in recent literature. Chang et al. (2009), an initial work on systematically validating topic modeling, conduct experiments that show misalignment between human coders' and machine judgment about the best mapping among words, topics, and documents.[2] The type of validation methods has recently been extended with political text data. Ying et al. (2019) develop a generic tool to validate political text labeling produced by topic modeling and researchers' interpretation of topics. Spirling and Rodriguez (2019) design a "Turing test" to evaluate the quality of word embedding trained with different parameter setups and data sources.

While the new validation methods can enhance the audience's confidence in correct mapping between text and concepts, they only address part of the challenges in justifying measurement validity: they at best show that the measurement is correct end-to-end, but they cannot show how complex intermediate steps, often containing back-and-forth adjustment, are conducted. To elaborate the point, I outline typical intermediate steps of text-as-data research with reference to an older literature on content analysis and measurement validity (Neuendorf, 2002, Chapter 3; Adcock and Collier, 2001). As Figure 1 shows, text-as-data studies usually start with a set of text data and some initial conceptualization about what content in the text can be measured with simple and quantified variables. Based on the two inputs, researchers choose appropriate text-as-data methods (e.g., methods reviewed in Section 1.1). Models are trained with various types of human input. With output of models (e.g., fitted parameters and coefficients) and documents coded by the models, researchers validate the results by judging if they make intuitive sense. Satisfactory results almost never appear within a single attempt. Researchers adjust the analyses in two ways. First, they may decide the conceptualization and the models need tuning. For example, random seeds and numbers of topics in topic modeling may be changed; types and parameters of super-

[1] (See also Wilkerson and Casas, 2017, a more recent review article pointing out instability of text-as-data models)

[2] There have been numerous extensions upon this work in the computer science community. See, for example: Mimno et al. (2011); Lau et al. (2015). See also Blei (2012) for a review on how topic interpretation and validation has become a research frontier of topic modeling.

Figure 1: A General Workflow of Text-as-Data Studies



vised machine learning may be changed. Second, researchers may find an alternative way to conceptualize the variable of interest. They may reconsider choice of models and re-train models accordingly. The steps are repeated until the models' output and coded documents pass researchers' validation, which allows researchers to finalize their measurement. This is a long and complex process, which goes undocumented in most applications of text-as-data methods. Even with the latest development of validation methods (e.g., Ying et al., 2019), researchers still face challenges convincing their audience that this iterative process of adjustment is conducted correctly. The problem is worse when the dataset is large and when models are complex. Given the state-of-the-art, I argue that an important avenue for contributions in text-as-data methods is development of systems that standardize and clearly document the iterative process in its intermediate steps.

A related but different challenge with text-as-data measurement is replicability. The challenge of replication in text-as-data research has unique features distinguished from other types of political science empirical studies, which has not been extensively discussed. The call for replication is a symbolic feature of modern scientific inquiry in political science. To produce

replicable work, researchers are advised to publicize the exact process of data generation and analysis for evaluation (King, 1995). I argue that a unique replicability challenge for text-as-data measurement is to publicize a measurement model containing enough information to help produce the same kind of measurement with new data. Note that this is an issue related but different from the lack of documented iterative steps discussed above, whose crux is the process being untraceable and unexplainable. The crux of the replicability challenge is the output of the measurement exercise being uninterpretable and unusable by researchers trying to follow up. As is shown in the bottom row of Figure 1 (the three shaded nodes), existing text-as-data studies publicize three types of information for replication: finalized concepts, chosen text-as-data models, and coded documents. First, researchers describe the concept measured by text data (e.g., incivility of a social media post; ideological leaning or issue area of a legislative bill). Second, researchers describe the chosen text-as-data models used. Studies describe the models with different levels of specificity. Most political science studies briefly describe the models' architecture and the softwares applied, while only a few share detailed parameters of the trained models. Third, researchers share the coded documents. Due to constraints such as data protection, a significant proportion of studies do not share raw text. A table mapping document ID to coding is the usual form of output. With these types of output, it is challenging to replicate studies with text-as-data measurement, primarily because the link between concepts and coded documents is usually neither interpretable nor generalizable to other data. As shown in Figure 1, finalized concepts and coded documents are connected through chosen text-as-data methods. The models can be complex and unstable. Thus it may not be conveniently interpreted on what explain the mapping.[3] Also, models trained with a specific dataset may not generalize to other datasets, blocking researchers' way to directly apply the original paper's trained text-as-data models (in rare cases that are shared) to their new data.

[3]Note that political methodologists are making efforts to make complex models interpretable. See Roberts et al. (2014) for the STM R package which provides a user-friendly interface to fit and visualize topic models. See also Sanford et al. (2019) for a recent contribution to interpreting deep neural network models for political text.

1.3 My Approach: Stepwise Guidance and Informative Representation

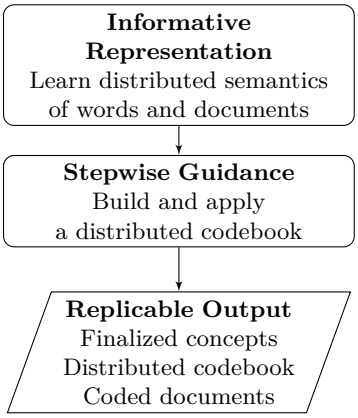
I suggest that two design features of text-as-data systems facilitate production of valid and replicable measurement. First, when the system can take human "guidance" in a stepwise manner (as opposed to only allowing validation after the final step of a long process), researchers gain control and understanding of how intermediate steps are systematically done right. Second, when the system's final output contains informative representation of concepts, documents, and their relation, readers can conveniently transfer the measurement scheme to new data. The former increases measurement validity. The latter increases replicability.

To build a text-as-data system that takes stepwise "guidance" from human, I divide it into modules, with each module containing an "encoder" and a "decoder" of intermediate output. First, I modularize the system, with each module processing a part of the measurement task. Each module (which usually contain a machine coding algorithm) is simple and stable enough for researchers to validate, intervene, and confidently choose an optimal solution for a part of the measurement task. Second, each module contains an "encoder" to transform human-readable text to machine-readable numeric representation and a "decoder" that transform numeric model output back to human-readable content to facilitate researchers' validation and intervention.

To build a text-as-data system producing informative output, I represent concepts, documents, and their relationship with rich numeric values transferable across text datasets. A common practice in state-of-the-art content analysis, researchers simplify results into verbal descriptions of finalized concepts and coding of documents. This loses a large amount of information that would have been valuable for replication. I use distributed semantics, a family of text representation models, to create a new type of interpretable and replicable output for text-as-data studies in political science.

In the following sections, I present a new text-as-data system that integrates the two design features. The system, namely All-Text-In-One-Space (ATIOS), introduces a new way for political scientists to incorporate distributed semantics, one of the best innovations in natural language processing in recent years, in their workflow of text-as-data measurement. Figure 2 presents an overview of the system. ATIOS contains two major steps, to be elaborated in detail in Section 2 and 3. The first step obtains informative representation of text by

Figure 2: All Text In One Space (Simplified workflow)



applying distributed semantics, a new natural language processing algorithm representing text with low-dimensional dense vectors. The second step, building on distributed semantics, combines machine learning algorithms and human evaluation to build and apply a distributed codebook to code documents. The system outputs a set of information that forms valid and replicable text-as-data measurement: description of concepts of interest, coded documents, and a distributed codebook that not only precisely maps concepts to documents but is also generalizable for replication.

2 Informative Representation: Distributed Semantics

Core to my system for valid and replicable text-as-data measurement, distributed semantics is a type of methods to create text representation that quantifies and categorizes their meanings. Recent innovation in natural language processing finds efficient algorithms to learn meanings of words, sentences, and documents from large text datasets and represent them as low-dimensional dense numeric vectors. The methods have proven powerful in increasing predictive accuracy in many standard natural language processing tasks. In addition, evidence shows that the representation uncovers linguistic regularity. Despite its promise, the method needs adaptation to fit the needs of text-as-data studies in political science. In this section, I introduce the methods of distributed semantics. The first part presents an overview of distributed semantics. The second part introduces how distributed semantics are generated in one of its most popular algorithms, *word2vec*. The third part discusses why its existing ap-

plications are incompatible with my goal, valid and replicable measurement and then outlines how I build a new system to serve my goal.

2.1 Distributed Semantics: An Overview

Distributed semantics is a new method based on an old idea: it is based on an idea that linguists proposed over six decades ago; computer scientists have built on it to make one of the best innovations for automated natural language processing in the past decade.

Distributed semantics is based on a simple linguistic idea called the distributional hypothesis: words that occur in similar contexts have similar meanings (see, for example, Joos, 1950; reviewed by **?**, Chapter 6). This idea has the power to categorize both known and unknown words. To elaborate, I use a simple example. Suppose we are presented the following four sentences: "Beijing is the capital of China." "Tokyo is the capital of Japan." "Berlin is the capital of Germany." "Paris is the capital of France." Common sense tells us that China, Japan, Germany, and France are names of countries and Beijing, Tokyo, Berlin, and Paris are names of their capital cities respectively. However, assume a reader does not have this common sense, he can still apply the distributional hypothesis to categorize words and infer their relationship: Beijing, Tokyo, Berlin, and Paris belong to one category as they all appear before "is the capital of" ; China, Japan, Germany, and Paris belong to another category as they all appear after "is the capital of" ; The former has some relationship with the latter, moderated by the phrase "is the capital of." Moreover, a reader can use categorization learned with the distributional hypothesis to understand new words. Assume a reader is presented with a fifth sentence "Ngerulmud is the capital of Palau." If he is unfamiliar with countries in the Pacific Ocean, he can recognize that Palau is a country and Ngerulmud is its capital city. Even if he has no common sense in geography, he can at least understand: Ngerulmud belongs to the same category as Beijing, Tokyo, Paris, and Berlin; Palau belongs to the same category as China, Japan, Germany, and Paris; the two have the same relationship moderated by "is the capital of."

The above example also helps demonstrate three intuitions of the distributional hypothesis, which transfer to distributed semantics. First, contexts are local. The text in my example is short, making it reasonable to use all other words as context to infer a word's meaning.

When a piece of text is long (common in most real-world text data), it should be divided into multiple context windows. Second, the distributional hypothesis helps learn relative relationships between words. But it cannot make sense of their absolute meanings. In the above example, a reader's lack of common sense in geography can use distributive hypothesis to infer that Beijing, Tokyo, Paris, and Berlin, and Ngerulmud belong to the same category. He cannot know that they are capital cities without additional information. Third, learning with distributional hypothesis need a lot of text. The above example sentences are simple and clean. And I started with a clear mind which words are of interest (e.g. "is," "the" are not words of interest). This enables inferences with only four sentences. Real-word text data are messier (e.g. large vocabulary, variety of ways to express the same meaning) and researchers start with few assumptions about what words are of interest. As a result, it requires a lot of text to learn meanings of text with the distributional hypothesis.

