Word Embeddings
What works, what doesn't, and how to tell the difference for applied research

Pedro L. Rodriguez* Arthur Spirling†

Abstract

Word embeddings are becoming popular for political science research, yet we know little about their properties and performance. To help scholars seeking to use these techniques, we explore the effects of key parameter choices—including context window length, embedding vector dimensions and pre-trained vs locally fit variants—on the efficiency and quality of inferences possible with these models. Reassuringly we show that results are generally robust to such choices for political corpora of various sizes and in various languages. Beyond reporting extensive technical findings, we provide a novel crowdsourced "Turing test"-style method for examining the relative performance of any two models that produce substantive, text-based outputs. Our results are encouraging: popular, easily available pre-trained embeddings perform at a level close to—or surpassing—both human coders and more complicated locally-fit models. For completeness, we provide best practice advice for cases where local fitting is required.

9,469 words

*Postdoctoral Fellow, Data Science Institute (joint with Political Science), Vanderbilt University and Instituto de Estudios Superiores de Administración (pedro.rodriguez@Vanderbilt.Edu)

†Professor of Politics and Data Science, New York University (arthur.spirling@nyu.edu)

1 Introduction

The idea that words and documents can be usefully expressed as numerical objects is at the core of modern political methodology. The exact method one uses to model "text as data" has been debated. But in recent times, "word embeddings" have exploded in popularity. The premise of these techniques is beguilingly simple: a token of interest ("`welfare`" or "`washington`" or "`fear`") is represented as a dense, real-valued vector of numbers. The length of this vector corresponds to the nature and complexity of the multidimensional space in which we are seeking to "embed" the word. And the promise of these techniques is also simple: distances between such vectors are informative about the semantic similarity of the underlying concepts they connote for the corpus on which they were built. Applications abound. Prosaically, they may be helpful for a 'downstream' modeling task where we need better inputs to some supervised problem. More excitingly, the similarities may be substantively informative *per se*: if the distance between "`immigrants`" and "`hardworking`" is smaller for liberals than for conservatives, we learn something about their relative worldviews.

Exploiting the basic principles behind these examples, word embeddings have seen tremendous success as feature representations in well-known natural language processing problems. These include parts-of-speech tagging, named-entity-recognition, sentiment analysis and document retrieval. Given the generality of those tasks, it is unsurprising that word embeddings are rapidly making their way into the social sciences (e.g. Kozlowski, Taddy and Evans, 2018), political science being no exception (e.g. Rheault and Cochrane, 2019; Rodman, 2019). But as is often the case with the transfer of technology, there is a danger that adoption will outpace understanding. Specifically, we mean comprehension of how well the technology performs—technically and substantively—on specific problems of interest in the domain area of concern. The goal of this paper is to provide that understanding for political science, enabling practitioners to make informed choices when using these approaches.

As conveyed in our examples above, word embeddings serve two purposes. First they have an instrumental function, as feature representations for some other learning task. Second, embeddings

are a direct object of interest for studying word usage and meaning—i.e. human semantics. Good performance in the former need not correlate with good performance in the latter (Chiu, Korhonen and Pyysalo, 2016). In this paper we focus on this second purpose: embeddings as measures of meaning. The reasoning is simple. First, we cannot pretend to foresee all the downstream use cases to which political scientists will apply embeddings. Moreover, given a well-defined downstream task, how to think about performance is trivial—these are usually *supervised* tasks with attendance metrics measuring accuracy, precision and recall. Second, word usage, including differences between groups and changes over time, is of direct and profound interest to political scientists, but validating such quantities automatically is challenging (Faruqui et al., 2016).

With this in mind, our specific contribution goes beyond a useful series of results. We propose the framework used to generate them, that will guide researchers through the maze of choices that accompany word embeddings. These include whether to use cheap pre-trained or (more) expensive "local" corpus trained embeddings. And, within models, we demonstrate the effects of altering core hyperparameters such as context *window size* and *embedding dimensions*. In addition to standard predictive performance and computational cost metrics though, we present two novel approaches to model comparison and validation. First, framing the task as an information retrieval one, we show how models may be mechanically compared in terms of the words they place close to others—including politics-specific tokens. As a second "gold-standard" approach, we propose a new take on the classical "Turing test" wherein human judges must choose between computer generated nearest neighbors and human generated nearest neighbors. While we necessarily make certain choices in terms of embedding architecture and which hyperparameters to focus on, we stress the framework developed is completely general and not beholden to these choices. It is easily adaptable to evaluate new models—including non-embedding models of human semantics—and other parameter variations.

Our findings are reassuring. In particular, (cheap, readily available) pre-trained embeddings perform extremely well on a multitude of metrics relative to human coding and (more expensive) locally trained models for political science problems. This is true beyond our focus *Congressional*

Record corpus, and extends even to non-English collections. Separate to our intellectual contribution, we also provide the full set of all local models we fit—250 in all—so practitioners can use them "off-the-shelf" in their own work along with two novel corpora of parliamentary speeches in Spanish and German.

In the next section we clarify terms and provide some background on origins of word embeddings. We then lay out the choices practitioners face, before discussing evaluation methods and how we implement them. Subsequently, we extend our work to a variety of corpora, and we then summarize the main takeaways for researchers.

2 Word Embeddings in Context

To fix ideas, consider two popular problems—and solutions—in political science:

1. estimating *political attention*; specifically, the focus that legislators place on different aspects of policy at different times.

2. characterizing the *nature of political conflict*; specifically, the way in which politicians of different parties use tellingly different words to discuss similar issues

In the case of (1), topic models have proved extremely popular (e.g. Quinn et al., 2010). In the case of (2), various supervised machine-learning efforts have emerged (e.g. Diermeier et al., 2012; Monroe, Colaresi and Quinn, 2008). With some exceptions, analysts have not attacked these problems with *embeddings* approaches, which differ from current practice both conceptually and practically.

At a high level, these differences boil down to a particular implementation of the *distributional hypothesis*—the belief that we may "know a word by the company it keeps" (Firth, 1957, 11). That is, that a word's meaning may be learned from its context. In the case of topic models fit via Latent Dirichlet Allocation (LDA) (Blei, Ng and Jordan, 2003), this insight is obtained via the co-occurrence of terms with the word of interest in a *document*. In the case of embedding models, it is operationalized as the other words that appear near it in text—literally in a narrow window around the word of interest, looking over the corpus as a whole.

An immediate practical consequence of this difference is that the way text data is represented for the models differs fundamentally for embeddings approaches. In the case of both (1) and (2) above, the standard arrangement would be to represent each document as a vector, with each element of that vector a (possibly reweighted) set of frequencies of the terms it contains (see e.g. Denny and Spirling, 2018, on "preprocessing"). But this "bag of words" approach is not how we proceed in the case of embeddings. Specifically, our texts are now represented as ordered sequences: the occurrence of a given word(s) is used to predict the occurrence of the next one(s). There are many approaches and "architectures" and they define the windows (the "context") around the word in different ways. But they generally all proceed by using an (artificial) neural network that maps words to real-valued vectors: indeed, the embeddings are the coefficients of that neural network model. Intuitively, each element of those vectors represents some hypothetical characteristic of the word(s). The vector for a given word can be called a *word embedding*. Scholars (e.g. Collobert and Weston, 2008) have shown that embeddings capture substantive syntactic and semantic information about language *per se* that can be explored using basic algebraic operations. A textbook example is using embeddings for analogical reasoning like `king - man + woman` $\approx$ `queen`, where each word in the equation is the vector that represents it. Clearly, what emerges from an embedding model—a real valued vector for every word in the corpus—is very different to the outputs of the traditional approaches above. There, respectively, we obtain a distribution over terms in the vocabulary of the corpus (a "topic"), or a set of words that maximally discriminate between the corpora produced by one party versus another. In neither case do we obtain estimates of semantic relationships between words at the word level.[1]

Given that the inputs, the model, and the outputs are so very different in the case of embeddings, it is unsurprising that the decisions one needs to make when using an embeddings approach are different. Everything that follows is about those decisions: both how they should be made generally, and what lessons may be learned when they are applied to representative political science

[1]While these may be retrieved from the output of topic models, this is not their intended use case.

corpora specifically. Broadly these decisions concern:

1. how large a **window size** we want the model to use. This is a decision about what we mean by "context": one word either side of our target? six words? twelve words? etc

2. how large an **embedding** to use to represent the words. This is a decision about how complex a model we wish to fit—literally, the number of dimensions (from say, 50 to 450) our word vectors should ultimately have. Too many dimensions, and we may be modeling noise. Too few dimensions, and we risk not capturing important subtleties in meaning.

3. whether to fit the embedding models **locally**, or to use **pre-trained** embeddings fit to some other (ideally related) corpus. This is a decision about whether our texts are sufficiently similar to other corpora for which embeddings have already been learned. In particular, pre-trained embeddings are typically based on the relationships between words in Wikipedia and other large-scale collections. They are computationally cheap (and can be downloaded in minutes), but may be inaccurate for our specific problem. In which case, we may wish to locally train: that is, fit an embeddings model from scratch to our corpus.

4. the specific **modeling approach** to use—where the choice for most practitioners is `GloVe` (Pennington, Socher and Manning, 2014) or `Word2Vec` (Mikolov et al., 2013). We will give more details below, but these are similar mathematically, differing primarily in the way they use co-occurrences of terms in the documents as a whole.

To guide readers more familiar with traditional approaches in political science, Table 1 provides an overview of how embeddings differ from those methods—in terms of structure, models, inputs and so on. In the final row of the table, we note the various approaches have issues with "stability". In the case of embeddings, the issue is that vectors learned on the same corpus even with the same technique for the same word, need not be identical for every run of the model. This is partly about inevitable stochastic behavior of algorithms in this field, but also connected with fundamental sampling uncertainty in (small) finite corpora (Antoniak and Mimno, 2018). Our

maintained assumption in what follows is that researchers would prefer more stable to less stable outputs, but would always like to know how much stability they could expect in practice.

	traditional Unsupervised	**traditional Supervised**	**Embeddings**
Bag of Words	yes	yes	no
Example Models	Latent Dirichlet Allocation; Structural Topic Model	Support Vector Machines; Random forest	`GloVe`, `Word2Vec`
Citations/Applications	Quinn et al. (2010); Roberts et al. (2014)	Diermeier et al. (2012); Montgomery and Olivella (2018)	Rodman (2019); Rheault and Cochrane (2019)
Inputs	document-term matrix	document-term matrix; labeled y	term-co-occurence matrix
Outputs	document distribution over topics; topic distribution over words	term importance matrix (for class prediction)	word vectors
Example User Decisions	weighting of tokens; number of topics	training/test split; weighting of tokens; number of trees (RF); number of variables at each split (RF); prior class probabilities	pre-trained or local fit; window-size; embedding dimensions; weighting of tokens
Stability Concerns	multiple modes	sensitivity to training/test set; labeling errors	algorithmic; corpus characteristics

Table 1: Comparing Traditional Approaches with Embeddings

What substantive light would embeddings, perhaps fit to the *Congressional Record*, shed on the problems we alluded to above? In either case, embeddings tell us how a word is associated with others. For the first problem, of "attention" estimation, embeddings would help us locate sets of documents/speeches that are about similar themes (e.g. Hamilton, Leskovec and Jurafsky, 2016), but also to understand how those themes and the words they contain evolve over time (e.g. Rodman, 2019). In terms of the second "conflict" problem, we could imagine learning embeddings for both Republican and Democrat Senators for particular words—say, `abortion` or `welfare`. The way that those embeddings differ—literally, the words those words are closest to for the partisans in each group—would presumably be informative about the nature of debate in the contemporary US Congress (see Rudolph et al., 2017; Rheault and Cochrane, 2019; Mebane Jr et al., 2018, for related applications on partisanship). Beyond being of interest *per se*, the embeddings could

obviously be used as inputs to other downstream tasks (see e.g. Zhang and Pan, 2019, for an example in sociology): for example, we might find that knowing the embedding representation of a bill is helpful for predicting whether parliamentarians are likely to vote for it or not.

To make matters concrete below, we will use the "partisan conflict" case as a running example. We will assume that the analyst wants to learn embeddings (vectors) for a set of political terms, but does not know *a priori* which modeling approach is optimal—in terms of run-time, quality of fit to their data and so on. Importantly, we also assume that the researcher does not know the "accuracy" (in an intrinsic, substantive sense) of the representations they might learn. To clarify, suppose the model informs the researcher that, for Republicans, the terms closest to `welfare` are `dependency` and `reform`. Crudely, is this a "good" embedding? Does it reflect judgements that human researchers (an assumed gold standard) would make about the context for this word? Assessing this in a general way will be one of our primary contributions. Before we get there, we must deal with the series of choices we outlined above. As a road map, in Figure 1, we give flowchart of decisions that a typical user (including our "partisan conflict" researcher) would need to make in learning embeddings for their corpus. The terms in bold will be our focus.

