Understanding Word Embedding Stability Across Languages and Applications

by

Laura Anne Wendlandt Burdick

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2020

Doctoral Committee:

Professor Rada Mihalcea, Chair
Professor Joyce Chai
Assistant Professor David Jurgens
Postdoctoral Research Fellow Jonathan K. Kummerfeld
Associate Professor David Mimno, Cornell University

Laura Anne Wendlandt Burdick

lburdick@umich.edu

ORCID iD: 0000-0002-9953-4592

To my grandmother Dr. Leslie McCasey Thom, the first woman in my family to earn her Ph.D., and to my parents, without whom I would not have made it this far.

In all things, to God be the glory. “Come and see what God has done: He is awesome in His deeds toward the children of man.” (Psalm 66:5)

ACKNOWLEDGMENTS

Many, many people supported and encouraged me throughout the process of writing this dissertation, and without such a community, this work would not be half of what it is today.

My biggest thanks go to my advisor, mentor, and friend Dr. Rada Mihalcea. She has guided me through every part of this journey, from learning how to do research to compiling the final dissertation. She taught me how to find interesting research problems, how to take a research idea from its initial germination through to its final publication, and, perhaps most importantly, how to continually work to make computer science a field that is welcoming for all.

The members of my thesis committee have been invaluable in helping me shape the ideas presented here. Dr. Jonathan Kummerfeld has mentored me throughout much of this dissertation, providing research insights, helping debug code, and seeing the vision for this work even when I failed to see it. Drs. Joyce Chai, David Jurgens, and David Mimno, through conversations and other feedback, have sharpened my thinking and provided many excellent suggestions.

In thinking through those people that have encouraged me professionally, I owe a particular debt of gratitude to my undergraduate professors at Grove City College who believed in me as a new computer scientist. Without the support of Drs. Bill Birmingham, Dorian Yeager, and others, I would not have continued down this path and applied to graduate school.

As a graduate student at the University of Michigan, many people have made this time both enjoyable and productive. I have become a better researcher by working and collaborating with members of the LIT lab, including Verónica, Michalis, Yiqun, Laura, Santiago, MeiXing, Oana, Ash, Allie, Charlie, Carol, Yiming, Yumou, Mohamed, Mahmoud, Alberto, Aparna, Uriah, Shibu, Steve, Harry, Kostas, Felix, Max, Mohini, Hui, Berrin, and Quincy. Others in the department, including Preeti, Jeeheh, Cathy, Nilmini, Meghan, and Allison, have brought joy through their friendship.

Outside of CSE, I am grateful to have become a part of the loving and supportive community of Christ Church Ann Arbor. In particular, those who were graduate students at the same time as me - Steph, Emily, Maya, and Phil - encouraged me, gave me perspective, and helped me to persevere to the end of this process.

The last acknowledgments here must go to those who have walked the closest with me through every bit of this Ph.D., my family. Thank you to Mom and Dad, Tim, Jess, Nate, Kathy and Frank, Amy, Julie, and Kate for constantly loving and believing in me. Thank you to my love, Scott, who always sees the best in me, who is patient when I am stressed, who loves me in so many little ways, and who never doubted that I would finish this.

This material is based in part upon work supported by the National Science Foundation (NSF #1344257), the Defense Advanced Research Projects Agency (DARPA) AIDA program under grant #FA8750-18-2-0019, and the Michigan Institute for Data Science (MIDAS). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, DARPA, or MIDAS.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Despite the recent popularity of word embedding methods, there is only a small body of work exploring the limitations of these representations. In this thesis, we consider several aspects of embedding spaces, including their stability. First, we propose a definition of stability, and show that common English word embeddings are surprisingly unstable. We explore how properties of data, words, and algorithms relate to instability. We extend this work to approximately 100 world languages, considering how linguistic typology relates to stability. Additionally, we consider contextualized output embedding spaces. Using paraphrases, we explore properties and assumptions of BERT, a popular embedding algorithm.

Second, we consider how stability and other word embedding properties affect tasks where embeddings are commonly used. We consider both word embeddings used as features in downstream applications and corpus-centered applications, where embeddings are used to study characteristics of language and individual writers. In addition to stability, we also consider other word embedding properties, specifically batching and curriculum learning, and how methodological choices made for these properties affect downstream tasks.

Finally, we consider how knowledge of stability affects how we use word embeddings. Throughout this thesis, we discuss strategies to mitigate instability and provide analyses highlighting the strengths and weaknesses of word embeddings in different scenarios and languages. We show areas where more work is needed to improve embeddings, and we show where embeddings are already a strong tool.

CHAPTER 1

Introduction

"Black-box" methods are becoming increasingly common in natural language processing (NLP) and machine learning. These methods have clear inputs and outputs, but the mapping between inputs and outputs is complex and not necessarily well understood. Word embeddings, low-dimensional vectors that capture some semantic and syntactic information about individual words, can be considered black-box methods. While they have been tremendously useful as features in many tasks, the strengths and limitations of word embeddings are not fully understood. We know how embedding algorithms work mathematically, but more work is needed to have a clear understanding of what they are learning. This thesis focuses on understanding the limits and properties of word embedding algorithms, as well as how these affect usage in downstream applications.