Algorithms of distributed semantics perform the above exercise with standardized procedures and quantified output (elaborated in Section 2.2). Recent computer science research builds on the distributional hypothesis to create vector representation of words that reflects their meanings. Prior to the prominence of distributed semantics (as in most text-as-data methods reviewed in Section 1.1), words have atomic representation in automated natural language processing systems: A word is represented by a numeric index that has no relation to its meaning. A document is represented by a list of indices of words appearing in it. In contrast, distributed semantics represents words with vectors of numeric values that best "describes" in what context it appears, which in turn represents its meaning. Note that a single dimension of this vector has no interpretable meaning: it doesn't represent a topic, intensity of a type of sentiment, or count of a linguistic element. However, a word vector represents its meanings as evident in multiple interpretable ways. First, words with similar syntactic or topic meaning have word vectors numerically close to one another. For example: $\mathbf{v}_{\text{Beijing}} \approx \mathbf{v}_{\text{Tokyo}} \approx \mathbf{v}_{\text{Paris}} \approx \mathbf{v}_{\text{Berlin}}$.[4] Second, analogy of words can be captured by simple linear operations between word vectors (Mikolov et al., 2013). For example: $\mathbf{v}_{\text{China}} - \mathbf{v}_{\text{Beijing}} \approx \mathbf{v}_{\text{Japan}} - \mathbf{v}_{\text{Tokyo}} \approx \mathbf{v}_{\text{Germany}} - \mathbf{v}_{\text{Berlin}}$. Third, adding word vectors results in meaningful representation of larger text units: Adding vectors of words in a phrase results in a valid representation of the phrase [an example in Mikolov et al., 2013,

[4] $\mathbf{v}_w$ denotes the word vector of a word w.

$\mathbf{v}_{\text{Russian}} + \mathbf{v}_{\text{river}} \approx \mathbf{v}_{\text{Volga}} + \mathbf{v}_{\text{river}}$]; A weighted sum of vectors of words in a document makes a valid representation of the document (Iyyer et al., 2015; Mitchell and Lapata, 2010; Arora et al., 2017).

2.2 Learning Distributed Semantics from Data

How do machines learn distributed semantics from text data? In this part, I introduce how word vectors are trained. First, I give an overview of the rationale. Second, I introduce technical details of two foundational and most popular algorithms for distributed semantics: *word2vec* by Mikolov et al. (2013) and *GloVe* by Pennington et al. (2014). Third, I discuss the differences among variants of the type of algorithms, how they are applied, and what is to my interest in my system.[5]

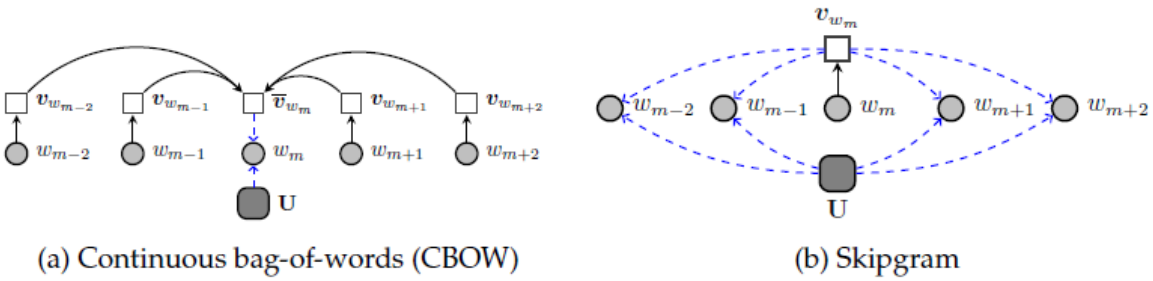
Machines assign words with numeric vectors so that mathematical operations among these vectors can best explain "which words likely appear with which words in what context", given the text data. Differences among algorithms map into different parts of this non-technical description. First, the operationalization of "which words likely appear with which words" is the major difference among two types of algorithms. "Prediction-based" algorithms operationalize it into a prediction problem: the machine passes through all the text, divides it into local context windows, and learns word vectors to have words best predict the existence of one another within each window. On the other hand, "cooccurrence-statistics-based" algorithms first construct a matrix with statistics for how often each pair of words co-occurs and then learn word vectors that best approximately reconstruct the matrix. Second, what information counts towards "context" varies among algorithms and implementations. Most conventional algorithms use word vectors as context, though the size of the context window (i.e., numbers of words before and after the target word) are subject to users' discretion. In recent development, algorithms incorporate other semantic information of words, sentences, and paragraphs as context.

[5]To my knowledge, this is the first systematic introduction of the methods' technical details to the political science community. While this is an original introduction, it refers to a variety of sources including textbooks, review papers, and code (as the original papers introducing the methods do not provide enough engineering details). This part is written with the aim to introduce the models' intuition and engineering techniques without going into excessive details. Major references are listed below. Textbooks: Eisenstein, 2019; Jurafsky and Martin, 2019. Review articles: Rong (2014); Goldberg and Levy (2014); Goldberg (2016). Original papers: Mikolov et al. (2013,?); Pennington et al. (2014).

2.2.1 *word2vec*

word2vec is a foundational prediction-based algorithm for distributive semantics. It has two variants: the Skip-gram model and the Continuous Bag-of-Words (CBOW) model. The difference between the two lies at the specific task of prediction performed. As shown in Figure 3, the Skip-gram model (on the right) predicts context words with the target word; the CBOW model (on the left) predicts the target word with context words. Thus, the definition of likelihood functions slightly differs, leading to different optimization tasks.

Figure 3: Two Variants of word2vec: Skip-gram and CBOW



(a) Continuous bag-of-words (CBOW) (b) Skipgram

Source: Eisenstein (2019, Chapter 14)

To formalize, I start with the following setup: Consider a sequence of text of T words in total. Let the size of context windows be c (i.e., c's immediate neighbors before and after a word are considered its context). Let $\mathbf{v}(w)$ be the vector of distributed semantics of word w. Let the size of the vocabulary be V, (i.e., there are V unique words in the text). Let $p(w_i|w_j)$ be the probability of word w_i appearing, given word w_j. Let $\mathcal{L}$ be the likelihood.

The *word2vec* uses the a softmax function to link distributed representation of words (word vectors) with their predicted probabilities. Specifically, the probability of word w_i given word w_j in its context window is the exponential of the dot products of the word vectors $\mathbf{v}_{w_i}, \tilde{\mathbf{v}}_{w_j}$ over the sum of the exponentials of the dot products of $\tilde{\mathbf{v}}_{w_j}$ with word vectors of all words in the vocabulary:

$$\log p(w_i|w_j) = \log \frac{\exp(\mathbf{v}_{w_i}^T \tilde{\mathbf{v}}_{w_j})}{\sum_{k=1}^{V} \exp(\mathbf{v}_{w_k}^T \tilde{\mathbf{v}}_{w_j})} \tag{1}$$

$$= \mathbf{v}_{w_i}^T \tilde{\mathbf{v}}_{w_j} - \log \sum_{k=1}^{V} \exp(\mathbf{v}_{w_k}^T \tilde{\mathbf{v}}_{w_j}) \tag{2}$$

Thus, the log-likelihood of the Skip-gram model is computed as follows. At location t of the text sequence, the joint conditional probability of words in the context window (conditional on the target word at t) is calculated. The conditional probabilities are obtained by applications of softmax on the target word vector against each context word vector. Then the algorithm moves to location $t+1$ and repeat the process until the end of the sequence. The log-likelihood is the sum of all log probabilities. Formally:

$$\log \mathcal{L} = \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \tag{3}$$

$$= \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log \frac{\exp(\mathbf{v}_{w_{t+j}}^T \tilde{\mathbf{v}}_{w_t})}{\sum_{k=1}^{V} \exp(\mathbf{v}_{w_k}^T \tilde{\mathbf{v}}_{w_t})} \tag{4}$$

$$= \sum_{t=1}^{T} \left[\sum_{-c \leq j \leq c, j \neq 0} \mathbf{v}_{w_{t+j}}^T \tilde{\mathbf{v}}_{w_t} - \log \sum_{k=1}^{V} \exp(\mathbf{v}_{w_k}^T \tilde{\mathbf{v}}_{w_t}) \right] \tag{5}$$

Similarly, the log-likelihood of the CBOW model is computed as follows: at location t of the text sequence, the probability of target word given context words is calculated. The conditional probability is obtained by a softmax of the target word vector and the average of context word vectors. Then the algorithm moves to location $t + 1$ and repeats the process until the end of the sequence. The log-likelihood is the sum of all log probabilities. Formally:

$$\log \mathcal{L} = \sum_{t=1}^{T} \log p(w_t | w_{t-c}, w_{t-c+1}, ...w_{t+c-1}, w_{t+c}) \tag{6}$$

$$= \sum_{t=1}^{T} \log \frac{\exp(\mathbf{v}_{w_t}^T \bar{\mathbf{v}}_t)}{\sum_{k=1}^{V} \exp(\mathbf{v}_{w_k}^T \bar{\mathbf{v}}_t)} \tag{7}$$

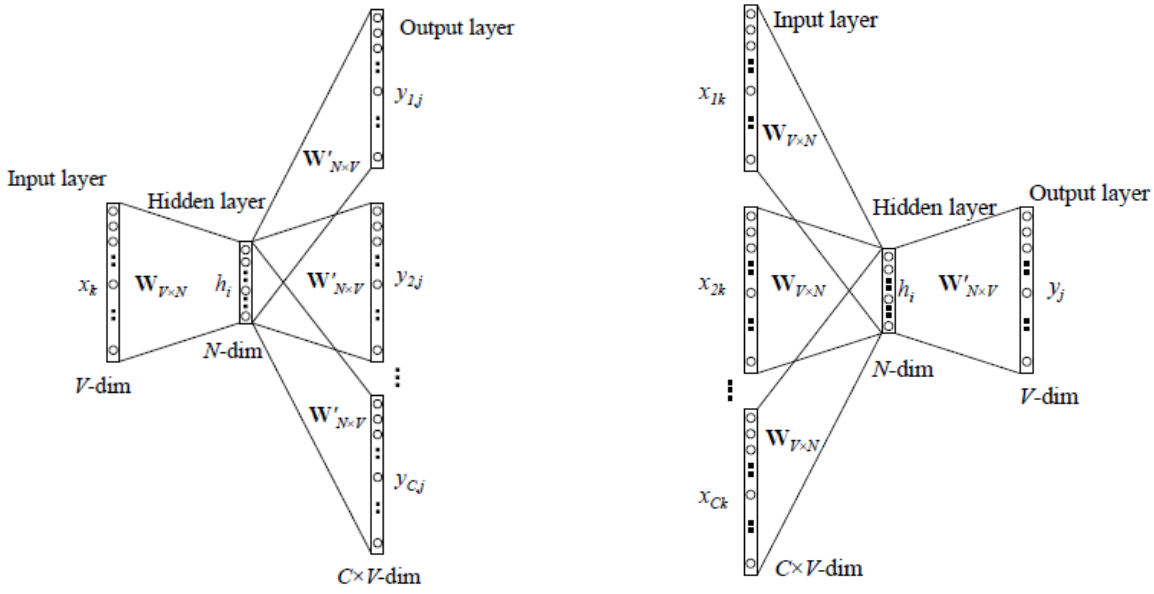
$$= \sum_{t=1}^{T} \left[\mathbf{v}_{w_t}^T \bar{\mathbf{v}}_t - \log \sum_{k=1}^{V} \exp(\mathbf{v}_{w_k}^T \bar{\mathbf{v}}_t) \right] \tag{8}$$

$$\text{where } \bar{v}_t = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} \tilde{\mathbf{v}}_{w_{t+j}} \tag{9}$$