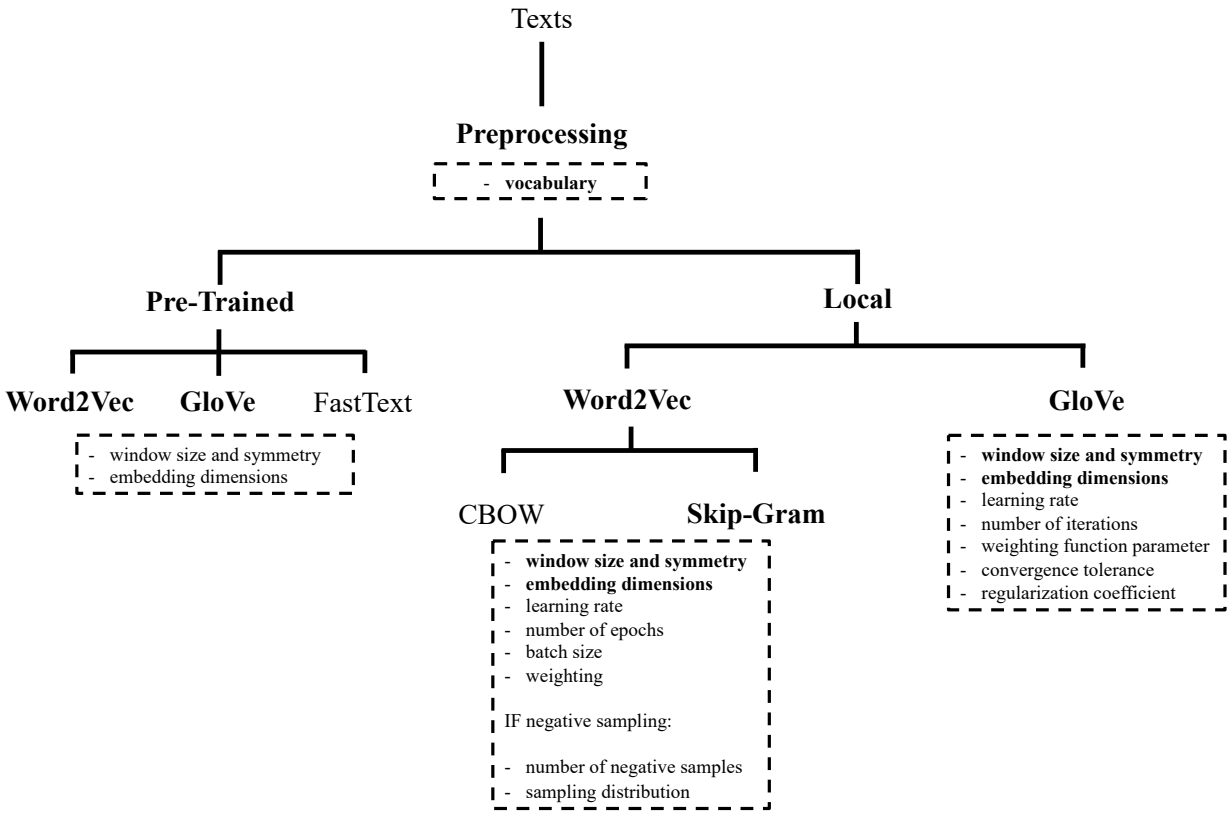


Figure 1: Decision Flowchart for Embeddings. Branches are either/or approaches. Boxes list decisions to be made under those approaches. Decisions in **bold** are ones we evaluate below

Keeping with the spirit of our running example, we will be interested in modeling the *Congressional Record*; specifically, the set of transcripts for the 102nd–111th Congresses (Gentzkow, Shapiro and Taddy, 2018)—a medium sized corpus of around 1.4 million documents. These contain all text spoken on the floor of both chambers of Congress. We restrict our corpus to the set of speeches for which party information is available. We do minimal preprocessing: remove all non-text characters and lower case. Next we subset the vocabulary. We follow standard practice which is to include all words with a minimum count above a given threshold—we choose 10. This yields a vocabulary of $91,856$ words.[2] And we focus on `GloVe` embeddings simply because it seems more popular with social scientists.[3] We will nonetheless also provide commentary on taking

[2]The pre-trained `GloVe` vocabulary consists of $400,000$ tokens.

[3]In particular, the `GloVe` pre-trained with window size 6 and embedding dimensions 300, avail-

embeddings to other languages and using `Word2Vec`—the other popular model in this literature.

3 Evaluating Embedding Models for Social Science

To evaluate the effects of the choices, we need tasks. In the word embeddings literature (Schnabel et al., 2015), they fall into two categories: *extrinsic* and *intrinsic*. Extrinsic tasks are those in which researchers use embeddings as input variables to perform some other (typically, supervised learning) task which has pre-specified "correct" answers. One chooses the optimal embedding specification by varying the hyperparameters of the model, and then examining the changes to accuracy and other performance statistics that occur when doing this task. Our position is that these tasks are rare and highly specific in political science (Denny and Spirling, 2018): certainly we could not think of an obvious one for the *Congressional Record*. An intrinsic task is one in which we assess the worth of particular embeddings by how well they represent "true" relationships between words. In computer science, scholars proceed by e.g. comparing the analogies implied by embeddings (using the rough calculus we mentioned above) to a pre-existing (and commonly known) bank of such relationships between words (e.g. Mikolov et al., 2013). In principle, a politics researcher could create such a database for Congressional speeches. In line with the vector calculus mentioned above for contemporary times, we would want the following embedding equation for leadership to "work" (with closer approximations being better): `Pelosi` - `House` + `Senate` $\approx$ `McConnell`. We cannot think of many such examples, however. Moreover, these obviously do not travel: these relationships do not even make sense for the House of Commons/Lords in the UK (or equivalent in Canada), let alone for other areas of the discipline. While we will still focus on intrinsic tasks—i.e. "real life" uses of words—we will take a more "human" approach to evaluate embedding quality.

able on February 2, 2019 from `https://nlp.stanford.edu/projects/glove/`, for which the training corpus is Wikipedia 2014 and Gigaword 5.

3.1 Turing Assessment

To evaluate semantic coherence we draw inspiration from the principles laid out by Turing (1950) in his article on computer intelligence. There, a machine showed human-like abilities if a person engaging in conversation with both a computer and a human could not tell which was which. We use that basic intuition. In particular, an embedding model achieves "human" performance if human judges—crowd workers—cannot distinguish between the output produced by such a model from that produced by independent human coders. In our case, the idea is not to "fool" the humans, but rather to have them assert a preference for one set of outputs over another. If a set of human judges are on average indifferent between the human responses to a prompt and the model's responses, we say we have achieved human performance with the model. By extension, a model can achieve *better than human* performance by being on average preferred by coders. Naturally, models may be *worse than human* if the judges like the human output better.

Notice that while the traditional Turing test connotes a human versus machine contest, the approach here is more general. Indeed, any output can be compared to any other—including where both sets are produced by a model or both by humans—and conclusions drawn about their relative performance as judged by humans. In contrast with other intrinsic evaluation metrics found in the literature, our proposed assessment can incorporate both absolute and comparative evaluations.

The steps we take to assess the relative Turing performance of the models are:

1. **Human generated nearest neighbors:** For each of ten political prompt words (described below) have humans—crowd workers on Amazon MTurk—produce a set of nearest ten neighbors—we have 100 humans perform this task. Subsequently rank "human" nearest neighbors for each prompt in terms of the number of mentions and choose the top 10 for each prompt.

2. **Machine generated nearest neighbors:** For the embedding model under consideration—pre-trained or some variant of the locally fit set up—produce a list of ten nearest neighbors

for each of the ten given prompt words above.[4]

3. **Human rating:** Have a separate group of humans perform a triad task —135 subjects on average for each model comparison—wherein they are given a prompt word along with two nearest neighbors—a computer and a human generated nearest neighbor—and are asked to choose which nearest neighbor they consider better fits the definition of a context word. See Supporting Information C for task appearance/wording.

4. **Compute metric:** For each prompt compute the expected probability of the machine generated nearest neighbor—our *candidate model*—being selected vis-a-vis a *baseline model*—humans in our gold-standard. We divide this number by 0.5, as such the index will range between 0 and 2. A value of 1 implies the machine is on par with human performance (i.e. a human rater is equally likely to choose a nearest neighbor generated by the embedding model as one generated by another human) while a value larger (smaller) than 1 implies the machine performs better (worse) than humans.

We give specific details on how we handle the crowdworkers themselves, and the comparisons they produce, in Supporting Information D.

Our approach is predicated on our human coders being able to make reasonable judgments about contexts in the way we described, which seems plausible based on other work (Benoit et al., 2016). Notice that for our comparisons to be useful, we do not require that humans are error-free in their judgements: any applied coding problem will involve some mistakes. Rather, we require that, absent time or budget constraints, human coding would be the best possible approach. And thus to the extent an embedding model can replicate it, that model is of value.

If one believes the crowdworkers differ systematically in their understanding of terms from the "population" underlying the corpus (however defined), one might use covariate information to stratify. That is, one might re-weight responses in a way that better reflects the population of interest. Related is the idea that the crowdworkers might differ in their responses to embeddings

[4]It is common in the literature to focus on the *top ten* nearest neighbors.

as a function of their own covariate profiles (e.g. whether they are male or female, Democrat or Republican etc). However, in what follows, we will not use covariate information on our crowdworkers. The reason for this is connected to both research design and ethics. On research design, note that our crowdworkers are being treated as *de facto* research assistants: they are helping us study embeddings; embeddings are not helping us study our research assistants. Thus, regarding ethics, this is *not* human subjects research and we have not obtained permissions for the same.

In reference to our running example, we could imagine a traditional approach using research assistants to read Democrat and Republican speeches to identify the different ways they talk of the same issue. These assistants would be required to find the context for terms—literally, what they connote. Here, at a high level, our embedding model is suggesting possible contexts. If humans "like" those suggestions (relative to those suggested by crowd equivalents of research assistants), we have evidence that the models work for this purpose. But to reiterate, we as researchers do not believe model performance—on human or other metrics (below)—will vary by party, so we will focus on aggregate performance in what follows.

In practice, we push our comparison beyond mere binary preferences. Specifically, we also investigate how similar (via log rank deviation) the human output vs embedding output is in terms of the set of nearest neighbors they produce. This gives us a finer-grained sense of how inferences from humans or models would differ in practice.

3.2 Other Metrics

The foregoing section set out how to assess whether embeddings "make sense". But we also consider other practical issues that applied researchers might have. In particular, we will report results related to:

1 technical criteria—model loss and computation time;

2 query search ranking correlation—across-model (varying hyperparameters) Pearson and rank correlations of nearest neighbor rankings;

3 model variance (stability)—within-model (holding hyperparameters constant) Pearson correlation of nearest neighbor rankings across multiple initializations.

For (1), we mean how well different models fit our Congressional corpus according to their internal optimization procedures (Pennington, Socher and Manning, 2014) and then literally how long it takes each model to fit (in minutes). We assume researchers would like to know how time-consuming such procedures are for an example problem. For (2), we mean the correlation between the outputs of one embedding model specification and another. If these are high across many different choice combinations, there is a sense in which it does not much matter what the applied researcher chooses. Finally, for (3), we measure how different the results are for the *same* model run multiple times. Intuitively, if this is high for a given specification (i.e. the correlation is low) there is a danger that researchers will update from an "unusual" run, the general output of which would not be replicated by others using the same code and data. We privilege this concern over what we believe to be a conceivable but limited benefit of different runs revealing different relationships in the data—some of which may be particularly useful. In any case, we suspect users will want to know how much variance they can expect in practice.

4 Estimation Setup

We focus our analysis on two hyperparameter choices—five values of each for 25 combinations in all—though of course the framework we lay out is not specific to these parameter pairs:

1. window-size—1, 6, 12, 24 and 48 and

2. embedding dimension —50, 100, 200, 300, 450

To account for estimation-related instability we estimate 10 sets of embeddings for each hyperparameter pair, each with a different randomly drawn set of initial word vectors. In total we estimate 250 different sets of embeddings. The only other hyperparameter choices we make and

leave fixed are the *number of iterations* and *convergence threshold*. For each model we set the maximum number of iterations to 100 and a use a convergence threshold of 0.001 such that training stops if either the maximum number of iterations is reached or the change in model loss between the current and preceding iterations is below the convergence threshold. None of our models reached the maximum number of iterations before meeting the convergence threshold. We set all remaining hyperparameter values at their default or suggested values in the `Glove` software.[5] The set of locally trained models—available on the project's GitHub—represents a contribution in and of itself that will hopefully save researchers time and computational resources.

4.1 Query Selection

We explained that a natural quantity of interest is the set of nearest neighbors of a given word—a query—in the embeddings space. These form the core of our comparison metric in the sense that we will want to know how similar one set of nearest neighbors from one model specification is to another. And, by extension, how "good" one set of nearest neighbors is relative to another in terms of a quality evaluation by human judges. We use two sets of queries: a random sample of 100 words from the common vocabulary and a set of 10 curated political terms.

First among our curated political terms, there are a series of concept words that we suspected would be both easily understood, but also exhibit different meanings depending on who is asked: `democracy`, `freedom`, `equality`, `justice`. Second, there are words pertaining to policy issues that are debated by political parties and motivate voting: `immigration`, `abortion`, `welfare`, `taxes`. Finally, we used the names of the major parties, which we anticipated would produce very different responses depending on partisan identification: `republican`, `democrat` (see Halpern and Rodriguez, 2018). Obviously, we could have made other choices. And indeed, we would encourage other researchers to do exactly that. Our queries are intended to be indicative of what we expect broader findings to look like, and to demonstrate the utility of our generic approach.

[5] We use the `text2vec` R package to run all our models.

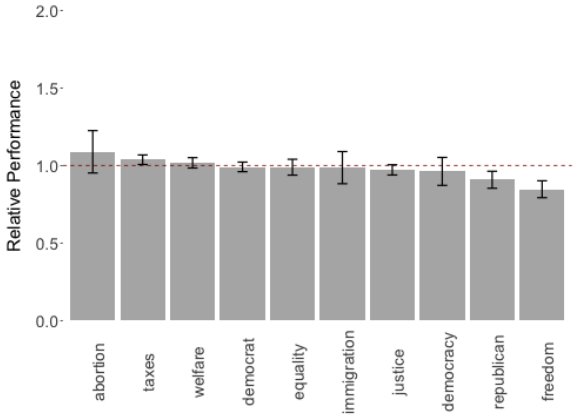
5 Results: Performance Compared

Now we report the results for the evaluation metrics outlined in Section 3. We begin with the "human" performance of embeddings. To reiterate, our presumption is that we would like embedding models to produce nearest neighbors that accord with human priors—and that thus might otherwise have been suggested by research assistants. As we will see, the good news is that they come close, and sometimes do better.

5.1 Turing Assessment

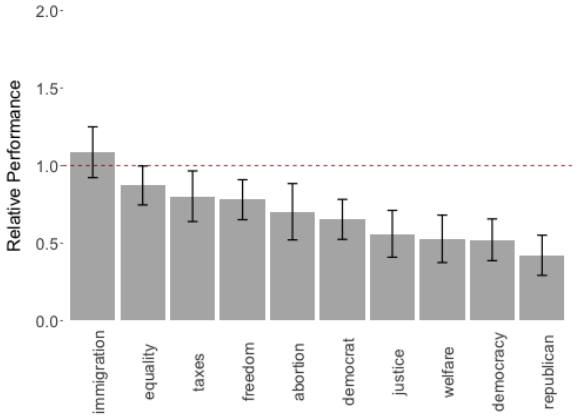
Figures 2a– 2d measure performance of a "candidate" model relative to a "baseline" model. Here, values above (below) 1 mean nearest neighbors from the "candidate" model were more (less) likely to be chosen by human raters. A value of 1 means human raters were on average indifferent between the two models. Figure 2a compares two local models: $48-300$ (candidate) and $6-300$ (baseline). There is no unqualified winner and, in fact, these models have a 0.92 correlation (see Figure 5b). If nothing else, this suggests that—from a human perspective—different hyperparameter choices, at least around common practice (window of 6, vector length 300) do not matter much.