1.1 Word Embeddings

The concept of representing words in a corpus using co-occurrence statistics and other information theoretic measures goes back a long way in NLP (see Turney and Pantel [123] for a survey). However, with the recent availability of larger corpora and new neural network-based methods, such as word2vec [86, 83], word embeddings have gained ubiquity in NLP, becoming integrated into nearly every system and even driving new areas of research. Before introducing this dissertation's research questions, we briefly discuss various uses of word embeddings, as well as types of algorithms that have emerged.

In many cases, word embeddings are used as features in a larger system architecture. Collobert et al. [25] was an early paper that took this approach, incorporating word embeddings as inputs to a neural network that performed part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. This line of work has been expanded to include many other tasks, including text similarity [66], sentiment analysis [40], and machine translation [85].

Word embedding research has also developed in a parallel direction, corpus-centered research [6]. In this paradigm, embeddings are used to study the language of a corpus and to draw direct conclusions from the corpus. Word embeddings are not simply a means to better performance on a downstream task; they are a window into the language and the culture of the author of the corpus being examined. This kind of research is prevalent in the digital humanities and computational social sciences. For instance, previous work has used word embeddings to investigate shifts in word meaning over time [51].

Another example of corpus-centered research is the analysis of bias in corpora used in NLP systems. Bolukbasi et al. [15] use word embeddings to show that stereotypically female occupations (e.g., homemaker, nurse, receptionist) are more closely related to female words (e.g., she, woman), while stereotypically male occupations (e.g., warrior, architect, philosopher) are more closely related to male words (e.g., he, man). Garg et al. [46] additionally demonstrate that racial stereotypes are present in word embedding spaces. Caliskan et al. [18] formalize the notion of bias by studying embeddings using an adapted version of the Implicit Association Test, a test commonly used in psychology to measure gender, racial, and other biases in humans [49].

While the research described so far has focused on English, word embeddings have proven to be useful in many different languages and have particularly been leveraged for low-resource languages [2, 62]. To build embeddings across many different languages efficiently, recent research has focused on building cross-lingual embedding spaces, where words from different languages are embedded in the same vector space [108, 20]. There has also been recent interest in using word embeddings as input to a single model that works across a wide range of languages, for instance, in sentiment analysis [135].

As word embeddings have gained popularity, many different types of embedding algorithms have emerged. These algorithms can be roughly divided into two categories: *context-free output* embeddings and *contextualized output* embeddings. Context-free output algorithms produce one embedding per word, regardless of the context (surrounding words) that the word appears in. Contextualized output algorithms produce separate embeddings for the same word, depending on the context of the word (see Chapter 2 for a thorough discussion of these methods).

The majority of this thesis (Chapters 3-5) focuses on context-free output embeddings. While contextualized output word embeddings have made great advancements and are continuing to grow in effectiveness, there is still an important place for research on context-free output embeddings.

Many contextualized output embedding algorithms, such as ELMo [99], BERT [31], and XLNet [130], require huge amounts of computational resources and data. For example,

it takes 2.5 days to train XLNet with 512 TPU v3 chips. Similarly, it takes two weeks to train ELMo with three NVIDIA GTX 1080 Ti GPUs [117]. Strubell et al. [117] estimate that this costs between $433 and $1,472 and emits 262 lbs of CO_2. In addition to requiring heavy computational resources, most contextualized output embedding algorithms need large amounts of data. XLNet uses over 29GB of language data from five different corpora, and BERT uses 3.3 billion words of training data.

In contrast to these large corpora, many datasets from low-resource languages are fairly small [78]. To support scenarios where using huge amounts of data and computational resources is not feasible, it is important to continue developing our understanding of context-free output word embeddings, such as GloVe [96] and word2vec [86]. These algorithms continue to be used in a wide variety of situations, including the computational humanities [1, 53] and languages where only small corpora are available [64]. According to Google Scholar, in 2019 alone, GloVe was cited approximately 4,700 times and word2vec was cited approximately 5,740 times.[1]

Furthermore, recent work by Arora et al. [7] show that in many scenarios, GloVe is able to perform within 5-10% accuracy of BERT on standard benchmark tasks. While there are scenarios that contextualized output embedding algorithms do better on (particularly tasks with language containing complex structure, ambiguous word usage, and words unseen in training), there are many applications where context-free output embeddings work equally well.

At the end of this thesis, in Chapter 6, we shift our focus to a popular contextualized output embedding algorithm, BERT, and extend our analysis to this category of algorithms.

1.2 Properties of Word Embeddings

Much of the previous research using context-free output word embeddings assumes that these embeddings learn meaningful relations, and that relations present in the text between words will always be captured in the embedding spaces. The vast majority of these papers only run their word embedding algorithm once, making the assumption that the embeddings produced are reliable and reasonably consistent.