Both Skip-gram and CBOW models train vector representations of words to maximize the above defined likelihood. The processing is operationalized as neural networks trained by stochastic gradient descent. Figure 4 shows the models‘ architecture. In general, they are both neural networks with one hidden layer and two weight matrices. The first weight

matrices $\mathbf{W}_{V\times N}$ contain vector representations of all V words as targets in the vocabulary: $\mathbf{W}_{V\times N} = [\mathbf{v}_{w_1}, \mathbf{v}_{w_2}, ..., \mathbf{v}_{w_N}]^T$. The second weight matrices $\tilde{\mathbf{W}}_{N\times V}$ contain vectors of words as context: $\tilde{\mathbf{W}}_{N\times V} = [\tilde{\mathbf{v}}_{w_1}, \tilde{\mathbf{v}}_{w_2}, ..., \tilde{\mathbf{v}}_{w_N}]$. Input and output layers are one-hot-encoded words. The differences between Skip-gram and CBOW are evident in the model architectures. Skip-gram (Panel a) uses target words to predict context words, while CBOW (Panel b) uses context words to predict target words. Word vectors are updated with stochastic gradient descent. For final output, researchers can use either of the two weight matrices $\mathbf{W}_{V\times N}, \tilde{\mathbf{W}}^T_{N\times V}$ or the two matrices' average as words' representation of distributed semantics.

Figure 4: The Neural Networks of word2vec Models



(a) Skip-gram (b) CBOW

Note: The figure demonstrates how algorithms Skip-gram and CBOW fit the text data in a neural network. This visualization is created by Rong (2014).

Training *word2vec* models can be computationally taxing. Two methods are used to reduce the computational demands of the model: hierarchical softmax and negative sampling. The algorithm in its naïve version described above can be computationally taxing primarily because the complexity of the softmax step (Equation 1) grows linearly with the vocabulary size (i.e., $O(V)$ complexity): in the forward pass, it takes summations over the whole vocabulary of size V for the denominator; in the backpropagation, it updates all V word vectors in the vocabulary. Two methods have been developed to boost efficiency. First, hierarchical softmax

uses a binary tree where words are represented by their leaf units. The probability of a word being the output is estimated by the probability of the path from root to leaf of the word. The method reduces computational complexity from $O(V)$ to $O(\log_2 V)$ given its tree structure. A second and more intuitive method, negative sampling, takes a random sample of words from the vocabulary to approximate the denominator in the forward pass and to update only the sample in the backpropagation. Thus, the computational complexity depends on the size of the negative sample and does not grow with the vocabulary size. The two methods have both demonstrated good performance in existing applications.

2.2.2 *GloVe*

The *GloVe* model is the most popular co-occurrence-statistics-based algorithm to learn distributed semantics of words. The most important distinction between this model and *word2vec* is that it operationalizes context data in a different way. Recall that *word2vec* walks through the whole sequence of text to predict. *GloVe* walks through the whole text sequence to count: the model starts by constructing a matrix of co-occurrence statistics, indicating the number of times each pair of words in the dictionary co-appear in the same context window. As it naturally follows, the objective of the model is to best predict this matrix of co-occurrence counts. Formally, let $\log X_{w_i,w_j}$ be the logarithm of the actually number of times word w_i and w_j co-appear in the same context; let $\log \hat{X}_{w_i,w_j}$ be the predicted logarithm of number of times words w_i and w_j co-appear. The square loss of predictive performance for a word pair w_i, w_j is:

$$(\log \hat{X}_{w_i,w_j} - \log X_{w_i,w_j})^2 \tag{10}$$

With the above square loss, a crucial step is to link vectors of distributed semantics to the count statistics and the above objective function. A key innovation of the *GloVe* model, the predicted logarithm of two words ‘co-appearance count, is modeled as the multiplication of their word vectors plus two bias terms. Formally:

$$\log \hat{X}_{w_i,w_j} = \mathbf{v}_{w_i}^T \tilde{\mathbf{v}}_{w_j} + b_{w_i} + \tilde{b}_{w_j} \tag{11}$$

$\mathbf{v}_{w_i}$ is the word vector of target word w_i; $\tilde{\mathbf{v}}_{w_j}$ is the word vector of context word w_j. $b_{w_i}, \tilde{b}_{w_j}$ are two bias terms not used in the final output. Note that the above link function is chosen for exchange symmetry: in this model, w_i being the context of w_j is equivalent to w_j being the context of w_i. [6] [7]

The final step to set up the objective function is to aggregate square losses of all pairs with some weighting scheme. The design of this final step is considered the decisive factor of the success a co-occurrence-statistics-based algorithm. In general, frequent co-occurrences should be weighed up, but not too much so. In one extreme, rare and frequent word pairs are assigned the same weight, then rare cooccurrences which contain lots of noise can harm the performance. At another extreme, if pairs are weighted by their occurrences, then frequent words dominate the training. The *GloVe* model weighs the cooccurrence statistics by count with some smoothing. Specifically, Let $f(X_{w_i,w_j})$ be the weight assigned to the pair of words w_i, w_j, the objective function of *GloVe* is as follows:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{w_i,w_j}) \left[\mathbf{v}_{w_i}^T \tilde{\mathbf{v}}_{w_j} + b_{w_i} + \tilde{b}_{w_j} - \log X_{w_i,w_j} \right]^2 \tag{12}$$

$$f(X_{w_i,w_j}) = \begin{cases} (X_{w_i,w_j}/x_{max})^\alpha & \text{if } X_{w_i,w_j} < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

where $0 < \alpha \leq 1$ and x_{max} is a positive number no greater than the maximum number of co-occurences in the dataset. In the original *GloVe* experiments, x_{max} is set to 100.

With the above objective function, the optimization problem is to minimize J with respect to a set of word vectors $\mathbf{v}$ and biases b. A variety of gradient descent algorithms are good for this optimization task. Gradients are defined as follows:

$$\nabla_{\mathbf{v}_{w_i}} J = \sum_{j=1}^{V} f(X_{w_i,w_j}) \left[\mathbf{v}_{w_i}^T \tilde{\mathbf{v}}_{w_j} + b_{w_i} + \tilde{b}_{w_j} - \log X_{w_i,w_j} \right] \tilde{\mathbf{v}}_{w_j} \tag{14}$$

$$\frac{\partial J}{\partial b_{w_i}} = f(X_{w_i,w_j}) \left[\mathbf{v}_{w_i}^T \tilde{\mathbf{v}}_{w_j} + b_{w_i} + \tilde{b}_{w_j} - \log X_{w_i,w_j} \right] \tag{15}$$

[6] Despite symmetry, the GloVe model, like word2vec, assigns to a word w two vectors: a target word vector $\mathbf{v}_w$ and a context word vector $\tilde{\mathbf{v}}_w$.

[7] Pennington et al. (2014), the method's original paper, has an extended elaboration of the choice of this link function with reference to linguistic regularity and mathematic properties of the functions. I skip the parts and only summarize the intuition here.

Finally, regarding computational complexity, the *GloVe* model scales sub-linearly with regards to the length of the text sequence T, outperforming *word2vec*, as its authors posit. The complexity of *GloVe* depends on number of co-occurring words, while that of *word2vec* depends on the total length of the text data. Recall that *word2vec* walks through the whole text sequence of length T, making complexity of one iteration through all data $O(T)$. In comparison, *GloVe* scales no worse than $O(V^2)$, that is, the maximum number of pairs of co-occurring words. This size could be huge especially when the dictionary is large. However, according to the assessment of *GloVe*'s authors, the total number of co-occurring words $|X|$ of a long sequence of text is usually much less than V^2. They estimate $|X| = O(T^{0.8})$, leading to the conclusion that *GloVe* outperforms *word2vec* in computational complexity (Pennington et al., 2014, Section 3.2). But later work refutes the claim (See Levy et al., 2018, p. 220).

2.2.3 Which algorithm is better?

With the architecture of *word2vec* and *GloVe* explained, I conclude the section by addressing an important question: Which algorithm is better? Existing research shows that neither algorithm dominates, while careful choice of parameters is critical.

Which algorithm is better? Existing works compare the two by discussing the similarity of their architecture and the differences between their performance in a variety of tests. First, studies show that the architecture of *word2vec* share a lot in common. Pennington et al. (2014) shows that *GloVe* can be considered a "global skip-gram" model: grouping Skip-gram's softmax terms globally by word co-occurrences, the objective function is close to that of *GloVe*. In addition, Levy and Goldberg (2014b) shows that a Skip-gram model with negative sampling is implicitly factorizing a word-context matrix of pointwise mutual information. Second, tested with a set of standardized tasks, neither algorithm dominates. Although the paper introducing *GloVe* claims that it beats *word2vec* in speed and accuracy, later assessment refutes it (Levy et al., 2018). A recent evaluation with political text also shows results of the two methods are not significantly different (Spirling and Rodriguez, 2019).

While it is not clear which algorithm is better, it is clear that careful choice of parameters is critical to the success of both algorithms. The two most important parameters are size of word vectors and length of context windows. First, size of word vectors (i.e., the number of dimensions) should be small enough for computational efficiency but large enough to

capture the meaning of words in a numeric space. As a rule of thumb, applications typically set the size as several hundred. Experiments with standardized tasks show that increasing word vectors beyond 300 dimensions has little marginal gain. Second, the length of context windows determines which words are considered "similar", which then determines which word vectors are made close to one another in the numeric space. Models with a small context window assign similar vectors to words of similar syntactic functions, while models with a large context window assign similar vectors to words of similar semantics and topical belonging. For example, with a small context window, the word "Hogwarts" (a fictional school from the Harry Potter series) has a word vector close to other fictional schools such as "Sunnydale" (from *Buffy the Vampire Slayer*). With a larger context window, the same word has a vector close to other words related to Harry Potter such as "Dumbledore" and "Snape," names of characters in the series (Jurafsky and Martin, 2019, Chapter 6; Levy and Goldberg, 2014a, p. 305). Beyond vector size and length of context window, a set of other parameters also affects performance: the choice of a weighting scheme to eliminate infrequent words and down-sample frequent words affects the quality of vectors and the size of the final dictionary; the number of iterations run affect the models' ability to converge to good word vectors that optimize their objective functions by gradient descent.

2.3 Distributed Documents: A Bag-of-Word-Vector Approach

The algorithms of distributed semantics learn informative representation for words, the basic units of text. However, to create measurement with text data, researchers care about larger units of text: documents and concepts. In this section, I introduce how to create informative representation for documents and concepts based on distributed semantics of words. Contrary to the complex modeling for word vectors discussed in Section 2.2, I use simple models to link words to documents for two reasons. First, with a set of useful properties of word vectors, simple models suffice. Second, simple models make human validation and intervention convenient.