How do local models fare directly against human generated nearest neighbors? Except for one query (`immigration`), the local model of choice—6-300—shows *below-human* performance for all but two of the queries. On average, for the set of ten political queries, the local model achieves 69% (std devn= 0.20) of human performance. Turning to pre-trained `GloVe` embeddings, we observe that they are generally preferred to locally trained embeddings (see Figure 2c). Moreover, pre-trained embeddings are more competitive against humans—albeit with greater variance—achieving an average of 86% (std dev $= 0.23$) of human performance (see Figure 2d). That is, a researcher using pre-trained embeddings for this corpus (available in minutes, at no cost) is able to produce results very similar to those that would have been produced by (relatively slow, costly) research assistants. This is comforting news.



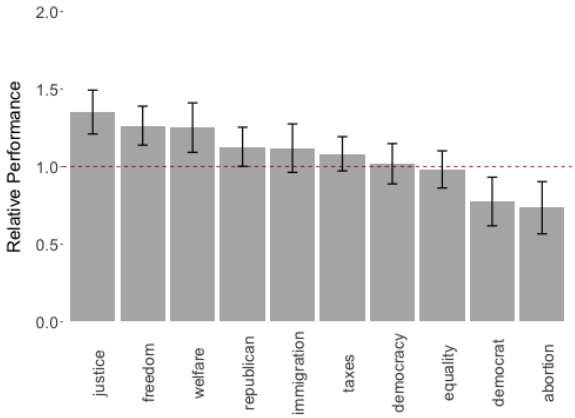
(a) Candidate: Local 48-300

Baseline: Local 6-300



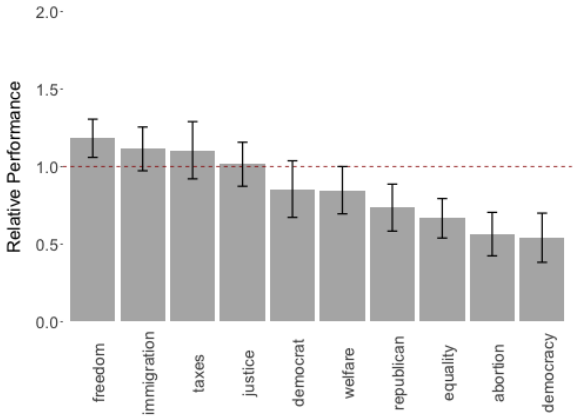
(b) Candidate: Local 6-300

Baseline: Human



(c) Candidate: `GloVe`

Baseline: Local 6-300



(d) Candidate: `GloVe`

Baseline: Human

Figure 2: Human Preferences-Turing Assessment

Using the log rank deviation measure, we can compare all models given our set of human generated lists (see Figure 3). We find that models with larger windows and more dimensions show lower log rank deviations, indicating better performance but with decreasing returns. Put simply, this means while "bigger" models will get the researcher results closer to those that human coders would produce, so long as users do not opt for the very simplest models (very small windows, short vectors) they can obtain relatively excellent performance with moderately complex ones.

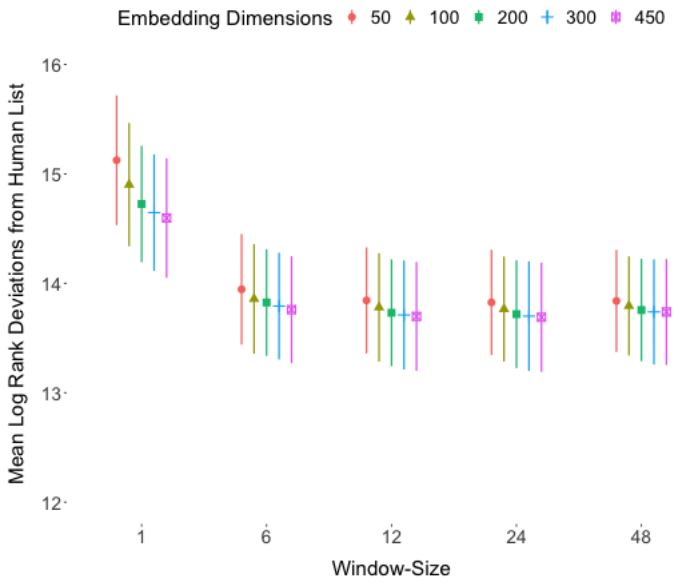


Figure 3: Human Preferences-Log Rank Deviations: complex models come closer to 'human' assessments, but medium size models are almost as good as very large ones.

These aggregate results are encouraging, but our experiments can also shed light on exactly where—i.e. on what types of queries—researchers might expect to do better or worse in terms of their model selections. First, we note that human coders respond (relatively) poorly to the pre-trained outputs for `abortion` and `Democrat` as compared to those from the locally trained model (6-300). We give those nearest neighbors in Table 2, in the first two groups (first four columns). Digging deeper, we found that human coders particularly liked "lateterm" and "partial" when they were offered them—context words that were not even selected by the pre-trained model. In the case, of "Democrat", the human coders liked "liberal" when it was offered. Exactly *why* human coders preferred these is debatable, but our observation would be that these accord with more "everyday" uses of these terms.

"abortion"		"democrat"		"justice"	
`pre-trained`	`local`	`pre-trained`	`local`	`pre-trained`	`local`
abortions	abortions	senator	republican	supreme	rehnquist
contraception	partialbirth	democratic	democratic	justices	scalia
euthanasia	lateterm	republican	democrats	judicial	owens
antiabortion	procedure	democrats	republicans	court	ginsburg
legalized	ban	sen.	party	judge	court
homosexuality	partial	rep.	liberal	criminal	souter
opposes	clincs	congressman	majority	law	oconnor
pro-life	sterilization	senate	conservative	attorney	brennan
pregnancy	clinic	incumbent	side	appeals	department
advocates	birth	gop	ranking	scalia	supreme

Table 2: Where pre-trained does better or worse than local. First two groups: words ("abortion", "democrat") for which crowdworkers preferred local model nearest neighbors (right columns) to pre-trained model ones (left columns) for `GloVe`. Last group ("justice"): crowdworkers prefer the pre-trained model.

Our investigation on where the *local* model does worse bears out this hunch. The final column of Table 2 reports the nearest neighbors for "justice". As can be readily seen, the local model mostly returns a list of Supreme Court Justices (unsurprisingly, given this is based on discussions in the *Congressional Record*). By contrast, crowdworkers liked "law" and "court" when it was offered. One lesson here, and one we will comment on below, is that crowdworkers prefer terms in general use over and above those that require very specific domain knowledge. We suspect this is true of research assistants too.

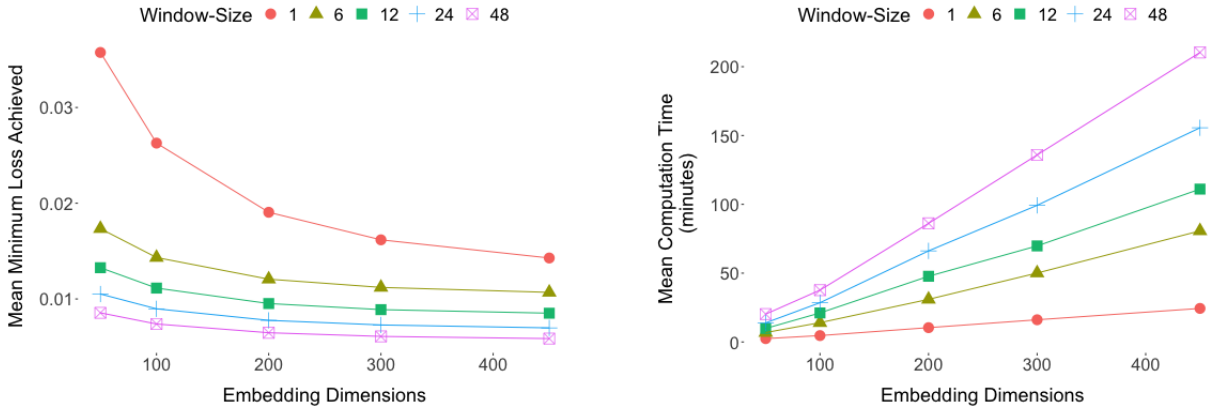
5.2 Other metrics

Figure 4a displays the mean—over all ten initializations—minimum loss achieved for sixteen (of the twenty-five) parameter pairs we considered.[6] Unsurprisingly, more dimensions and larger window-sizes both unconditionally improve model fit albeit with decreasing returns in both parameter choices. Except for very small window-sizes (< 6), improvements become marginal after around 300 dimensions. If we take loss seriously, then researchers ought avoid combining few dimensions (< 100) with small window-sizes (< 6). There are two caveats, however. First, models with different window sizes represent qualitatively different notions of context, and presumably the match between that and the substantive problem at hand is more important than comparing relative fit. With reference to our running example, what we mean by "context" for partisan difference should presumably be assessed by the qualitative meaningfulness of the comparisons, not simply an agnostic model loss.

Second, unsurprisingly, bigger models take longer to fit (see Figure 4b). The largest here ($48 - 450$) took over three hours to compute parallelizing over eight cores. This seems reasonable if only computing once, but can become prohibitive when computing over several initializations as we suggest. In this light, the popular parameter setting $6 - 300$ (window size 6, embedding dimensions 300) provides a reasonable balance between performance and computation time.

Different parameter choices produce different results in terms of performance, but what do these differences mean substantively? To answer this, we compare models with respect to how they rank query searches. Figure 5a displays a heatmap of pairwise correlations for all models, including `GloVe` pre-trained embeddings, for the set of random queries. We observe high positive correlations (> 0.5) between all local models. Correlations are generally higher between models of the same window-size, an intuitive result, as they share the underlying co-occurrence statistics. Somewhat less intuitive, comparing models with different window-sizes, correlations are higher

[6] We plotted sixteen of the twenty-five parameter pairs to avoid clutter. The left-out parameter pairs follow the same trend.



(a) Mean Minimum Loss Achieved

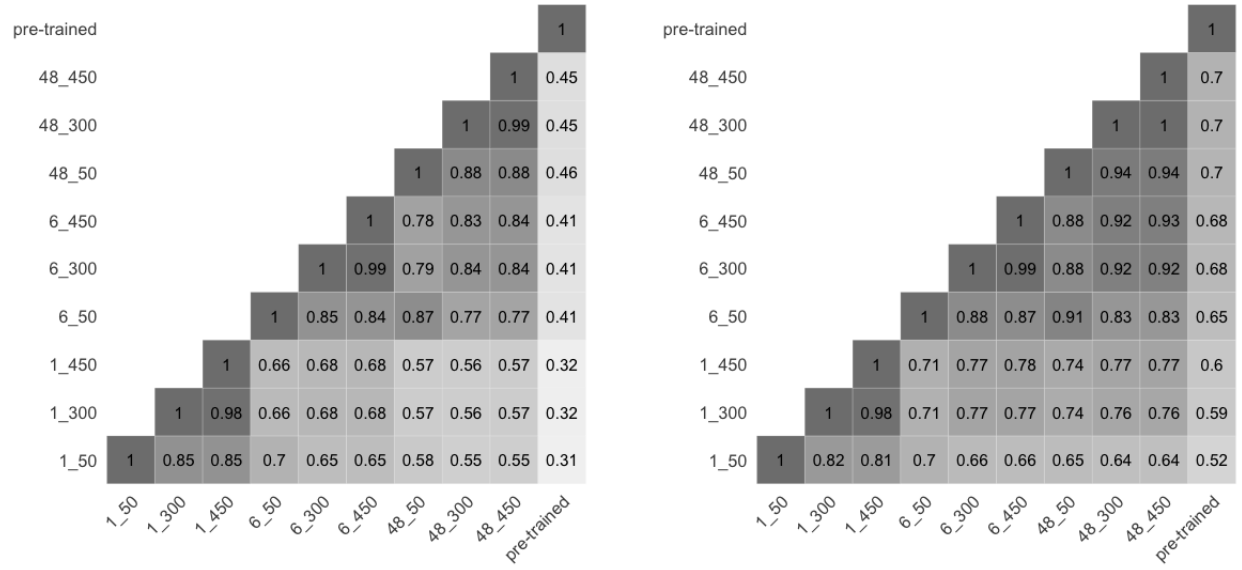
(b) Computation Time (minutes)

Figure 4: Technical Criteria: larger models fit better, but take longer to compute.

the larger the window-size of the models being compared (e.g. 6 and 48 vis-a-vis 1 and 6). Correlations are larger across the board for the set of political queries (see Figure 5b). These results suggest the embeddings for the *Congressional Record* are sensitive to window-size but this decreases quickly as we go beyond very small window-sizes (i.e. models with window-size of 6 and 48 show much higher correlation than models with window-size of 1 and 6).

The last column of Figures 5a and 5b compare `GloVe` pre-trained embeddings with the set of local models. For this comparison we subsetted the respective vocabularies to only include terms common to both the local models and the pre-trained embeddings. As would be expected, correlations are lower than those between local models, yet they are still surprisingly large—especially for local models with larger window-sizes and for the set of political queries (all above 0.5). Our reading is that `GloVe` pre-trained embeddings, even without any modifications (Khodak et al., 2018), may be a suitable alternative to estimating locally trained embeddings on present-day political corpora. This is good news for political scientists who have already relied on pre-trained embeddings in their work.