I use a bag-of-word-vector approach to obtain distributed semantics of documents using learned distributed semantics of words. The design is simple, but its justification is non-trivial.

The model for distributed semantics of documents is simple: For a document, I take a

weighted sum of distributed semantics of all words appearing in it. I call it a bag-of-word-vector approach as it, just like the bag-of-word approach, disregards the order of words in a document. Formally, let D represent the collection of documents. Let d_i be a document in the text dataset, represented as a collection of words $d_i = \{w_1, w_2, ..., w_{M_i}\}$ where M_i is the number of unique words document d_i contains (the number of words may differ across documents). Let $\mathbf{v}(d_i)$ be the distributed semantics of document d_i. Let $f(d_i, w_j)$ be some weight assigned to word w_j in document d_i. Also, $\mathbf{v}_{w_j}$ is the distributed semantics of word w_j, as defined in Section 2.2.

$$\mathbf{v}(d_i) = \sum_{w_j \in d_i} f(d_i, w_j)\mathbf{v}_{w_j} \tag{16}$$

Following common practice in bag-of-word methods, I use the TF-IDF algorithm to weight word vectors in aggregating them into a document vector. Compared to a simple count, the method weights down words appearing frequently in a document with a non-linear term frequency function and weighs down words appearing in many documents with an inverse-document-frequency normalizer. This specification of TF-IDF algorithm follows Jurafsky and Martin (2019, Chapter 6).

$$f(d_i, w_j) = \text{tf}_{d_i,w_j} \times \text{idf}_{w_j} \tag{17}$$

$$\text{tf}_{d_i,w_j} = \begin{cases} 1 + \log_{10} \text{count}(d_i, w_j) & \text{if count}(d_i, w_j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$\text{idf}_{w_j} = \log_{10} \frac{|D|}{\text{df}_{w_j}} \tag{19}$$

where $\text{count}(d_i, w_j)$ denotes the number of words w_j in document d_i. df_{w_j} denotes the number of documents containing word w_j. $|D|$ denotes the total number of documents.

Taking the summation of distributed semanticsof words disregarding their order, is justified by their property of compositionality. Word vectors are compositional: the meanings of a phrase can be represented by the summation of vectors of words in the phrase. This important property is demonstrated in a set of examples by the authors of *word2vec*. For example, a trained word semantics can tell Lufthansa is a German air carrier as $\mathbf{v}_{German} + \mathbf{v}_{airline}$ is close to $\mathbf{v}_{carrier} + \mathbf{v}_{Lufthansa}$ (Mikolov et al., 2013, p. 7). The property of compositionality of short phrases is extended to longer documents in a few studies. Aggregating word vectors

with multiplicative and additive functions has proven effective in cognitive studies (Mitchell and Lapata, 2010). Representing documents by averaging vectors of words shows good performance in natural language processing tasks like sentiment analysis and question answering (Iyyer et al., 2015).

Admittedly, the simple linkage model is a radical simplification compared to existing applications in natural language processing. Many distributed semantics algorithms for documents use more complex modeling. For example, the *doc2vec* model learns a document-specific vector in each context window (Dai et al., 2015; Le and Mikolov, 2014). In the training processing, document-specific vectors are used as target and context in addition to word vectors (consider it the "document fixed-effect"). The *Skip-thought* model encodes distributed semantics of a sentence as a hidden layer of the final stage of a recurrent neural network and optimizes its predictive power of vectors of neighboring sentences (Kiros et al., 2015). In addition, word vectors are stacked by their order in documents and modeled with convolutional neural networks (Kim, 2014). On top of these examples, most deep neural network modeling aggregating distributed semantics of words into documents does not take the type of simplification I do. They have good reason not to do so: retaining information helps prediction. However, prediction is not the goal of my system.

I choose this simple model to achieve the important design feature for valid and replicable text-as-data measurement: informative representation. I achieve informative representation by embedding words, documents, and concepts in one space – one stable and generalizable space. Complex modeling fails the goal of mapping all text in one space, stability, and generalizability. First, when distributed semantics of documents go through complex transformation from word representation, they are no longer in the same space, making it hard for researchers to make sense of the relationship between vectors of words and vectors of documents. Second, the results are more unstable when I use more complex models in the system. Neural network models use stochastic optimization algorithms to learn parameters, which do not find global optimum. A system with multiple components of complex models that outputs unstable locally optimal parameters imposes challenges for researchers' evaluation and intervention. Last but not least, results learned by fitting complex models to a specific dataset are less generalizable. Words are phrases are used in similar ways across text data. Thus, their distributed semantics learned in one dataset can be used in another. With complex

modeling, word vectors reflect too much of document-level features of the dataset and cannot be generalized, defeating the purpose of using distributed semantics in the first place.

2.4 Decoding Distributed Semantics for Validation

Now that we have gone through the technical challenges to learn distributed semantics from data, how do we make sense of it? Put another way, after the meaning of words and documents has been "encoded" by algorithms, how can it be "decoded" again? Distributed semantics are several-hundred-dimensional vectors whose individual dimensions do not have interpretable meanings. This imposes challenges against researchers' evaluation.

A simple decoding method, a piece of distributed semantics can be decoded by reviewing the set of text whose vectors are close to it. Though absolute values of distributed semantics have no interpretable meanings, relative positions do. With this property, existing works usually evaluate the quality of distributed semantics by examining whether text considered similar by machines makes sense to humans. Cosine similarity is typically used to quantify the closeness of word vectors:

$$\cos(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) = \frac{\mathbf{v}_{w_i}^T \mathbf{v}_{w_j}}{||\mathbf{v}_{w_i}|| \times ||\mathbf{v}_{w_j}||} \tag{20}$$

This distance metrics is chosen over the others because it is scale invariant (the lengths of vectors are normalized in the denominator), and it is close to the objective functions of the algorithm (taking the dot product of word vectors). Note that, although cosine similarity is the most common similarity measure for vector semantics, as long as they are standardized (i.e. $||\mathbf{v}_{w_i}|| = ||\mathbf{v}_{w_j}|| = 1$), the Euclidean distance is simply a linear transformation of the cosine similarity measure:[8]

$$\begin{aligned} ||\mathbf{v}_{w_i} - \mathbf{v}_{w_j}||^2 &= ||\mathbf{v}_{w_i}||^2 + ||\mathbf{v}_{w_j}||^2 - 2\mathbf{v}_{w_i}^T \mathbf{v}_{w_j} && (21) \\ &= ||\mathbf{v}_{w_i}||^2 + ||\mathbf{v}_{w_j}||^2 - 2||\mathbf{v}_{w_i}|| \times ||\mathbf{v}_{w_j}|| \cos(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) && (22) \\ &= 2 - 2\cos(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) && (23) \end{aligned}$$

[8] I make use of this property when I fit word vectors into clustering algorithms that use Euclidean distance as distance metrics.

The decoding method is typically used to evaluate algorithms of distributed semantics: cosine similarity is expected to capture generic language features of words. Two types of standardized tasks are usually performed: word similarity/relatedness and word analogy. The former identifies a set of words, rates their similarity and relatedness by cosine similarity of word vectors, and then compares it with scores assigned by human coders. The latter examines if word analogy (e.g., Beijing is to China as Tokyo is to Japan) found by linear operations of word vectors and cosine similarity matches with human coders' judgment. For both tasks, there exist multiple widely accepted "ground-truth" datasets created by different researchers (see Levy et al., 2018, p. 217 for a review). My experiments show that the same method can validate distributed semantics of documents by examining if documents with close vectors are similar in their substantives.

2.5 Summary

In this section, I introduce distributed semantics, a new family of language processing algorithms that produces informative representation of text with low-dimensional dense vectors. I elaborate both the methods' general intuition and engineering details of two of its most important models: *word2v* and *GloVe*. I then introduce methods to aggregate word-level distributed semantics into document-level distributed semantics and methods to validate results.

Although distributed semantics has been widely applied in the natural language processing community, it has not been designed for or applied to text-as-data measurement in ways political scientists need. In typical applications, distributed semantics are used as tools for dimensional reduction and noise reduction. First, in standard natural language processing tasks with machine learning (e.g., sentiment analysis, named entity recognition, semantics parsing), input text data are embedded with pre-trained or locally trained distributed semantics primarily to increase generalizability of learned models. Second, in predictive modeling using text as predictors, text is treated as simply a type of noisy high-dimensional data, for which distributed semantics is applied to improve predicted performance. Neither typical applications use the method to understand text data and create measurements out of it. I develop a new system for political scientists to use distributed semantics for measurement, elaborated in Sections 3 and 4.

3 Stepwise Guidance: Building a Distributed Codebook

All systems for text-as-data measurement explicitly or implicitly build codebooks. A codebook is essential to a text-as-data study because it guides the decision on how raw text is transformed to quantified and simplified indicators. I argue that good codebook building should have the following three qualities: good coverage of data, relevance to concepts, and replicability of the final product. First, good coverage of data means a thorough understanding of what is in the text. A codebook usually selects only a small proportion of relevant information in the text data. But its building should be informed by the "universe" of the data. Second, codebook building relevant to concepts allows researchers to: incorporate prior information about concepts of interest, adjust concepts with reference to data, and precisely map the final coding scheme to final conceptualization. Third, a replicable codebook can be conveniently applied to other datasets in follow-up observational or experimental studies. State-of-the-art methods face limitations on at least one of these dimensions. In this section, I introduce a new method based on distributed semantics to improve the quality of codebook on all three dimensions for text-as-data measurement.

Existing text-as-data methods face a trade-off between coverage of data and relevance to concepts, while all have limitations in replicability. Recall that text-as-data methods can be categorized into two types in terms of whether machines or humans take the "lead" (see Section 1.1). First, in systems where machines "read" with minimal human input, coverage of data is maximized but relevance to concepts is sacrificed. Consider topic modeling, the most popular algorithm of the type. The algorithm maximizes coverage of data as it automatically creates a "coding scheme" considering all data. However, its relevance to concepts is low: with the algorithm's design, researchers can hardly use prior conceptualization to guide it; adjusting concepts with reference to intermediate output and mapping concepts to final output are also difficult because topics can be unstable and lack interpretability. Second, systems where humans "teach" machines face completely different challenges. Their relevance to concepts is high, as human coders have full control over the dictionary they apply, the subset of data they read, and in which way they want to link text data to their concepts. However, the coverage of data is low because researchers have little systematic understanding of what is in the data beyond the part they read. This problem is severe especially when the size of

data is far beyond human readers' capacity (e.g., millions of social media posts). Finally, for both types of methods, replicability is challenging: Coding rules in pure human coding and machine coding can be both hard to explain and not generalizable to new data.

I develop a new codebook building system, distributed codebook, to eliminate the trade-off between data coverage and relevance to concepts, and to improve replicability. The system is built on two simple principles: First, work with words, not documents; second, integrate judgments of humans and machines.

Principle 1: Work with words, not documents. I argue for three reasons to work with words instead of documents: political scientists care about words; working with words reduces data dimensionality; distributed semantics makes working with words pleasant. First, with recent techniques, researchers can quantify text in a large set of documents without examining the full dictionary of words from which they are drawn. However, interpreting text-as-data studies, political scientists primarily care about words: When analyzing results of topic modeling, researchers make sense of a topic by reading the top words associated with it; when quantified text is fit into supervised learning models, researchers interpret the coefficients or variable importance associated with indicators of a word or a group of words (e.g., a topic); when findings of observational text-as-data studies are taken to experiments, treatments are operationalized as words. The extensive use of words in interpretations makes it important to bring them back as the focal point. Second, working with words reduces the dimensionality of data, especially when the number of documents grows. In the "small data" era, it is efficient working with documents because the number of words outnumbers that of documents: consider the case where researchers study hundreds of documents that likely use tens of thousands of unique words. However, in the big-data era where researchers work with millions or billions of documents (e.g., social media posts), the situation is reversed. Now it is more efficient working with small of units of text like words re-used in many documents. Third and to the core of this paper, distributed semantics of words creates a space to link words, documents, and concepts, and to bridge prior information and data. It makes it much more pleasant to work with words than before. For these three reasons, the system focuses on words in development of codebooks, while putting documents in the background.

Principle 2: Integrate judgments of humans and machines. The system attempts to combine strengths of machines and humans for research with large and messy text data. Social

scientists starting a text-as-data research project with a large text dataset and some prior conceptualization iteratively perform three tasks: link a part of data to pre-formed concepts; adjust conceptualization in light of the data; and, importantly, discard a large proportion of irrelevant information in the data. Machines are powerful in summarizing and searching for information quickly in the full data space. But they cannot link data to concepts or discard irrelevant information. Humans are good at making sense of data. But they only have the capacity to make judgment on a selection of data. I develop a standardized procedure to combine distributed semantics, unsupervised machine learning models, and human reading, to facilitate the three tasks, with a goal to produce stable and interpretable codebooks.

The rest of this section introduces my system for distributed codebook building. Section 3.1 introduces how to summarize concepts from data. Section 3.2 introduces how to combine prior information of concepts with data. Section 3.3 discusses methods to apply a distributed codebook to a dataset of documents. Finally, Section 3.4 summarizes and discusses how the method of distributed codebooks is related to existing methods.

3.1 Exploratory Concepts Clustering

What concepts of interest do the text data contain? Researchers usually want to inductively learn from the data to help conceptualization. But the size of the text data can be overwhelmingly large. I introduce a simple method to summarize concepts from data: exploratory concepts clustering. I suggest fitting distributed semantics of words to unsupervised learning models to assist with the task. Intuitively, I assume that words in the dictionary are generated from a set of concepts. Then, operationalizing it in the space of distributed semantics, word vectors can be clustered with unsupervised learning models for high-dimensional continuous variables. Thus, clusters learned by machines can represent concepts.

I model the distributions of word vectors and concepts with a Gaussian Mixture Model (GMM), a natural choice for the type of task. Let $\mathbf{v}_{w_i}$ be the N-dimensional distributed semantics of word w_i. Assume a word w_i is generated from a mixture of K concepts, then its word vector $\mathbf{v}_{w_i}$ is modeled as generated from a mixture of K multivariate normal distribu-

tions. For a word w_i, the distribution of its word vector $\mathbf{v}_{w_i}$ is modeled as below:

$$p(\mathbf{v}_{w_i}) = \sum_{j=1}^{K} p(\mathbf{v}_{w_i}, c_{w_i} = j) \tag{24}$$

$$p(\mathbf{v}_{w_i}, c_{w_i} = j) = p(\mathbf{v}_{w_i}|c_{w_i} = j)p(c_{w_i} = j) \tag{25}$$

$$p(\mathbf{v}_{w_i}|c_{w_i} = j; \mu, \Sigma) \overset{\mathcal{D}}{=} \mathcal{N}(\mu_j, \Sigma_j) \qquad p(c_{w_i}; \phi) \overset{\mathcal{D}}{=} \text{Multinomial}(\phi) \tag{26}$$

The model is trained to maximize its likelihood with respects to three parameters: the weights of word vectors' belonging to concepts ϕ, and the means and variances of the distributions of concepts μ, Σ:

$$\max_{\phi,\mu,\Sigma} \log \mathcal{L}(\phi, \mu, \Sigma) = \sum_{i=1}^{V} \log p(\mathbf{v}_{w_i}; \phi, \mu, \Sigma) \tag{27}$$

$$= \sum_{i=1}^{V} \log \sum_{j=1}^{K} p(\mathbf{v}_{w_i}|c_{w_i} = j; \mu, \Sigma)p(c_{w_i} = j; \phi) \tag{28}$$

I use the Expectation-maximization (EM) algorithm to estimate the parameters, following numerous previous applications of the GMM. With the estimated parameters, I obtain the probability of each word in the dictionary belonging to one of the concepts. A concept can be interpreted in terms of the set of words with highest probability belonging to it.

The above implementation comes with two problems: The number of concepts K is hard to decide; the results of EM are local optima. First, researchers need to input the number of concepts K as the parameter for a GMM model, but there is usually no theoretically grounded reason to make this choice, especially in the early phase of the study. An overestimated K breaks a concept into multiple clusters and increases researchers' burden in manual examination. An underestimated K conflates multiple concepts into one cluster or overlooks important groups of concepts. Second, the GMM is multimodal and the EM algorithm by design does not search for the globally optimal solution. A undesirable consequence, the output GMM depends largely on initial random seeds provided, making it unstable. The two problems lead to insufficient data coverage, imposing threats to the validity and replicability of measurement.

I propose a simple but effective remedy, a trial-select-merge workflow: run the model

many times, hand-pick relevant clusters of concepts found in each trial, and merge them. First, researchers run the GMM model as many times as they want with a varying number of concepts K and initial random seeds. Then, for each model, a table with top words in each cluster is reported (e.g., words belonging to the cluster with over 90% probability). Researchers manually examine each of the tables to assign labels to cluster that are relevant to their concepts of interest. Finally, clusters that are assigned labels across models are merged into the distributed codebook, while all unlabeled clusters (i.e., those that researchers considered irrelevant) are discarded.

This trial-select-merge workflow is possible because of the property of distributed semantics. Without distributed semantics, the probabilities of all words' concept belonging are needed for linking documents to concepts. As a result, combining parameters of distributions across models is not valid or at least takes some additional modeling effort. However, with distributed semantics that embed documents and words in the same space, the GMM only serves to search for local optima in the space of distributed semantics and return top words in local dense area for human evaluation. The task of document coding is done in another step: I combine the distributed semantics of documents and that of representative words in concepts to label documents with concepts (details discussed in Section 3.3).

This method for exploratory analysis helps researchers understand what is in the text data efficiently. It is especially helpful when a researcher explores the relationship between his initial conceptualization with little prior information about the data and concepts of interest. The method clusters words instead of documents to reduce dimensionality of data and make results more interpretable. As distributed semantics captures words' semantics, clusters of concepts found with the method are much more coherent and interpretable than those found in topic modeling. In addition, the method integrates judgements of machines and humans with the trial-select-merge workflow. It reduces the risk of insufficient data coverage caused by machines' randomness and humans' limited capacity.

3.2 Combining Priors with Data: Distributed Dictionary Augmentation

Researchers usually want to incorporate prior information about concepts of interest in their text data analysis. Prior information can be operationalized as a dictionary of keywords. I call

this type of dictionary "seed dictionaries". Seed dictionaries can come from numerous sources: random keywords coming out of brainstorming; exploratory analysis of the documents and words in the data (e.g., clusters found with the method introduced in Section 3.1); external dictionaries produced by other researchers; results of off-the-shelf natural language processing algorithms (e.g., named entity recognition, keyword extraction, sentiment analysis).

Direct application of a seed dictionary can cause two problems: lack of coverage and misplaced context. First, words in a seed dictionary may cover only a very small proportion of words used in the text dataset of interest. Using the dictionary directly to code the text data will miss context of similar meaning that is not expressed in the same way as the words included in the dictionary. Second, the word may have different meanings in different contexts. Using an off-the-shelf seed dictionary directly can cause mislabeling of meanings of words in a specific context. The problem can be especially severe in dataset where use of language is informal and unconventional, such as social media political.

To customize seed dictionaries for local use, I introduce a simple method based on distributed semantics: distributed dictionary augmentation. The method takes only two steps: match and augment, and manual adjustment. In addition, an optional step between Step 1 and 2 maybe added: unsupervised or semi-supervised clustering.

Step 1: match and augment. The step starts with finding the intersection: a set of words that appears in both the seed dictionaries and the data' s dictionary of distributed semantics. For each word in the intersection, a set of its most similar words in the space of distributed semantics are added to an augmented dictionary. Words' similarities are defined in terms of cosine similarities between word vectors (see Section 2.4). Researchers choose a threshold of consine similarity above which a word qualifies as a new addition to the augmented dictionary. There is a trade-off regarding the choice of the threshold: a higher threshold usually leads to a more coherent augmented dictionary but risks missing relevant words; a lower threshold leads to better inclusion of relevant words but a messier augmented dictionary. In my experiments, setting cosine similarity higher than 0.5 as the threshold is a reasonable choice. The output of this step is an automatically generated augmented dictionary where new additions are assigned the same label as their seeds.

Step 2: manual adjustment. Researchers manually edit the augmented dictionary. Two types of problems may appear in the dictionary automatically generated in the first step: the

new additions may be irrelevant, or the seed words may be mislabeled. First, new additions to the dictionary can be irrelevant, especially when the threshold is set low. Researchers can remove or relabel these words. If a high proportion of irrelevant words is observed, a proper fix is going back to Step 1 to increase the threshold. Second, the seed words can be mislabeled. Researchers can infer from seed words' similar words about their use in the context of the text data. In some cases, they may find a seed word has a different meaning in the dataset from the seed dictionary.

An optional step may be added between Step 1 and 2: unsupervised or semi-supervised clustering of the machine-generated augmented dictionary. Researchers may use unsupervised or semi-supervised learning algorithms to cluster distributed semantics of words in the augmented dictionary before manual adjustment. This between-step is especially helpful in three situations. First, when a low threshold is chosen in Step 1, the algorithms can be applied to identify groups of irrelevant words. Second, when concepts in the dictionary are close to one another, the algorithms can help to decide belonging of borderline words. Third, when researchers are interested in dividing concepts into sub-concepts automatically, they can use cluster algorithms setting the number of clusters higher than the number of concepts in the seed dictionary. Whether to use unsupervised or semi-supervised clustering depends on how much researchers want to keep the original concept grouping in the seed dictionary. In general, among the above three situations, semi-supervised learning is better for the first two situations, while unsupervised learning is better for the third situation.

The method of distributed dictionary augmentation helps researchers combine prior information with data. Like exploratory concepts clustering, it reduces the dimensionality of data researchers manually work with by focusing on words; produces a stable and interpretable codebook with a standardized workflow to combine judgments of humans and machines.