As a final check, and in keeping with the nature of our running example, we looked at whether pre-trained embeddings might do a 'worse' job of reflecting highly specific local embeddings for our focus corpus. In this case, we mean party: it could in principle be the case that while pre-trained



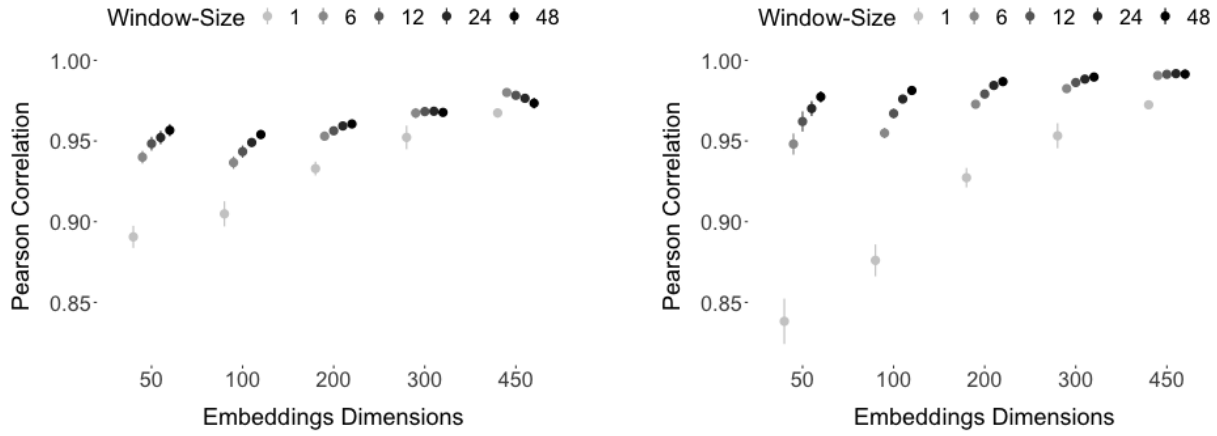
(a) Random Queries

(b) Politics Queries

Figure 5: Query Search Ranking Criteria: different models produce similar embeddings, especially for substantive "politics" queries.

embeddings do well in aggregate for the *Congressional Record* they do poorly for Democrats or Republicans specifically. To evaluate this we estimate a set of additional local models (again, 10 for each group and using 6-300 as parameter settings) for subsets—by party—of the aggregate corpus. We find no statistically significant differences in correlations (see Supporting Information E).

Finally, we report our results on stability. Figures 6a plots the distribution of Pearson correlations for the 100 random queries. Correlations are high—above 0.85—across the board, suggesting `GloVe` is overall rather stable—i.e. the organization of the embeddings space does not vary dramatically over different initializations. Still, models with larger window-sizes produce, on average, more stable estimates. As the number of dimensions increase, the difference in stability between different window sizes decreases and eventually flips—larger window sizes result in greater instability. This parabolic relationship between window-size, number of dimensions and stability is likely a function of corpus size—larger more generic corpora will require a greater number of dimensions to allow for multiple word senses—and token frequency—infrequent tokens are likely



(a) Random Queries

(b) Politics Queries

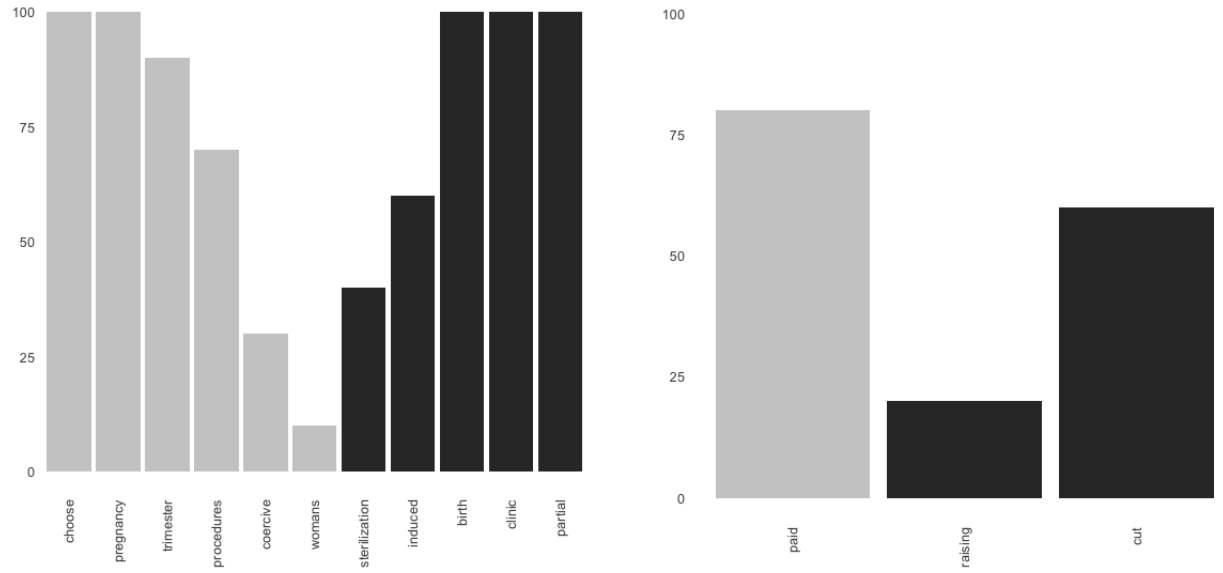
Figure 6: Stability Criteria: larger models are more stable, but with decreasing returns

to be more unstable.[7] For the set of 10 politics queries we observe the same trends although they do not reach the point at which the relationship reverses (see Figure 6b).

5.3 Our running example: partisan conflict results

To complete our main body of results, we return to our "partisan conflict" example. Do we have evidence that Republicans and Democrats differ in their understanding of various political terms? The answer to this is a qualified "yes". Given a query and any two models, we can identify the set of nearest neighbors among the top 10 that are unique to each model. Following our setup above we estimate 10 models for each party—10 different initializations—giving us 100 pairwise comparisons that we use to identify the nearest neighbors most common to each party. Figure 7 captures the aggregate of this comparison for two of our prompts, `abortion`—showing the least overlap—and `taxes` —showing the most overlap. A value of 100 indicates that nearest neighbor was unique to one party across all 100 pairwise comparisons. Moreover, the fewer the number of terms on the *x*-axis, the greater the overlap across parties—i.e. few terms were unique in any

[7]For the State of the Union corpus, a much smaller corpus, we find the flip occurs after 100 dimensions (see Supporting Information F and G).



(a) Abortion (b) Taxes

Figure 7: Nearest neighbors most common to each party. Democrat = light grey; Republican = dark grey.

pairwise comparison.

Our conclusion is that embeddings can tell us which words partisans "disagree" on. And thus embeddings would indeed be informative about conflict. But this is not our central focus here; instead we now move to understanding whether the *general* lessons we take from the *Congressional Record* apply to other corpora that are similarly based on (mostly pre-prepared) elite-level speech.

6 Other Corpora, Other Languages

We extend our evaluation to four other corpora, varying in size and language. The corpora are:

1. the full set of speeches from the UK Parliament for the period 1935 – 2016 obtained from Rheault et al. (2016);

2. all State of the Union (SOTU) speeches between 1790 and 2018;

3. the full set of speeches from both chambers of the Spanish Legislature —*Cortes Gen-*

erales— for the V - XII legislatures.[8] As political queries we use: `democracia`, `libertad`, `igualdad`, `equidad`, `justicia`, `inmigracion`, `aborto`, `impuestos`, `monarquia`, `parlamento`.

4. the full set of speeches from the German Legislature—*Deutscher Bundestag*— for the election periods 14 - 19.[9] The political queries in this case are: `demokratie`, `freiheit`, `gleichberechtigung`, `gerechtigkeit`, `einwanderung`, `abtreibung`, `steuern`, `cdu` and `spd`.

We did not find readily available `GloVe` pre-trained embeddings in German, as such all our comparisons in this case are between locally trained embeddings. Both the Spanish and German corpora are original datasets collected for the purposes of this paper.[10] Summary statistics for these corpora may be found in Supporting Information F, but suffice it to say that the SOTU corpus is substantially smaller than all the other corpora and also encompasses a much longer time period.

In Supporting Information G, we provide the same results plots for each of the above corpora as we gave for our *Congressional Record* corpus. Perhaps surprisingly, but no doubt reassuringly, these are almost identical to the ones above. That is, when we look at the embedding models we fit to these very different corpora, the lessons we learn in terms of hyperparameter choices, stability and correlations across search queries (i.e. on the issue of whether to fit local embeddings, or to use pre-trained ones) are the same as before. Of course, there are some exceptions: for example, we do find models of window-size equal to one perform well in the case of the SOTU corpus and for the German corpus—though to a lesser extent.

[8] As the XII was ongoing at the time of writing we used all speeches available up until Oct-18, 2018.

[9] As the 19$^{\text{th}}$ *Wahlperiode* was ongoing at the time of writing we used all speeches available up until Oct-18, 2018.

[10] We have made these publicly available, and these may be downloaded via the project's github page.

7 GloVe vs Word2Vec: some differences

In contrast to GloVe, which approximates global co-occurrence counts, Word2Vec follows an *online learning* approach—the model is progressively trained as we move the context window along the corpus. Word2Vec at no point sees the global co-occurrence counts. Despite this difference, Pennington, Socher and Manning (2014), the authors of GloVe, show that GloVe and Word2Vec's skip-gram architecture are mathematically similar. We might then conclude that they produce similar embeddings when trained on the same corpus. We find this is not the case, though human raters do not seem to judge one or other as being better.

In Supporting Information H we explain how we compared the two algorithms, and give extensive results discussion. The main points are these: first, for Word2Vec (unlike with GloVe) pre-trained embeddings exhibit much lower correlations with the set of local models.

Second, the correlation between both algorithms is never particularly high (for our set of parameter values). Our explanation for this difference is rooted in the way the algorithms are implemented. Whereas GloVe explicitly underweights relatively rare terms, Word2Vec explicitly underweights high frequency terms. Consequently, Word2Vec often picks out relatively rare terms (including misspellings) as nearest neighbors. In practice this means Word2Vec is likely to be less "robust," i.e. embeddings will tend to be more corpus specific, than GloVe.

Third, for our set of politics queries (and some vocabulary subsetting to improve the quality of the Word2Vec nearest neighbors), the Turing test implied that our human raters are on average indifferent between the two models.

8 Advice to Practitioners

Here we report the main takeaways for practitioners looking to use word embeddings in their research. First, in terms of *choice hyperparameters* in applied work:

- **Window-size and embedding dimensions:** with the possible exception of small corpora like the State of the Union speeches, one should avoid using very few dimensions (below

100) and small window-sizes (< 5), especially if interested in capturing topical semantics. If one cares about syntactic relationships, then the model choice should be based on that criterion first (i.e. small windows may be desirable). While performance improves with larger window-sizes and more dimensions, both exhibit decreasing returns—improvements are marginal beyond 300 dimensions and window-size of 6. Given the tradeoff between more dimension/larger window-size and computation time, the popular choice of 6 (window-size) and 300 (dimensions) seems reasonable. This particular specification is also fairly stable meaning one need not estimate multiple runs to account for possible instability.

- **Pre-trained vs local embeddings:** `GloVe` pre-trained embeddings generally exhibit high correlations (> 0.4 for the set of random queries and > 0.65 for the set of curated queries) with embeddings trained on our selection of political corpora.[11] At least for our focus *Congressional Record* corpus, there is little evidence that using pre-trained embeddings is problematic for subdivisions of the corpus by party—Republican vs Democrat speech.

  Human coders generally prefer pre-trained representations, but not for every term, and it is quite close for many prompts. Specifically, `GloVe` pre-trained word embeddings achieve on average—for the set of political queries—80% of human performance and are generally preferred to locally trained embeddings. Where they do prefer pre-trained embeddings, it is apparently due to (some) local ones simply requiring too much specialized domain knowledge. Thus, if a researcher has a particularly technical or specialized understanding of a term (beyond something which could be easily communicated to a human coder), they should use locally trained embeddings.

  These results suggest embeddings estimated on large online corpora (e.g. Wikipedia and Google data dumps) can reasonably be used for the analysis of contemporaneous political texts. Moreover, the output of our local models is also reasonably liked by our human raters

[11] This is lower in the case of small corpora like the State of the Union, and in the case of random queries for the Spanish corpus.

though they prefer the pre-trained models.

Second, in terms of methodology lessons on *how* to evaluate models:

- **Query search:** in the absence of a clearly defined evaluation metric—a downstream task with labeled data—embeddings can be compared in terms of how they "organize" the embedding space. We propose doing so using query search ranking correlations for a set of randomly selected queries and—given a specific domain of interest— a set of representative domain-specific queries. To discriminate between models resulting in very different embedding spaces, both can be compared to a baseline, either a model known to perform well or, as we do, a human baseline.

- **Crowdsourcing:** Crowdsourcing provides a relatively cheap alternative to evaluate how well word embedding models capture human semantics. We had success with a *triad task* format, a choice-task with an established track-record and solid theoretical grounding in psychology.

- **Human "Turing" test:** a given embeddings model—or any model of human semantics for that matter—can be said to approximate human semantics well if, on average, for any given cue, the model generates associations (nearest neighbors) that a human cannot systematically distinguish from human generated associations.

  Specifically, we define human performance as the point at which a human rater is on average indifferent between a computer and a human generated association.

Third, in terms of *instability*

- **Stability:** word embedding models have non-convex objective functions – can have multiple locally optimal solutions. This produces additional variability beyond sampling error which, if unaccounted for, can lead to mistaken and non-replicable inferences. To account for estimation-related instability we endorse estimating the same model several times, each with different randomly drawn initial word vectors and use an average of the distance metric

of choice. The good news, from our results at least, is that embeddings that perform well on the human and other metrics also tend to be the most stable.

Fourth, in terms of algorithm (`GloVe` or `Word2Vec` (skip-gram))

- **`GloVe` vs. `Word2Vec` (skip-gram):** although `GloVe` is mathematically very similar to `Word2Vec`'s skip-gram architecture, in practice they will diverge, often quite substantially, in their mapping of the semantic space. `Word2Vec` benefits from a more careful filtering of the vocabulary (e.g. increasing the minimum count or setting a lower maximum number of words in the vocabulary) as it tends to *over* weight relatively rare terms (often misspellings). Once `Word2Vec` has an appropriately filtered vocabulary, it performs as well as `GloVe` with human raters.