3.3 Applying a Distributed Codebook to Documents

With exploratory concept clustering (Section 3.1) and distributed dictionary augmentation (Section 3.2), researchers can build a reliable codebook. The next step is to label documents with the distributed codebook. A straightforward application of properties of distributed semantics, this step labels how close a document is to a concept defined in the distributed

codebook referring to closeness of their representations of vector semantics. I introduce three alternatives of similarity measures: document-level cosine similarity, word-level binarized cosine similarity, and count of words. Researchers may choose one of the measures according to the length of the text data, peculiarity of word choice, and meanings of weights associated with words in concepts.

To start with, the distributed semantics of a concept is calculated as a summation of vectors of its associated words. For a concept c_k:

$$\mathbf{v}_{c_k} = \sum_{w_j \in c_k} \mathbf{v}_{w_j} \tag{29}$$

3.3.1 Document-level Cosine Similarity (cossim-d)

The document-level cosine similarity measure quantifies a document's relation with a concept by the cosine distance of their distributed semantics. Let $\text{score}_{d_i}^{c_k}$ be a score indicating the closeness between document d_i and concept c_k:

$$\text{score}_{d_i}^{c_k} = \cos(\mathbf{v}_{d_i}, \mathbf{v}_{c_k}) \tag{30}$$

Note that when the similarity level drops beneath some low level, differences in values are irrelevant: they are equally irrelevant in human judgment. Researcher may consider truncating the above measure. Let the lower bound be s_{low}:

$$\text{score}_{d_i}^{c_k} = \max\left(\cos(\mathbf{v}_{d_i}, \mathbf{v}_{c_k}), s_{\text{low}}\right) \tag{31}$$

Documents whose similarity with all concepts are at the lower bound should be considered irrelevant. Researchers can remove them from the dataset or assign them a separate label (e.g. a control or placebo group for a text-for-causal-inference study), depending on their research designs.

This measure is good for short documents. As discussed in Section 2.3, calculating distributed semantics of documents with a bag-of-word-vector is a radical simplification. The measure can be problematic with long documents because simple addition of a large number of word vectors goes beyond the verified compositionality property of distributed semantics,

so that its property is unknown.

3.3.2 Word-level Binarized Cosine Similarity (cossim-w)

A document can be considered close to a concept when a significant number of its words are close enough to it, but not exactly the same. The word-level binarized cosine similarity measure codes documents with this logic, combining the strength of distributed semantics and the traditional count-based method. Let s_{low} be the lower-bound cosine similarity to consider a word associated with a concept:

$$\text{score}_{d_i}^{c_k} = \sum_{w \in d_i} \mathbb{1}\left(\cos(\mathbf{v}_w, \mathbf{v}_{c_k}) > s_{\text{low}}\right) \times \text{tf}_{d_i,w} \tag{32}$$

$$\text{where } \text{tf}_{d_i,w} = 1 + \log_{10} \text{count}(d_i, w) \tag{33}$$

I use a shifted log-transformation of counts to reduce the score of long documents. This is especially useful for datasets with a high variance in length of documents: Without this penalty, long documents are more likely to get high similarity scores to all concepts compared to short documents.

Researchers can go further to compare vectors of words in documents with vectors of each representative word in concepts. I argue that the payoff of this exercise is small. Distributed semantics of representative words in concepts are by design very close. Thus there is little chance of an outlier far away from their mean. If the researcher is particular about use of words, a pure count-based similarity measure should be applied.

3.3.3 Count of Words (count-w)

The third coding scheme uses the traditional count-based method. A document' s closeness to a concept is measured by the count of its words in the keyword dictionary of the concept.

$$\text{score}_{d_i}^{c_k} = \sum_{w \in d_i \cap c_k} \text{tf}_{d_i,w} \quad \text{where } \text{tf}_{d_i,w} = 1 + \log_{10} \text{count}(d_i, w) \tag{34}$$

Again, I use a log-transformation to penalize the score of long documents. This traditional count-based method is useful for tasks of document coding that have strict requirements on

use of exact words in the codebook.

3.3.4 Incorporating Weights

In many codebooks, words are associated with weights. For example, codebooks for sentiment analysis usually assign scores of polarity and intensity to words; researchers may assign weights indicating level of relevance of words to the crux of a concept. The latter two measures, word-level binarized cosine similarity and count of words, can conveniently incorporate such weights. Let weight_{w,c_k} be the weight associated with a word w in concept c_k:

$$\text{(cossim-w)} \quad \text{score}_{d_i}^{c_k} = \sum_{w \in d_i} \mathbb{1}\left(\cos(\mathbf{v}_w, \mathbf{v}_{c_k}) > s_{\text{low}}\right) \times \text{tf}_{d_i,w} \times \text{weight}_{w,c_k} \tag{35}$$

$$\text{(count)} \quad \text{score}_{d_i}^{c_k} = \sum_{w \in d_i \cap c_k} \text{tf}_{d_i,w} \times \text{weight}_{w,c_k} \tag{36}$$

3.3.5 Caveats and Summary

How to make sense of these similarity measures? The above similarity scores do not have a substantively meaningful unit and, importantly, do not indicate the "probability" documents contain certain concepts. First, the absolute values of these measures only indicate the closeness of distributed semantics between a pair of concepts and documents. As discussed in Section 2.4, distributed semantics are only interpretable by their relative positions, not their absolute values. As a result, the absolute value of these measures based on distributed semantics do not have a substantively meaningful unit. However, these measures are intervals: rankings and gaps among measures are meaningful.

Second and importantly, the similarity measures should not be interpreted as anything related to the probability of a document containing a certain concept or the proportion of a certain concept in a document. The lack of linkage with probability and proportion is by design: the cosine similarity measure has no relation with proportion or probability; the truncations in cossim-d and binarization in cossim-w I suggest further pull them away from probability and proportion interpretations. This design reflects my view of what constitutes the crux of document coding in text-as-data measurement: I see it as a task of information extraction instead of summarization. Summarization considers the universe of content when labeling documents, while information extraction first makes a decision on what is relevant

and then only cares about relevant content in document labeling. I argue that, as text data is becoming larger and messier, coding document as information extraction helps researchers filter out noise and focus on what is relevant to their substantive interest.

To summarize, I suggest three simple similarity measures to code text documents with a distributed codebook. The coding scheme is simple because concepts and documents are represented with distributed semantics in one space. Their simplicity and flexibility enable researchers to control and explain the coding process. Researchers should interpret the measures with caution, as discussed above. To improve the robustness of coding, researchers are advised to experiment with different versions of similarity measures and use a selection of good ones as final measures.

3.4 Summary and Relevance to Existing Methods

In this section, I present an original method to code documents: a distributed codebook approach. The method is built on distributed semantics that embeds words, documents, and concepts in one space. The method is developed with two principles: working with words instead of documents and integrating judgements of humans and machines. I introduce two approaches to build a distributed codebook linking text data with concepts of interest: exploratory concepts clustering and distributed dictionary augmentation. I suggest three similarity metrics to apply a distributed codebook to documents in the data: document-level cosine similarity, word-level binarized cosine similarity, and count of words.

The distributed codebook approach draws inspiration from many classic and recent text-as-data methods. First, the design of the human-machine integration system draws from my experience in text-as-data analysis and conventional practice in traditional content analysis (Neuendorf, 2002). This approach replaces some important part of tedious human labor with machine reading. Second, exploratory concepts clustering draws from the popular topic modeling with Latent Dirichlet Allocation (Blei et al., 2003; Roberts et al., 2016). This approach makes the unsupervised model simpler and results more interpretable and stable by shifting the focus from documents to words and using a trial-select-merge workflow. Third, the approach borrows from insights of in the weakly supervised learning literature on using unlabeled or messily labeled data to improve supervised learning models (Basu et al., 2002;

Eisenstein, 2019, Chapter 5; Jain, 2010; Zhou, 2018). A part of the workflow incorporates semi-supervised clustering algorithms. What is unique in my approach is that, instead of letting only machines learn from humans, it allows humans and machines to learn from one another in the process of data analysis. Finally, recent efforts in multiple disciplines use distributed semantics to summarize or extract information from text. The *lda2vec* model combines word embedding and topic modeling to summarize text data (Moody, 2016). The *DDR* method combines distributed semantics and psychological dictionaries for text coding and dictionary augmentation (Garten et al., 2018). Built on the strength of many classic and recent works, the distributed codebook approach introduced in this section is the first system for text-as-data measurement based on distributive semantics in political science.

4 ATIOS: A System for Valid and Replicable Measurement

In this section, I introduce ATIOS, All Text in One Space, a system for valid and replicable text-as-data measurement. In previous sections, I have introduced two integral parts of the system: Section 2 introduced distributed semantics, a method to represent words and documents with dense and low-dimensional vectors that quantify their substantive meanings; Section 3 introduced a new approach to develop a reliable codebook combining judgment of humans and machines and applied it to label documents. The rest of this section contains three parts: First, I introduce a typical workflow with ATIOS. Second, I discuss how it increases measurement validity. Third, I discuss how it increases measurement replicability.

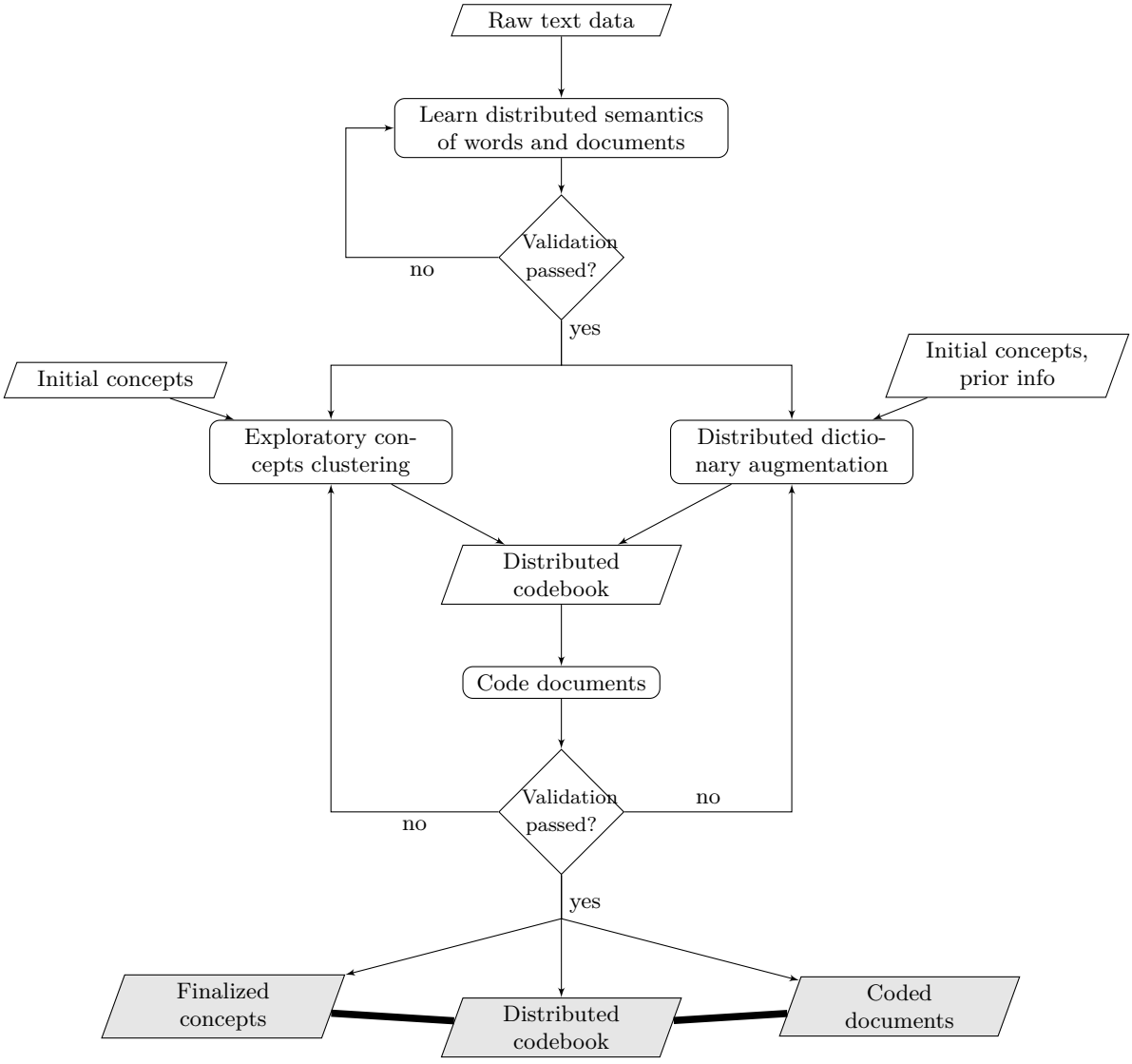
4.1 A Typical Workflow with ATIOS

A typical workflow with ATIOS contains three blocks: learning distributed semantics, building and applying a distributed codebook, and output. In this part, I walk through a full workflow with ATIOS, as visualized in Figure 5.

Block 1: Learning distributed semantics. An initial step, text of documents are fitted into algorithms of learn distributed semantics (e.g., *word2vec* or *GloVe*). Raw text data need minimal pre-processing beyond tokenization.[9] Removal of extremely rare words helps

[9]Common pre-processing steps for existing methods are not encouraged, such as stop word removal, lemmatization, and stemming. This is because algorithms of distributed semantics learn meanings of words from their context. This makes keeping documents in their original forms helpful. In addition, the algorithm can

Figure 5: All Text In One Space (Full workflow)



improve word vectors.[10] Researchers can choose between the two major algorithms introduced. They can also customize a set of parameters, such as the size of vectors and the length of context windows. A learned set of distributed semantics is validated by querying similar words. Researchers can input a selection of words, search for their similar words measured by respective cosine similarities, and evaluate the quality of distributed semantics. Researchers are advised to settle on a set of distributed semantics before proceeding to Block 2. Note that the algorithms of this step are unstable by design because they are trained with stochastic optimization. Though I encourage researchers to try different model specifications to search for the best model, I would also remind researchers to feel comfortable moving on even if some similarity queries of a model "do not make sense," because errors can be corrected by human intervention in Block 2.

Block 2: Building and applying a distributed codebook. After researchers have validated a version of distributed semantics, they can use word vector representation to find concepts with two methods: Exploratory concepts clustering uses an unsupervised learning algorithm to cluster word vectors for researchers to label; distributed dictionary augmentation combines a seed dictionary with distributed semantics to produce customized codebook. The two methods each contain back-and-forth interactions between humans and machines. Final products of the two methods are combined into a distributed codebook, which can then be applied to code documents in the data. Three alternatives of similarity scores are available. The coded documents are subject to validation, by examining whether documents most relevant to a selection of concept makse intuitive sense to humans. Researchers can choose among them according to length of documents, fuzziness about word choice, and evidence in validation. I suggest reporting multiple scores for robustness check.

Block 3: Output. When coded documents pass researchers validation, three final output can be prepared: finalized concepts, coded documents, and a distributed codebook. First, finalized concepts are documented verbally, like many other text-as-data studies. Researchers can describe what concepts are being measured with the text data. Second, coded documents are a table of document IDs and their similarity scores with each concept described in the set of finalized concepts. Finally and importantly, a distributed codebook is prepared, linking verbal

group words of similar meaning in a dense space, so researchers need not worry about including different forms of the same word causing high dimensionality.

[10] I find setting the lower bound of total frequency 10 reasonable in my experiments.

expression of concepts and coded documents. A distributed codebook typically contains two elements: a table of concepts and their associated words, and a corpus of distributed semantics containing vector representations of all words in the documents.

4.2 ATIOS for Measurement Validity

ATIOS addresses the challenges of measurement validity for text-as-data measurement by combining a robust language model, a standardized workflow that can be clearly documented and reasoned, and an informative set of outputs.

First, the foundational algorithm for this system, distributed semantics, is mature and robust, with verified theoretical foundation, predictive power, and stability. The algorithm is based on a solid linguistic theory, the distributional hypothesis and distributional representation, studied by linguists for decades. It has been proven effective in boosting predictive performance of natural language processing tasks in the computer science literature. In addition, despite its stochastic optimization procedure, the algorithm produces more stable results than many other text-as-data and machine learning models, because of its simple architecture (compared to other neural network models) and computational linguists' long-time efforts in improving its training algorithm.

Second, the workflow to combine judgement of researchers and machines can be clearly documented and reasoned. Admittedly, this system is not simpler than any other existing methods: the stepwise guidance may even look complicated. However, clarity, not simplicity, is my objective. Applications of most methods for text-as-data measurement go through a messy iterative process, as reviewed in Section 1.1. However, these iterative processes are usually opaque and untraceable because researchers lack tools to document and reason about their choices in the iterative steps, causing difficulty in justifying measurement validity. ATIOS provides such a tool. It modularizes machine algorithms into three parts and spreads human intervention across them: learning language representation, building a reliable codebook, and obtaining document labels. The design imitates a natural process of content analysis. The system's intermediate output is clear and self-explanatory: all iterations are documented and under researchers' control; reasoning is provided on how humans and machines supplement each other's limitations in the process.

Third, final output of ATIOS contains an informative map between concepts and data, accessible and falsifiable by readers. Key to its final output, a distributed codebook links verbal distribution of concepts of interest and quantified labels of documents. It allows researchers to present a concrete and concise coding scheme for readers' evaluation. With a distributed codebook, readers are enabled to criticize a text-as-data study based on specific mis-conceptualization or mis-labeling. They can also suggest specific robustness checks without tasking researchers to re-run the whole analysis. Compared to many existing methods that output a combination of verbal description of concepts, black-box models, and a few example document-label pairs, I argue that ATIOS produces output that enhances measurement validity and makes its communication straightforward.

4.3 ATIOS for Measurement Replicability

ATIOS produces a set of replicable output for follow-up observational and experimental studies. The key to the system's replicability is a distributed codebook in its final output. It can be used in four ways for different replication tasks: for replication with similar observational text data, the learned distributed semantics can be directly applied; for replication with moderately different observational data, the learned distributed semantics can be fine-tuned; for replication with significantly different observational text data, the vocabulary can be used as a seed dictionary; for follow-up experimental studies, the distributed codebook guides their designs in multiple ways.

A distributed codebook should be used in different ways for different replication tasks because of the property of distributed semantics. A brief recap, algorithms of distributed semantics learn quantified meanings of words from the context they are placed. Vector representation of words makes sense through relative instead of absolute positions. As a result, distributed semantics are contextualized on the language patterns of text data from which it is trained and researchers should use caution when applying them to another dataset.[11]

First, to replicate a study with a similar dataset, the procedure is straightforward: researchers simply use the distributed codebook to embed documents of the new dataset, then apply the same coding scheme to code documents.

[11] I should add that it is already more generalizable than many text-as-data modeling working with documents.

Second, a slightly trickier case is replication with a moderately different dataset: researchers may want to make some slight adjustment to the learned distributed semantics to fit local context. In this case, vector representations produced by the previous study can be input as initial seed of the current study's embedding step to fine-tune the distributed semantics. After fine-tuning, researchers can follow the coding scheme of the original step to embed and code documents.

Third, for significantly different text data, it does not make sense to use any of the original distributed semantics. In this case, the distributed codebook is used as a traditional dictionary mapping keywords to concepts. Researchers can use the codebook as a seed dictionary in the step of distributed dictionary augmentation.

Finally, for designs of follow-up experimental studies, the distributed codebook can serve as a guideline in designs of text as treatment. Keywords of concepts can be used as treatments per se. Semantic closeness between words and concepts of interest measured by distributed semantics can inform the relevance and strength of certain treatments. Clusters of words considered irrelevant in the coding process can inform design of text presented to control and placebo groups.

5 Conclusion: A Low-tech and Unautomated System

In political science studies, researchers are interested in creating valid and replicable measurement based on text data. The task has become increasingly challenging in the "big data" era: the growing size of text data available exceeds researchers' capacity to manually examine; frontier machine learning algorithms developed in the computer science community are not designed to help political science researchers analyze data for scientific inquiries. In this paper, I address the challenge by building a system around a frontier machine learning algorithm for political scientists. First, I introduce distributed semantics: a natural language processing algorithm to represent text data in low-dimensional dense vectors. This is the first comprehensive introduction of the intuition and engineering details of this most important natural language processing algorithm to the political science community. Second, building upon distributed semantics, I present a new system for text-as-data measurement featured by its capacity to integrate judgment of humans and machines in the analysis process and to

produce informative output facilitating replication. This is the first text-as-data system in political science incorporating distributed semantics in its workflow. More importantly, it is a method that improves upon existing methods in the validity and replicability of text-based measurement.

The system is being extended in multiple ways. First, software development is work in progress. I have developed a Python program for its applications to an example dataset of political communication in a Chinese social media site. Ongoing efforts include: application of the method to more example datasets, including datasets of interest to the political science community and standardized test datasets of interest to the computer science community; development of an R program; development of a graphic user interface for researchers not familiar with coding. Second, the workflow of stepwise guidance can be further improved. For example, algorithms can be developed to weight or pre-select important words for more efficient exploratory concepts clustering. A collection of suggested seed dictionaries can be provided by the system. More methods helping researchers validate and intervene in machine coding are to be added. User-friendly evaluation metrics are being developed to help researchers understand and report the quality of results in each step. Third, the system can be extended to allow researchers to choose larger basic units of text, such as sentences or paragraphs. Despite the advantage of using words as basic unit in reducing dimensionality and increasing interpretability, researchers may find it useful to use larger basic units of text: phrases, sentences, or paragraphs. A simple extension, collocation extraction algorithms can be applied to detect phrases before learning distributed semantics (e.g., using Pointwise Mutual Information to find frequent co-appearing words and group them into a token) (Eisenstein, 2019, Chapter 14). With further extensions, the system can provide options to replace word embedding with sentence or paragraph embedding to fit the need (Dai et al., 2015; Le and Mikolov, 2014; Kiros et al., 2015). Fourth, a farther extension, the system can incorporate more complicated representation of text data. For example, recent models of distributed semantics such as ELMo and BERT can be applied in an interpretable way (Devlin et al., 2018; Peters et al., 2018).