9 Discussion of Results

Why do we get the results we do? That is, why are pre-trained embeddings sometimes preferred to locally fit ones given that the latter are domain specific? And why do humans sometimes prefer human created neighbors, but sometimes prefer those generated by a statistical model?

On the issue of poor local fits, one possibility is simply a lack of data. That is, corpora being used for such fits are too small to exhibit the helpful smoothing that a very large corpus (like Wikipedia) would allow. Thus, even with weighting down rare terms, small corpora have idiosyncratic co-occurrences (perhaps even typos) that are unappealing to our human coders. It is also possible that our crowdworkers have strong pre-existing beliefs or opinions about our political prompts. Suppose, for example, that the majority of crowdworkers are (strong) Democrat partisans, and that therefore their priors regarding the nearest neighbors of `Republican` are systematically different to those of a more removed expert coder. We expect these crowdworkers—relative to the researcher—to be skeptical of the nearest neighbors from a model trained on the relatively abstruse language found in the *Congressional Record*. Related, above we noted that on some particular cues, it was the case that crowdworkers did not much like model suggestions that required

highly specific domain knowledge. To reiterate a point we made earlier though, we assume such effects are small; in any case, our present focus is not on human-subjects research (in which our crowdworkers become subjects of an experiment).

As to the core Turing issue—that humans sometimes prefer model output rather than that of other humans—we suspect this is fundamentally connected to issues of sampling. Even though we remove outlier human suggestions, it may be the case that a model aggregating over millions of words is more reasonable, on average. Meanwhile, one pathology of embeddings is that they can quickly become out of date (e.g. until recently "`Trump`" would be a word with nearest neighbors pertaining to real estate or casinos, rather than the presidency).

Finally, is there any evidence that an end user would suffer in terms of the merits of their study should they choose the wrong model specification? This is beyond the scope of the current paper, but in Supporting Information I we give an example of "negative" consequences.

10 Conclusions

Word embeddings in their modern scalable form have captured the attention of industry and academia—including social science. As with all methodological advances, it is vital that we understand what they can do for us and what they cannot.

For our domain, we have good news: by all the technical and substantive criteria we used, off-the-shelf pre-trained embeddings work very well relative to—and sometimes better than—both human coders, and more involved locally trained models. Furthermore, locally-trained embeddings perform similarly—with exceptions—across specifications. This should reduce end-user angst about their parameter choices. The general form of these findings extend to historical and non-English texts. Lastly and with caveats, `GloVe` and `Word2Vec` both enjoy similar performance as rated by human coders.

We dealt with a broad but necessarily limited number of possible options. This includes the nature of our text data, which is elite-level (mostly pre-prepared) speeches—not letters, tweets

or free-form communication. Of course, other researchers will care about different concepts and specifications. Irrespective of those particularities however, our work-flow will be useful. Finally, of course, we have focused on *relative* performance: we have not studied whether embeddings are interesting or useful *per se* for understanding behavior, events and so on. We leave such questions for future work.

References

Antoniak, Maria and David Mimno. 2018. "Evaluating the stability of embedding-based word similarities." *Transactions of the Association for Computational Linguistics* 6:107–119.

Benoit, Kenneth, Drew Conway, Benjamin E Lauderdale, Michael Laver and Slava Mikhaylov. 2016. "Crowd-sourced text analysis: Reproducible and agile production of political data." *American Political Science Review* 110(2):278–295.

Blei, David M, Andrew Y Ng and Michael I Jordan. 2003. "Latent dirichlet allocation." *Journal of machine Learning research* 3(Jan):993–1022.

Bolukbasi, Tolga, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*. pp. 4349–4357.

Chiu, Billy, Anna Korhonen and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. pp. 1–6.

Collobert, Ronan and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM pp. 160–167.

Denny, Matthew J and Arthur Spirling. 2018. "Text preprocessing for unsupervised learning: why it matters, when it misleads, and what to do about it." *Political Analysis* 26(2):168–189.

Diermeier, Daniel, Jean-François Godbout, Bei Yu and Stefan Kaufmann. 2012. "Language and ideology in Congress." *British Journal of Political Science* 42(1):31–55.

Faruqui, Manaal, Yulia Tsvetkov, Pushpendre Rastogi and Chris Dyer. 2016. "Problems with evaluation of word embeddings using word similarity tasks." *arXiv preprint arXiv:1605.02276* .

Firth, John Rupert. 1957. *Studies in linguistic analysis*. Wiley-Blackwell.

Gentzkow, Matthew, J.M. Shapiro and Matt Taddy. 2018. "Congressional Record for the 43rd-114th Congresses: Parsed Speeches and Phrase Counts.".
URL: *https://data.stanford.edu/congress_text*

Halpern, David and Pedro Rodriguez. 2018. Partisan representations: Partisan differences in semantic representations and their role in attitude judgments. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*. pp. 445–450.

Hamilton, William L, Jure Leskovec and Dan Jurafsky. 2016. "Diachronic word embeddings reveal statistical laws of semantic change." *arXiv preprint arXiv:1605.09096* .

Islam, Aylin Caliskan, Joanna J Bryson and Arvind Narayanan. 2016. "Semantics derived automatically from language corpora necessarily contain human biases." *CoRR,abs/1608.07187* .

Khodak, Mikhail, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart and Sanjeev Arora. 2018. "A la carte embedding: Cheap but effective induction of semantic feature vectors." *arXiv preprint arXiv:1805.05388* .

Kozlowski, Austin C, Matt Taddy and James A Evans. 2018. "The geometry of culture: Analyzing meaning through word embeddings." *arXiv preprint arXiv:1803.09288* .

Mebane Jr, Walter R, Patrick Wu, Logan Woods, Joseph Klaver, Alejandro Pineda and Blake Miller. 2018. "Observing Election Incidents in the United States via Twitter: Does Who Observes Matter?".

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

Monroe, Burt L, Michael P Colaresi and Kevin M Quinn. 2008. "Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict." *Political Analysis* 16(4):372–403.

Montgomery, Jacob M and Santiago Olivella. 2018. "Tree-Based Models for Political Science Data." *American Journal of Political Science* 62(3):729–744.

Pennington, Jeffrey, Richard Socher and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pp. 1532–1543.

Pierrejean, Bénédicte and Ludovic Tanguy. 2017. Towards Qualitative Word Embeddings Evaluation: Measuring Neighbors Variation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. pp. 32–39.

Quinn, Kevin M, Burt L Monroe, Michael Colaresi, Michael H Crespin and Dragomir R Radev. 2010. "How to analyze political attention with minimal assumptions and costs." *American Journal of Political Science* 54(1):209–228.

Rheault, L, Beelen K, Cochrane C and Hirst G. 2016. "Measuring Emotion in Parliamentary Debates with Automated Textual Analysis." *PLOS ONE* 11(12).

Rheault, Ludovic and Christopher Cochrane. 2019. "Word Embeddings for the Analysis of Ideological Placement in Parliamentary Corpora." *Political Analysis* pp. 1–22.

Roberts, Margaret E, Brandon M Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson and David G Rand. 2014. "Structural topic models for open-ended survey responses." *American Journal of Political Science* 58(4):1064–1082.

Rodman, Emma. 2019. "A Timely Intervention: Tracking the Changing Meanings of Political Concepts with Word Vectors." *Political Analysis* pp. 1–25.

Rudolph, Maja, Francisco Ruiz, Susan Athey and David Blei. 2017. Structured embedding models for grouped data. In *Advances in Neural Information Processing Systems*. pp. 251–261.

Sahlgren, Magnus. 2006. The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces PhD thesis.

Schnabel, Tobias, Igor Labutov, David Mimno and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp. 298–307.

Turing, Alan. 1950. "Computing machinery and intelligence." *Mind* 59(236):433.

Zhang, Han and Jennifer Pan. 2019. "CASM: A Deep-Learning Approach for Identifying Collective Action Events with Text and Image Data from Social Media." *Sociological Methodology* 49(1):1–57.

Online Supporting Information:
Word Embeddings
What works, what doesn't, and how to tell the difference for applied research

Contents (Appendix)

A Window Size and Discrimination for a real corpus

The claim is that larger windows allow us to better discriminate between term meanings. We looked at the evidence for this on our *Congressional Record* corpus. To assess the claim we first set up a set of 'true negatives'—words that should be (fairly) unrelated. In particular for us, these are just random pairs of words from our corpus. We also evaluated how the average distance varies for 'true positives', that is words that are in fact the same. To assess this we sampled 100 words

from the vocabulary. Suppose `congress` is one of those 100 words. We then...

1. tag half of the appearances (randomly selected) of `congress` in the corpus as `congress_tp`. So, if `congress` appears 10,000 times, in our transformed corpus it will appear as `congress` 5000 times, and `congress_tp` 5000 times.

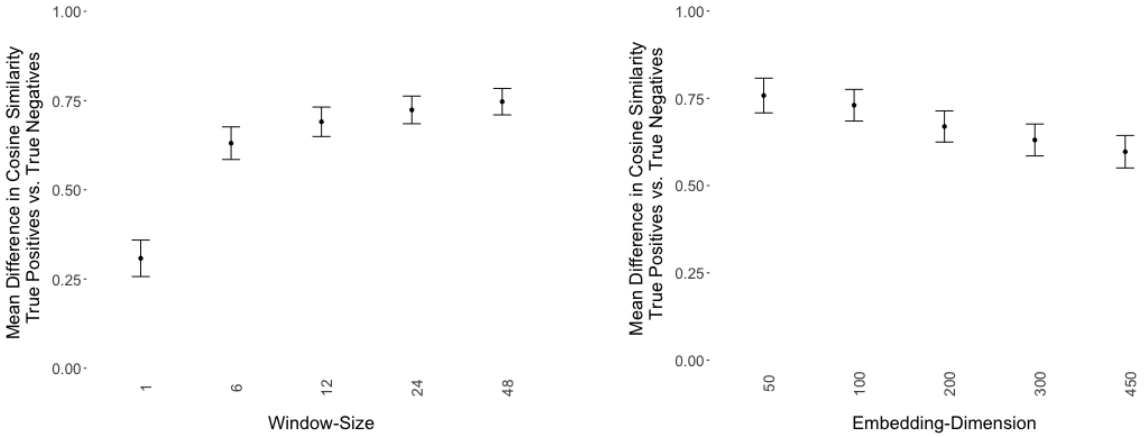
2. estimate a set of embeddings with the vocabulary including both `congress` and `congress_tp`.

Now we have an embedding for `congress` and `congress_tp`. These should be close in embedding space, since they are the same word albeit (randomly) half the incidences have been given a different token (hence we call them "true positives"). We interpret *how* close they are as measure of performance.

In Figure 8a we plot the mean difference in similarity terms between the true positives and the true negatives. When this number is large, we are saying similar words look much more similar to one another than random words (i.e. our model is performing well). When this number is smaller, the model is telling us it cannot distinguish between words that are genuinely similar and words that are not. On the left of the figure, fixing the embedding dimensions at 300, we see that larger windows translate to bigger differences—i.e. the model performs better in terms of discrimination. We call this *meaningful separability*. As an aside, on the right of the figure, we see that for a fixed window-size of 6, increasing the number of dimensions actually causes the model to do worse.

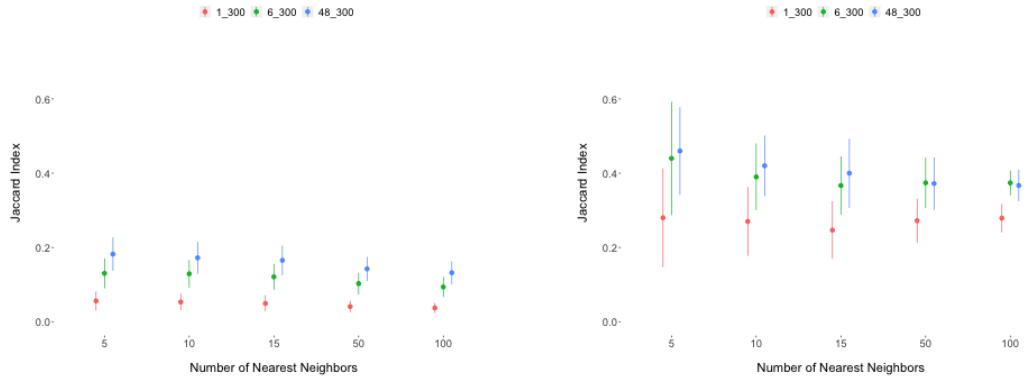
B Jaccard Index

To further evaluate the correspondence between pre-trained embeddings and local models we use the average Jaccard-index —also known as the intersection over the union (IoU)—over the set of random and politics queries (Sahlgren, 2006; Pierrejean and Tanguy, 2017). The Jaccard-index between two models for a given query corresponds to the number of common nearest neighbors in the top N (the intersect of the two sets), over the union of the two sets. For example, take the following two sets of top 5 nearest neighbors for the query term `democracy`:



(a) Varying Window-Size (dimensions = 300) (b) Varying Dimensions (window-size = 6)

Figure 8: Mean Difference in Cosine Similarity True Positives vs. True Negatives



(a) Random Queries (b) Politics Queries

Figure 9: Jaccard Index Between Pre-Trained and Local Models

$A = \{$ `freedom`, `democratic`, `ideals`, `vibrant`, `symbol`$\}$ and $B = \{$`freedom`, `democratic`, `dictatorship`, `democratization`, `socialism`$\}$. Given two nearest neighbors in common, the IoU is $\frac{|A \cap B|}{A \cup B} = \frac{2}{8} = 0.25$. Figure 9 plots the Jaccard-index, for various values of N, between GloVe pre-trained embeddings and several local models varying by window size. Unlike with the Pearson correlations we do not subset the respective vocabularies. As with the Pearson correlations, we observe larger values as window-size increases but with decreasing returns.

C Task Wording

Context Words

A famous maxim in the study of linguistics states that:

You shall know a word by the company it keeps. (Firth, 1957)

This task is designed to help us understand the nature of the ''company'' that words ''keep'': that is, their CONTEXT.

Specifically, for a CUE WORD, its CONTEXT WORDS include words that:

- Tend to occur in the vicinity of the CUE WORD. That is, they are words that appear close to the CUE WORD in written or spoken language.

AND/OR

- Tend to occur in similar situations to the CUE WORD in spoken and written language. That is, they are words that regularly appear with other words that are closely related to the CUE WORD.