The system is compatible with existing methods of supervised document coding and recent developments in crowdsourcing methods for text-as-data model validation. First, though the system is fully capable of producing final output of text-as-data measurement, researchers

can use its results as input of supervised document coding methods. This can be especially relevant when contents in documents are exceptionally complicated and researchers would like to manually adjust the results for robustness. Second, the system is compatible with latest developments in crowdsourcing evaluation of text-as-data models (Spirling and Rodriguez, 2019; Ying et al., 2019). All steps of human intervention and evaluation (e.g., distributed semantics validation, exploratory concept clustering, distributive document augmentation, and document coding validation) can be crowdsourced, as opposed to having researchers doing everything alone. In fact, intermediate and final output of this system is more friendly to crowdsourcing effort than many other existing methods since they are interpretable and replicable.

I conclude this paper by *embedding* the method in a larger theme: big data for political science research. The method presented in this paper is a low-tech and unautomated system. First, the method is low-tech in that it does not use any latest or complicated method of artificial intelligence. To the contrary, it is based on a mature and robust method that is theoretically founded and verified in applications for a few years. The architecture of the models is simple and straightforward compared to recent developments. Second, the method is unautomated or even anti-automation. As evident in its design, researchers' intervention, validation, and interpretation are requested everywhere in the process of data analysis.

I develop the low-tech and unautomated system to address two related inquiries on how to integrate humans and machines for big-data research in social sciences, one raised by political scientists and the other by data scientists. First, political scientists have long been aware of the importance of integrating humans and machines in research with big data, but few works in the line provide a practical guide. Specific to text-as-data methods, Grimmer and Stewart (2013), a ground-breaking piece on text-as-data in political science, proposes four important principles for the method. Two of them are about human-machine integration: "quantitative methods augment humans, not replace them" and "validate, validate, validate." The anti-automation design of this system is a direct response to the principles.

A related inquiry comes from the data science community on interpretable machine learning. A recent contribution, Rudin (2019) has a self-explanatory title: "stop explaining blackbox models for high-stakes decisions and use interpretable models instead." Warning that explaining blackbox models can cause "catastrophic harm to society," Rudin argues

that researchers should make models interpretable by design, instead of looking for ways to unpack blackbox algorithms. Consistent with this advice, I apply a mature and simple (low-tech) model to minimize the uncertainty induced by the algorithm blackbox and maximize interpretability with an unautomated system.

Finally, this paper contributes to an ongoing discussion on how to make theory, big data, and causal inference work together in political science. Political scientists have agreed that theory, big data, and causal inference are compatible trends that have the opportunities to enhance one another (Clark and Golder, 2015; Grimmer, 2015; Monroe et al., 2015). However, the scholarship is still exploring how to make it happen. The system introduced in this paper attempts offer a possibility to link big text data to theories and causal identification: the stepwise guidance in distributed codebook building links theory-driven conceptualization to data; the output of distributed codebook makes follow-up experimental studies convenient. With this system and its extensions, I attempt to provide a generic user-friendly tool serving researchers interested in exploring big text data per se, as well as those who use big text data as an inspiration for theory-building and experimental designs.

References

Adcock, R. and D. Collier (2001). Measurement Validity: A Shared Standard for Qualitative and Quantitative Research. *American Political Science Review 95*(3), 529–546.

Arora, S., Y. Liang, and T. Ma (2017). A simple but tough to beat baseline for sentence embeddings. *Iclr*, 1–14.

Basu, S., A. Banerjee, and R. Mooney (2002). Semi-supervised Clustering by Seeding. *Proceedings of the 19th International COnference on Machine Learning (ICML-2002)* (July), 19–26.

Blei, D. M. (2012). Probabilistic Topic Models. *Communications of the Acm 55*(4), 77–84.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*, 993–1022.

Chang, J., S. Gerrish, C. Wang, and D. M. Blei (2009). Reading Tea Leaves: How Humans Interpret Topic Models. *Advances in Neural Information Processing Systems 22*, 288—-296.

Clark, W. R. and M. Golder (2015). Big data, causal inference, and formal theory: Contradictory trends in political science? *PS - Political Science and Politics 48*(1), 65–70.

Dai, A. M., C. Olah, and Q. V. Le (2015). Document Embedding with Paragraph Vectors. pp. 1–8.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (Mlm).

Egami, N., C. J. Fong, J. Grimmer, M. E. Roberts, and B. M. Stewart (2018). How to Make Causal Inferences Using Texts. *Working paper*.

Eisenstein, J. (2019). *Natural Language Processing*. MIT Press.

Garten, J., J. Hoover, K. M. Johnson, R. Boghrati, C. Iskiwitch, and M. Dehghani (2018). Dictionaries and distributions: Combining expert knowledge and large scale textual data content analysis: Distributed dictionary representation. *Behavior Research Methods 50*(1), 344–361.

Gentzkow, M., B. T. Kelly, and M. Taddy (2017). Text as Data.

Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research 57*, 345–420.

Goldberg, Y. and O. Levy (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. (2), 1–5.

Grimmer, J. (2015). We're All Social Scientists Now : How Big Data , Machine Learning , and Causal Inference Work Together. *PS: Political Science & Politics 48*(1), 80–83.

Grimmer, J. and B. M. Stewart (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis 21*(3), 267–297.

Iyyer, M., V. Manjunatha, J. Boyd-Graber, and H. D. III (2015). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings ofthe 53rd Annual Meeting ofthe Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1681–1691.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters 31*(8), 651–666.

Joos, M. (1950). Description of language design. *The Journal of the Acoustical Society of America 22*(6), 701–707.

Jurafsky, D. and J. H. Martin (2019). *Speech and Language Processing: An Introduction to Natual Language Processing, Computational Linguistics, and Speech Recognition (3rd Edition)*.

Kim, S. E. (2018). Media bias against foreign firms as a veiled trade barrier: Evidence from Chinese newspapers. *American Political Science Review 112*(4), 918–938.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings ofthe 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751.

King, G. (1995). Replication, replication. *PS: Political Science & Politics 28*(3), 444–452.

King, G. and D. J. Hopkins (2010). A Method of Automated Nonparametric Content Analysis for Social Science. *American Journal of Political Science 54*(1), 229–247.

King, G., J. Pan, and M. Roberts (2013). How censorship in China allows government criticism but silences collective expression. *American Political Science Review 107*(917), 326–343.

King, G., J. Pan, and M. E. Roberts (2017). How the Chinese government fabricates social media posts for strategic distraction, not engaged argument. *American Political Science Review 111*(3), 484–501.

Kiros, R., Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler (2015). Skip-Thought Vectors. (786), 1–11.

Lau, J. H., D. Newman, and T. Baldwin (2015). Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality. pp. 530–539.

Le, Q. V. and T. Mikolov (2014). Distributed Representations of Sentences and Documents. In *Proceedings of the 31 st International Conference on Machine Learning*, Volume 32.

Levy, O. and Y. Goldberg (2014a). Dependency-Based Word Embeddings. In *Proceedings ofthe 52nd Annual Meeting ofthe Association for Computational Linguistics*, pp. 302–308.

Levy, O. and Y. Goldberg (2014b). Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems (NIPS)*, 2177–2185.

Levy, O., Y. Goldberg, and I. Dagan (2018). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics 3*, 211–225.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient Estimation of Word Representations in Vector Space. pp. 1–12.

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of NIPS 2013*, 1–9.

Mikolov, T., W.-t. Yih, and G. Zweig (2013). Linguistic regularities in continuous space word representations. *Proceedings of NAACL-HLT* (June), 746–751.

Miller, B., F. Linder, and W. R. Mebane (2018). Active Learning Approaches for Labeling Text. *10012*.

Mimno, D., H. Wallach, E. Talley, M. Leenders, and A. McCallum (2011). Optimizing semantic coherence in topic models. *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (2), 262–272.

Mitchell, J. and M. Lapata (2010). Composition in Distributional Models of Semantics. *Cognitive Science 34*(8), 1388–1429.

Monroe, B. L., J. Pan, M. E. Roberts, M. Sen, and B. Sinclair (2015). No! Formal Theory, Causal Inference, and Big Data Are Not Contradictory Trends in Political Science. *PS: Political Science & Politics 48*(01), 71–74.

Moody, C. (2016). Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec.

Mozer, R., L. Miratrix, A. R. Kaufman, and L. J. Anastasopoulos (2019). Matching with Text Data: An Experimental Evaluation of Methods for Matching Documents and of Measuring Match Quality. *Political Analysis* (Forthcoming).

Neuendorf, K. A. (2002). *The Content Analysis Guidebook*. London: SAGE.

Pan, J. and K. Chen (2018). Concealing corruption: How Chinese officials distort upward reporting of online grievances. *American Political Science Review 112*(3), 602–620.

Pennington, J., R. Socher, and C. Manning (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Peters, M., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). Deep Contextualized Word Representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237.

Roberts, M. E., B. M. Stewart, and E. M. Airoldi (2016). A model of text for experimentation in the social sciences. *Journal of the American Statistical Association 111*(515), 1–49.

Roberts, M. E., B. M. Stewart, R. A. Nielsen, D. Blei, N. Egami, C. Felton, J. Fowler, J. Grimmer, E. Hartman, C. Hazlett, S. Hill, K. Imai, R. Johnson, G. King, A. Lo, W. Lowe, C. Lucas, I. Lundberg, W. Mebane, D. Mimno, J. Pan, M. Ratkovick, M. Salganik, C. Tolbert, and S. Zhang (2018). Adjusting for Confounding with Text Matching. *Working paper*.

Roberts, M. E., B. M. Stewart, and D. Tingley (2014). stm: R Package for Structural Topic Models. *Journal Of Statistical Software VV*(November 2017).

Roberts, M. E., B. M. Stewart, D. Tingley, C. Lucas, J. Leder-Luis, S. K. Gadarian, B. Albertson, and D. G. Rand (2014). Structural topic models for open-ended survey responses. *American Journal of Political Science 58*(4), 1064–1082.

Rong, X. (2014). word2vec Parameter Learning Explained. pp. 1–21.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence 1*(5), 206–215.

Sanford, L., M. E. Roberts, and K. Li (2019). Discovery of Influential Text in Experiments Using Deep Neural Networks. *Working paper*.

Spirling, A. and P. L. Rodriguez (2019). Word Embeddings What works, what doesn't, and how to tell the difference for applied research. *Working paper*.

Wilkerson, J. and A. Casas (2017). Large-Scale Computerized Text Analysis in Political Science: Opportunities and Challenges. *Annual Review of Political Science 20*(1), 529–544.

Ying, L., J. M. Montgomery, and B. M. Stewart (2019). Inferring Concepts from Topics : Towards Procedures for Validating Topics as Measures. *Working paper*.

Zhou, Z.-H. (2018). A brief introduction to weakly supervised learning. *National Science Review 5*, 44–53.