For example, CONTEXT WORDS for the cue word COFFEE include:

1. *cup* (tends to occur in the vicinity of COFFEE).
2. *tea* (tends to occur in similar situations to COFFEE, for example when discussing drinks).

Click ''Next'' to continue

Next

(a) Context Words

Task Description

For each iteration of the task (13 in total including trial and screener tasks):

1. You will be given a cue word (top center of the screen) and two candidate context words (on either side of the cue word).
2. Please select the candidate context word that you find best meets the definition of a context word.
3. We are especially interested in context words likely to appear in **political discourse.**
4. If both are reasonable context words, please select whichever you find most intuitive.
5. You must select **one and only one** of the two candidate context words.

Keep in mind, some iterations are for screening purposes. These are tasks for which there is clearly a correct answer.

Wrong answers in these screening tasks will automatically end your participation so **be sure to read carefully.**

The trial task that follows is meant for you to practice. Like screening tasks, the trial task has a correct answer.

Click ''Next'' to continue to the trial runs

Next

(b) Task Instructions

Figure 10: Instructions

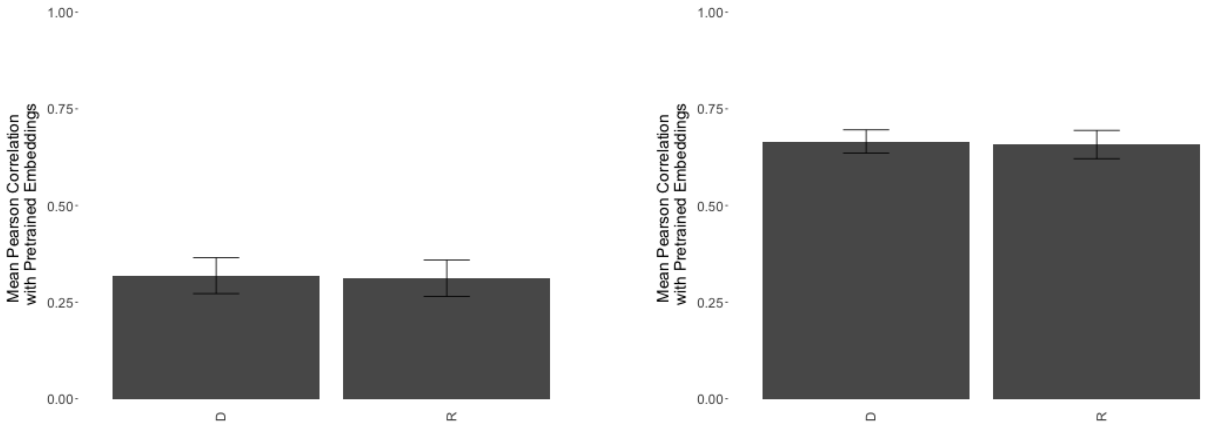
D Specific Details on Crowdsourcing: What and Who

In most cases there is some overlap in the set of nearest neighbors being compared. Rather than show subjects a triad task with identical candidate context words, we adjust the final tally for the probability of such tasks occurring —a function of the amount of overlap— and assume either model has 50% chance of being selected. For both tasks above—collecting human generated nearest neighbors and the triad task—we created specialized `RShiny` apps that we deployed on MTurk. We restrict the set of workers to being US based with at least 100 previously approved HITs and a "Masters" qualification on Amazon Mechanical Turk. For the triad task we paid workers $1 to perform 13 such comparisons—one for each of our political prompt words, one trial run and two quality checks; for the word generation task we paid workers $3 to generate 10 associations for each of ten political prompts. Workers were not allowed to perform both tasks. The code for both apps is available from our `GitHub`.

E Pre-trained embeddings perform equally across subgroups for *Congressional Record*

Above we showed that overall `GloVe` pre-trained embeddings correlate highly with locally trained embeddings. Next we ask whether these correlations differ by party. Such biases can be problematic if pre-trained embeddings are subsequently used to analyze texts and draw conclusions on the basis of party. To evaluate whether pre-trained embeddings exhibit bias we compute the correlation —in cosine similarity rankings for our set of queries— between pre-trained embeddings and group-specific (Democrat and Republican legislators) locally trained models. According to this metric, pre-trained embeddings are biased to the left or to the right if they correlated more highly with either model when used to find semantically related words.

This evaluation requires we estimate separate embeddings for each of these groups. To do so, we split the congressional corpus by party (Republican vs Democrat). We apply the same



(a) Random Queries

(b) Politics Queries

Figure 11: Pearson correlation of group embeddings with pre-trained `GloVe` embeddings.

estimation framework as laid out in section 4 to each sub-corpora except we fix window-size and embedding dimension at 6 and 300 respectively.

Figures 11a and 11b display the main results of our evaluation for a random set of queries and our set of politics queries respectively. For neither set of queries do we find evidence of partisan bias—as defined here—in pre-trained embeddings. To be clear, this result does not mean that pre-trained embeddings do not exhibit common cultural biases —they do according to previous research Bolukbasi et al. (2016); Islam, Bryson and Narayanan (2016)— but rather that pre-trained embeddings —`GloVe` specifically— are equally correlated with party specific embedding models.

F Other Corpora, Other Languages: Data

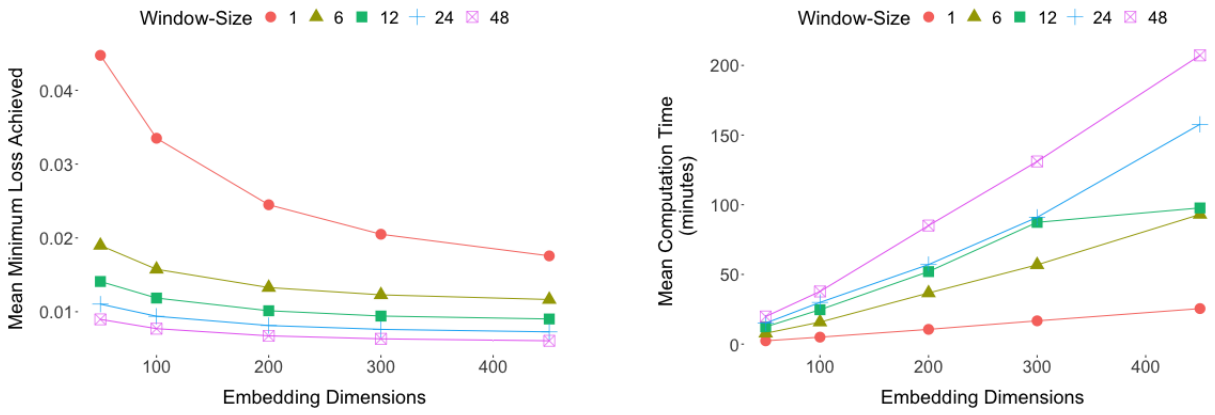
Corpus	Period	Num. of Docs.	Num. of Tokens	Avg. Tokens/Doc.	Vocab. Size	Lexical Div.
Congressional Record	1991 - 2011	1,411,740	3.4×10^8	238	91,856	0.0003
Parliamentary Speeches	1935 - 2013	4,455,924	7.2×10^8	162	79,197	0.0001
State of the Union	1790 - 2018	239	2.0×10^6	8143	11,126	0.0057
Spanish Legislature	1993 - 2018	1,320,525	3.0×10^8	224	94,970	0.0003
German Legislature	1998 - 2018	1,193,248	0.8×10^8	69	108,781	0.0013

Note: Lexical diversity is measured as the number of unique tokens over total number of tokens.

Table 3: Corpora Summary Statistics for this paper.

G Other Corpora, Other Languages: Results

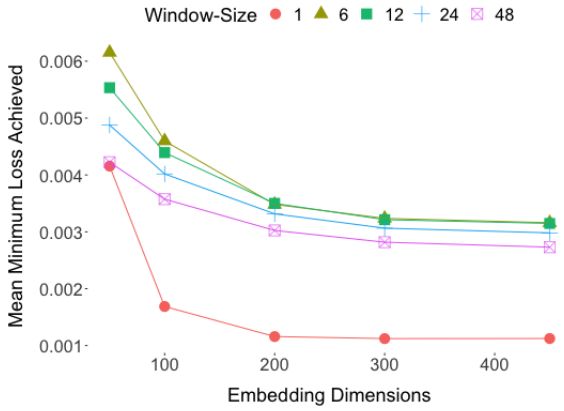
G.1 Technical Criteria



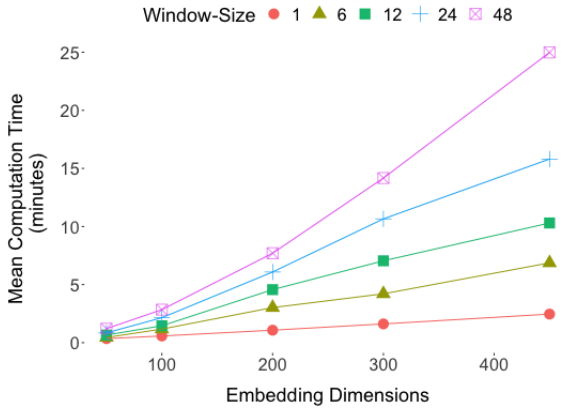
(a) Mean Minimum Loss Achieved

(b) Computation Time (minutes)

Figure 12: Technical Criteria: Parliamentary Speeches

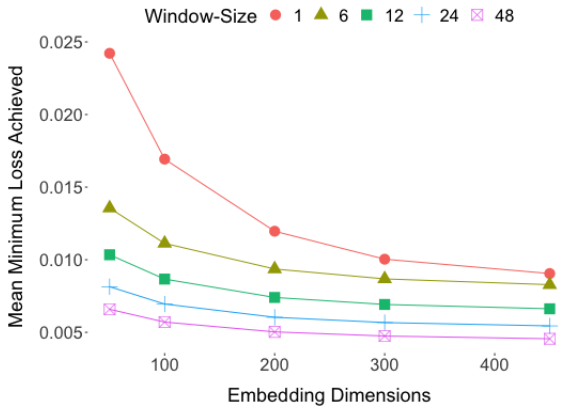


(a) Mean Minimum Loss Achieved

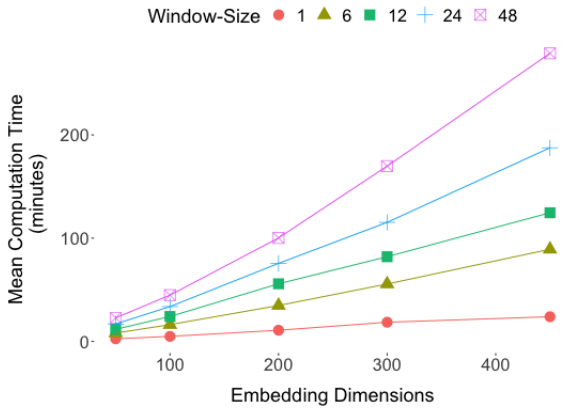


(b) Computation Time (minutes)

Figure 13: Technical Criteria: State of the Union Speeches

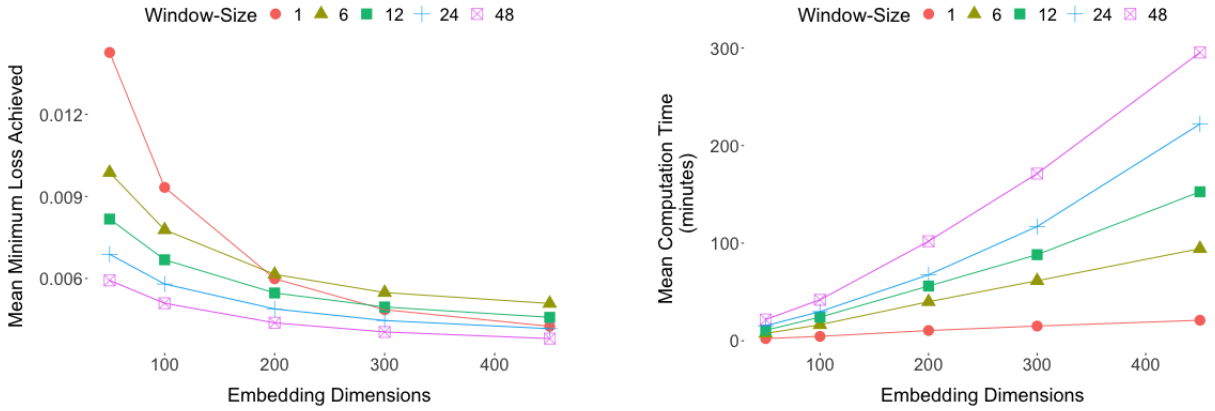


(a) Mean Minimum Loss Achieved



(b) Computation Time (minutes)

Figure 14: Technical Criteria: Spanish Corpus

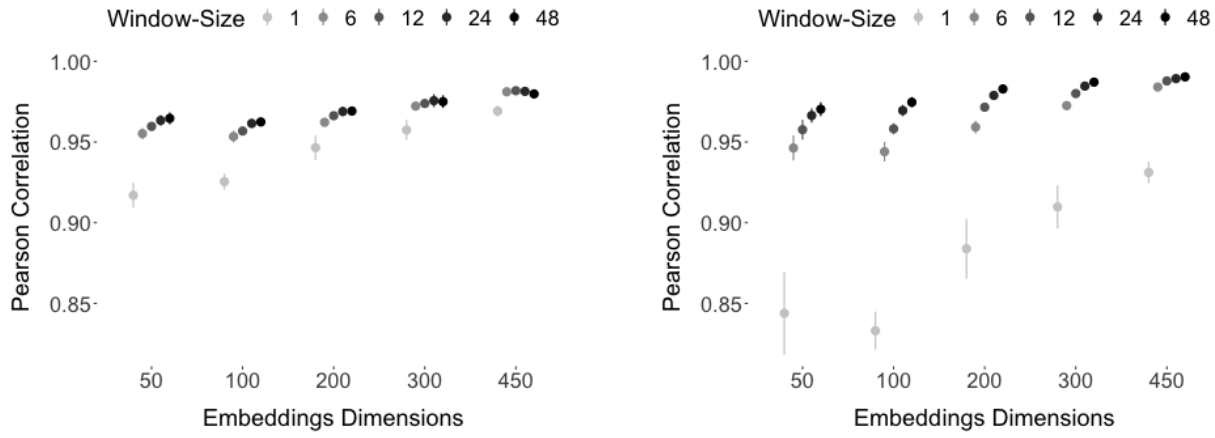


(a) Mean Minimum Loss Achieved

(b) Computation Time (minutes)

Figure 15: Technical Criteria: German Corpus

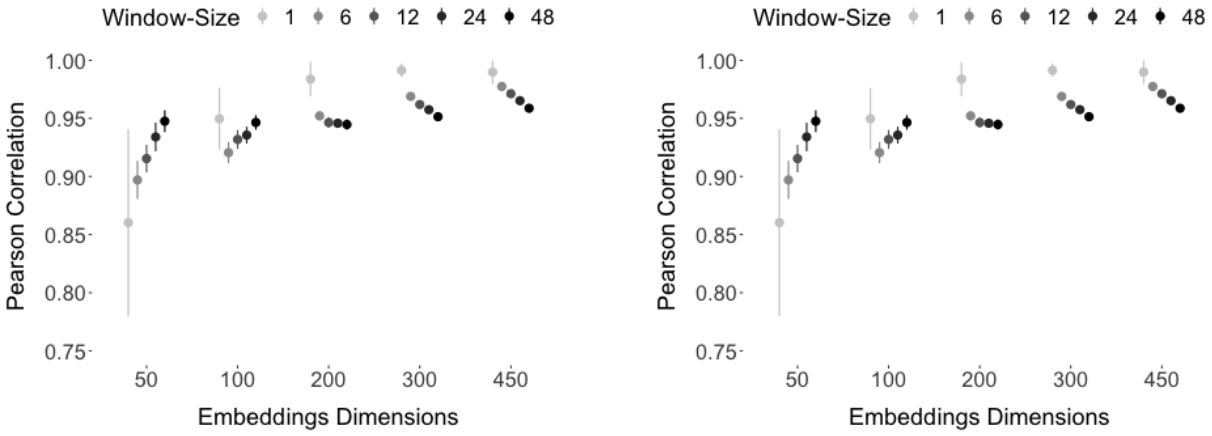
G.2 Stability



(a) Random Queries

(b) Politics Queries

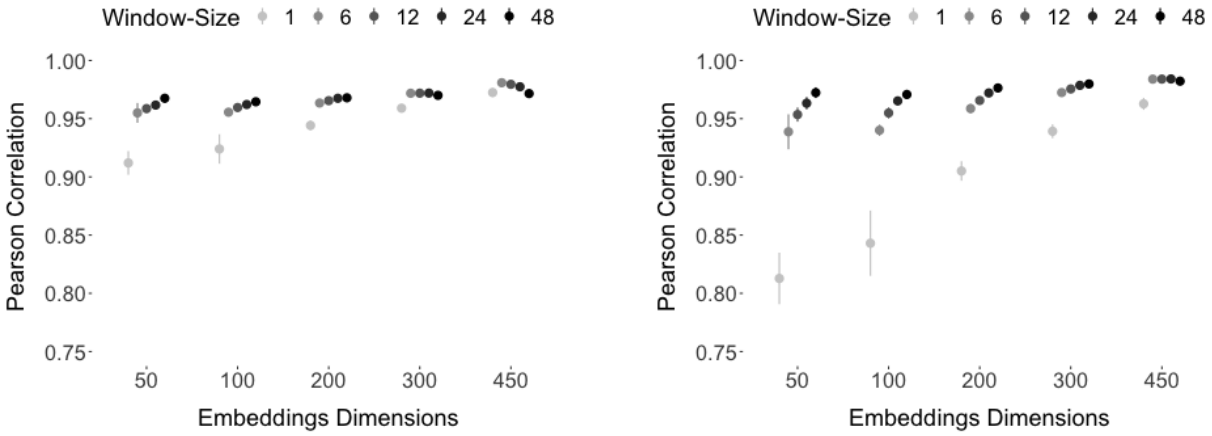
Figure 16: Stability Criteria: Parliamentary Speeches



(a) Random Queries

(b) Politics Queries

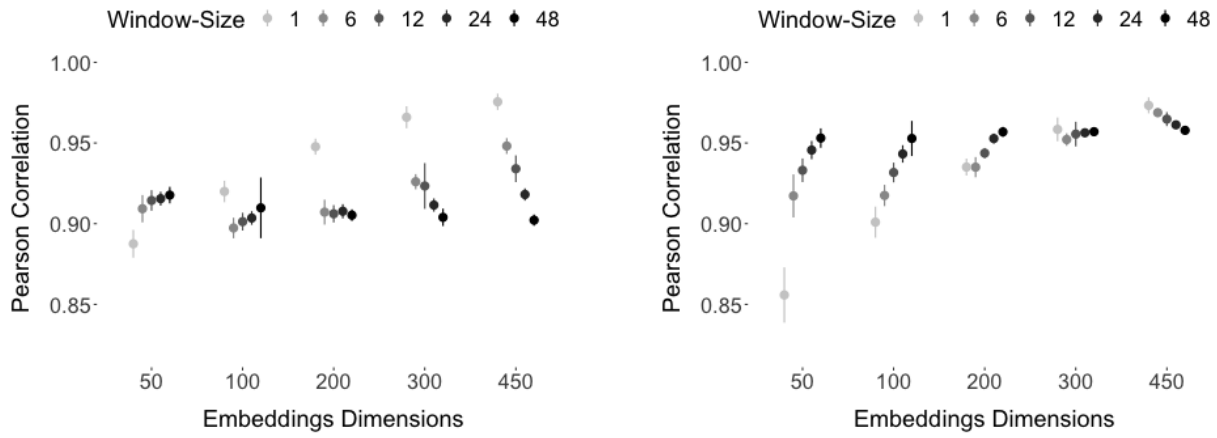
Figure 17: Stability Criteria: State of the Union Speeches



(a) Random Queries

(b) Politics Queries

Figure 18: Stability Criteria: Spanish Corpus

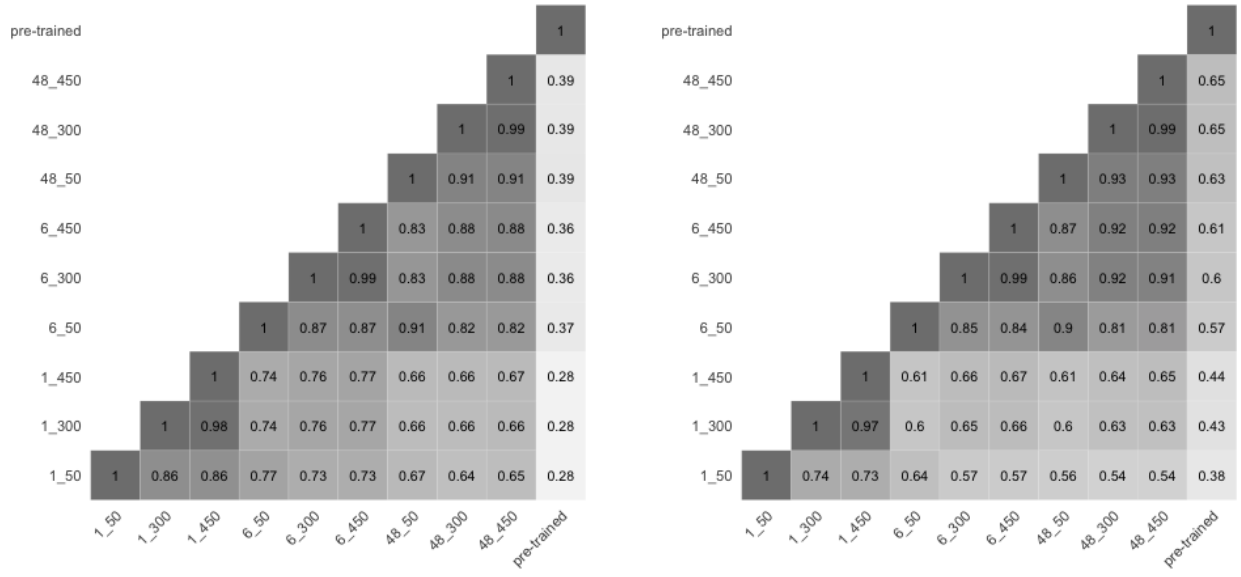


(a) Random Queries

(b) Politics Queries

Figure 19: Stability Criteria: German Corpus

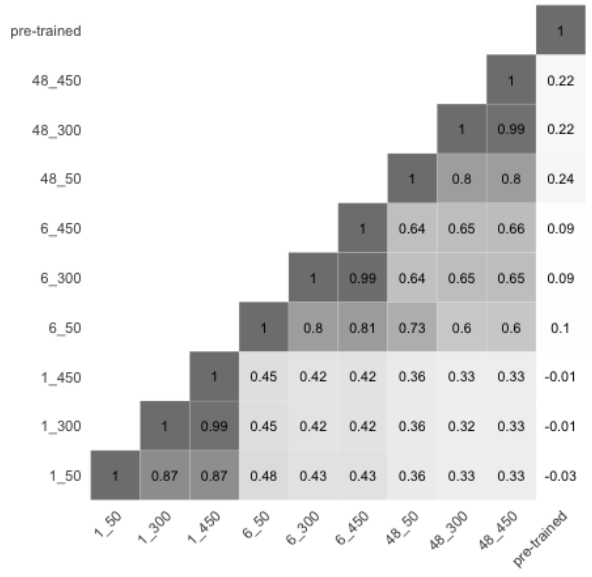
G.3 Query Search Ranking Correlation



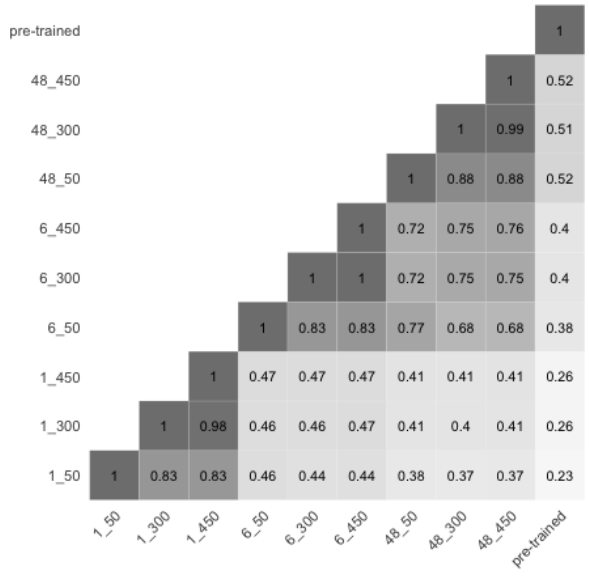
(a) Random Queries

(b) Politics Queries

Figure 20: Query Search Ranking Criteria: Parliamentary Speeches

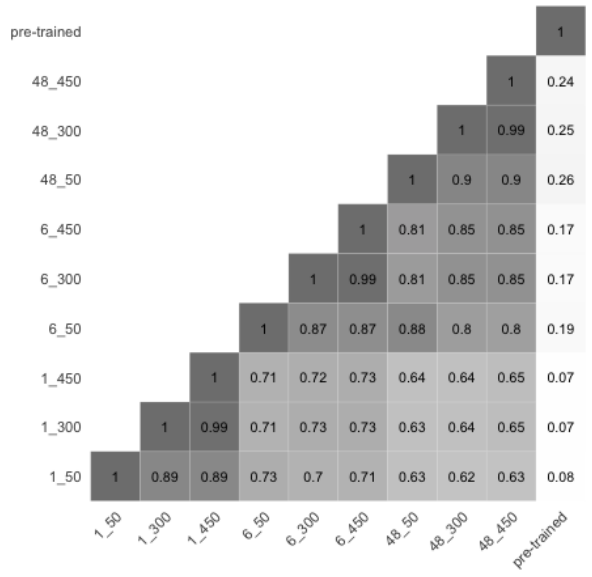


(a) Random Queries

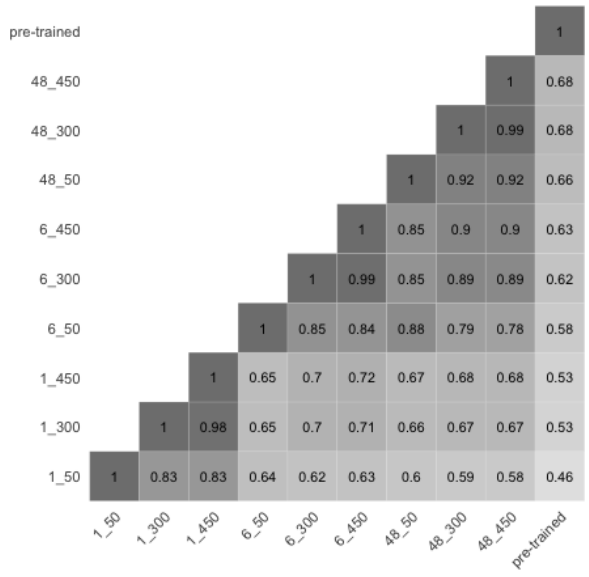


(b) Politics Queries

Figure 21: Query Search Ranking Criteria: State of the Union Speeches

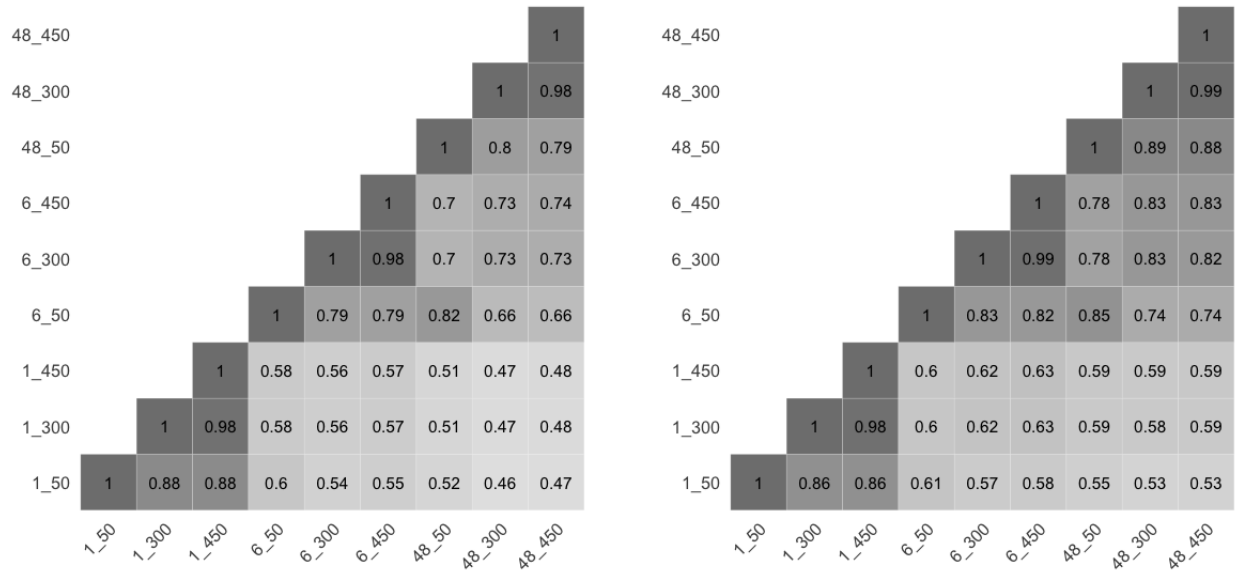


(a) Random Queries



(b) Politics Queries

Figure 22: Query Search Ranking Criteria: Spanish Legislature

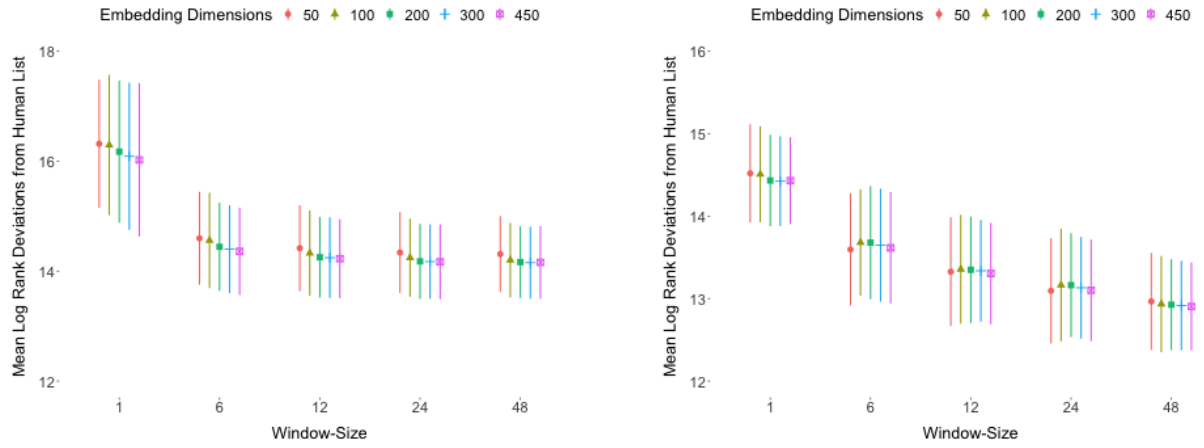


(a) Random Queries

(b) Politics Queries

Figure 23: Query Search Ranking Criteria: German Legislature

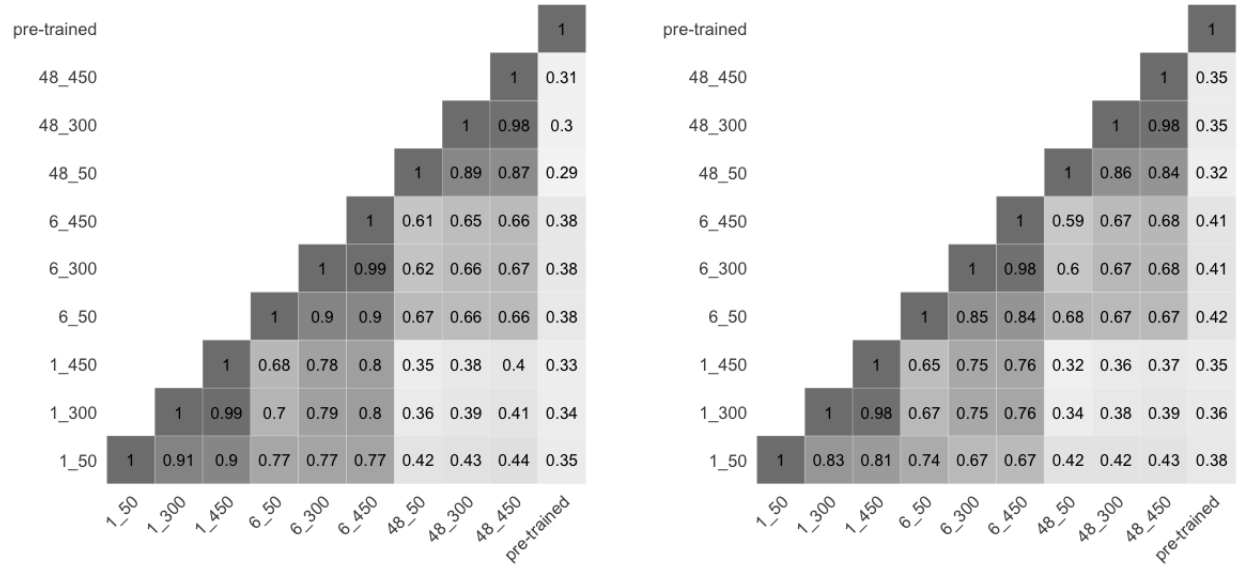
G.4 Human Validation



(a) Parliamentary Speeches

(b) State of the Union Speeches

Figure 24: Human Preferences-Log Rank Deviations



(a) Random Queries

(b) Politics Queries

Figure 25: Query Search Ranking Criteria

H Comparing `GloVe` and `Word2Vec`

To compare `GloVe` with `Word2Vec`, we implemented the same estimation setup with `Word2Vec`—skip-gram architecture—as we did with `GloVe`. So for each parameter pair we estimated ten different sets of embeddings, each starting from a different random initialization. We again restricted the vocabulary to words with a minimum count of 10 and ran each model for 5 epochs. Otherwise we set all hyperparameters to their default values in the Python `gensim` module.[12]

Figures 25a and 25b display the correlations between the set of locally trained models as well as between those and the pre-trained `Word2Vec` embeddings.[13] The results differ markedly from those obtained using `GloVe`. Models with window size 6 are now more highly correlated with the smaller—window size 1—than with the larger models—window size 48. More importantly, `Word2Vec` pre-trained embeddings exhibit much lower correlations with the set of local models than was the case with `GloVe`.

[12]We run `gensim` in `R` using the `reticulate` package.

[13]We use pre-trained embeddings with a window size of 6 and embeddings dimension 300.

In Figure 26 we directly compare both algorithms using a subset of the local models along with both sets of pre-trained embeddings. The correlation between both algorithms increases as we increase window size, yet it is never particularly high (for our set of parameter values). Moreover, and surprising to us, `Word2Vec` and `GloVe` pre-trained embeddings are themselves not all that highly correlated at 0.29.[14] One potential explanation for this result is that they are trained on different corpora.[15] We postulate however that the main source of differences lies in the implementation details. In particular, whereas `GloVe` explicitly underweights relatively rare terms, `Word2Vec` explicitly underweights high frequency terms. Consequently, `Word2Vec` often picks out relatively rare terms (including misspellings) as nearest neighbors as evidenced in Table 4.[16] In practice this means `Word2Vec` is likely to be less "robust," i.e. embeddings will tend to be more corpus specific, than `GloVe`.

democracy		**freedom**		**equality**		**justice**		**immigration**	
W2V	**GloVe**	**W2V**	**GloVe**	**W2V**	**GloVe**	**W2V**	**GloVe**	**W2V**	**GloVe**
pluralism	freedom	liberty	liberty	equal	equal	justices	rehnquist	naturalization	naturalization
freedom	democracies	freedoms	democracy	enfranchisement	racial	rehnqulst	scalia	ins	illegal
democracyand	democratic	democracy	freedoms	racial	fairness	nowchief	owen	immigrations	ins
democracies	liberty	freedomthe	expression	liberty	gender	rehnquist	ginsburg	aliens	reform
liberty	promoting	freedomfreedom	equality	faimess	freedom	justiceand	court	immigrants	customs
democracythe	capitalism	freedoma	free	egalitarianism	liberty	scalia	souter	asylum	border
democratization	stability	pluralism	speech	suffrage	struggle	brennan	oconnor	undocumented	nationality
pluralistic	promote	freedomand	religious	nonviolence	justice	bablitch	brennan	border	immigrants
selfgovernment	pluralism	freedomour	enduring	fairness	tolerance	antonin	department	illegal	laws
democracys	peace	tyranny	prosperity	inclusiveness	harmony	justicethat	supreme	immigrant	aliens

Table 4: Nearest Neighbors `Word2Vec` (local 6-300) and `GloVe` (local 6-300): note that `Word2Vec` selects rarer terms, including typos.

[14]For this comparison we subsetted both vocabularies to the intersection of the two. The `Word2Vec` vocabulary consists of 3 million words whereas the `GloVe` vocabulary consists of $400,000$ words.

[15]`Word2Vec` is trained on a Google News corpus while `GloVe` is trained on Wikipedia 2014 and Gigaword 5.

[16]This is not wrong per se—it makes sense for a word's misspellings to be its nearest neighbors—but is something researchers ought to keep in mind when prioritizing nearest neighbors.

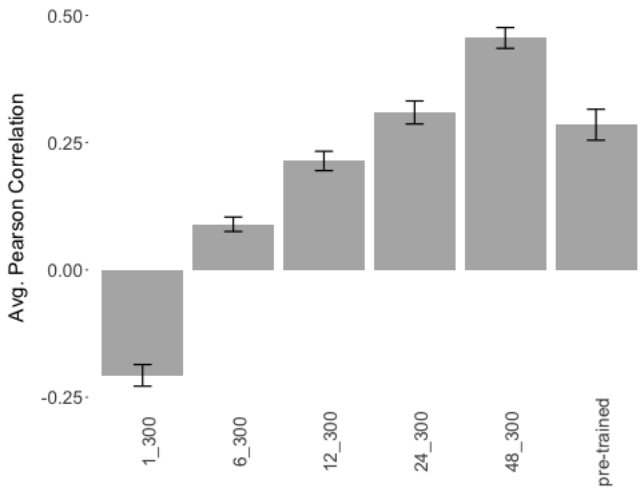


Figure 26: Avg. Pearson Correlation `GloVe` v `Word2Vec` (politics queries)

Additionally, we applied our Turing assessment to compare the two sets of pre-trained embeddings. For this exercise, we subsetted—post-estimation— the vocabularies to the intersection of the two. The latter greatly improved the quality of `Word2Vec`'s nearest neighbors by eliminating relatively rare terms (often typos). Figure 27 displays the results. Clearly, at least for our set of politics queries, our human raters are on average indifferent between the two models.

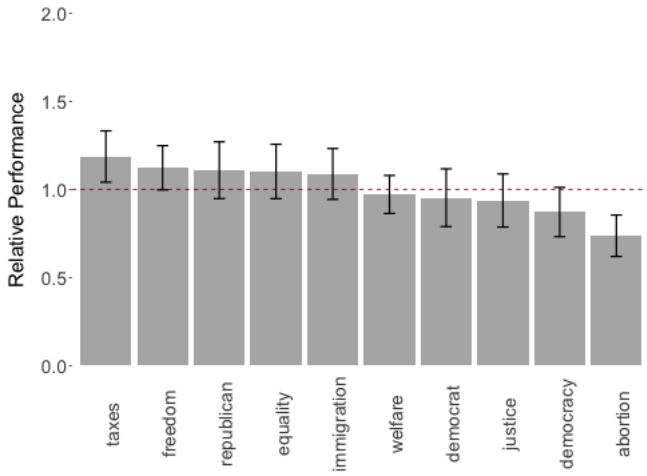


Figure 27: Human Preferences-Turing Assessment
Candidate: `Word2Vec` Baseline: `GloVe`

I What could possibly go wrong? Problems with using inappropriate embedding models

In Table 5 we report the top-5 nearest neighbors for two different specifications of `GloVe` models fit to two of our corpora. In the first two columns, we compare "immigration" for a model with a small window and short representation vector (1-50) with a more standard specification (6-300). This exercise is repeated for *Hansard* for the word "abortion".

The most immediate observation is that the representation and thus inference differs within corpus, depending on the specification. Thus, we see the 6-300 specification reports "naturalization", "illegal" and "INS" (Immigration and Naturalization Service) as the nearest neighbors for "immigration" in the *Congressional Record*, while the 1-50 reports "tort", "reform" and "laws". However, without reference to purpose, it is misleading to claim that one list is correct and one is incorrect. Topically, the 6-300 words do seem more appropriate; but that is what we would expect given previous results. Similarly, they might help us build a better dictionary, or locate search terms of interest. To reiterate though, the 1-50 neighbors are not "wrong" *per se*. It is more that the words are capturing a different sense of context. One possibility here, for example, is that the 1-50 context is about legislative issues arising at the same time (or pushed by the same legislators) as "immigration" was being discussed. Similarly, when we switch to *Hansard* we see that the topical context of "abortion" is best captured by the 6-300 model. But the 1-50 model perhaps captures some temporal context: the decriminalization of abortion in the UK occurred in 1968, and is approximately contemporaneous with ending "conscription" (1960) and the beginning of "fluoridation" of the public water supply, along with changes to "insolvency" law (1976).

Congressional Record		*Hansard*	
"immigration"		**"abortion"**	
`1-50`	`6-300`	`1-50`	`6-300`
tort	naturalization	extradition	abortions
reform	illegal	insolvency	contagious
laws	(ins) INS	arbitration	contraception
bankruptcy	reform	conscription	clinics
ethics	customs	fluoridation	pregnancy

Table 5: Comparing top-5 nearest neighbors across `GloVe` specifications. Note that 6-300 typically returns better topical context words.

Note that we have candidly "cherry-picked" our comparisons here. That is, for other words, the differences between specifications are minor. Nonetheless, as a *possibility* result, our findings stand.