In this thesis, we examine these assumptions by looking closely at some of the properties of word embeddings. Specifically, we introduce a new metric, stability, and show how this can be used to measure the consistency of word embeddings across various settings.

Consider the example embedding spaces shown in Figure 1.1. Both the left and the

[1] Papers are cited for many reasons, but citation count can be used as a rough estimate for the popularity of an algorithm.

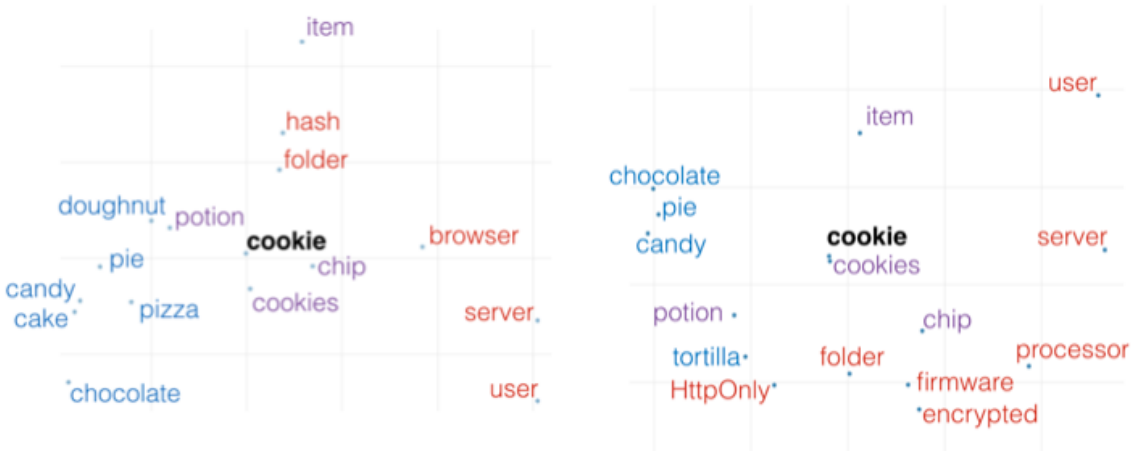


Figure 1.1: The word *cookie* and its fifteen nearest neighbors in two separate embedding spaces. Both spaces are trained on Wikipedia using word2vec; the random seed used to initialize word2vec varies between the two spaces. Here, we show the two-dimensional PCA [94, 58] projection of the embeddings (visualized partially with Parallax [89]). Nearest neighbors are chosen using cosine similarity distance. Food-related words are colored blue, technology-related words are colored red, and ambiguous words are colored purple.

right embedding space are trained using the same algorithm (word2vec), the same data (Wikipedia), and the same parameters. The only aspect that varies between these spaces is the random seed used to initialize the word2vec algorithm. Yet, we can see that there are substantial qualitative differences between these two spaces. In the left space, *cookie* is part of a cluster of words dealing with sweet foods (*candy*, *doughnut*, etc.). In the right space, *cookie* is between words related to sweet foods and words related to the computer sense of *cookie* (*HttpOnly*, *firmware*, etc.). Some words are close to *cookie* in the left space and not close to *cookie* in the right space, and vice versa. If a researcher were studying the usage of the word *cookie* based on one of these two embedding spaces, she might come to different conclusions than if she were using the other embedding space. As is evident from this example, inconsistency between embedding spaces has consequences for research that uses embeddings in a corpus-centered way.

We attempt to quantify some of this qualitative inconsistency using the notion of stability. Specifically, this dissertation focuses on three main research questions:

- **Research Question #1. Are word embeddings stable across variations in data, algorithmic parameter choices, words, and linguistic typologies?**

  We introduce the metric of stability, which measures the consistency of local neighborhoods across two or more context-free output embedding spaces.[2] We show that

[2]This metric was concurrently introduced in Antoniak and Mimno [6].

common English word embedding spaces are surprisingly unstable, even across simple variations such as the change in random initialization of an algorithm. Through exploring the relationship between instability and data properties, word properties, and algorithm properties, we make several interesting observations. Concerning the embedding algorithm used, the algorithm's curriculum (the order of training data given to the algorithm) is important to stability, and GloVe is a more stable embedding algorithm than word2vec or PPMI (see Chapter 2 for a full description of these methods). Properties of individual words also matter; part-of-speech is one of the biggest factors in stability. Finally, the data used to create embedding spaces is important. We show that word frequency within a corpus is not a major factor in stability, but stability when using the same domain data for training and testing is greater than stability across separate domains of data.

We additionally extend this work to approximately 100 languages around the world, where we consider how linguistic typology is related to stability. We draw out several aspects of the relationship between linguistic properties and stability, including that languages with more affixing tend to be less stable, and languages with no gender systems tend to be more stable. These insights can be used in future work to inform the design of embeddings in many languages.

Finally, we consider how to extend stability to contextualized output embeddings. Our initial definition of stability does not work in a contextualized output embedding space, since it relies on a single word having a fixed set of nearest neighbors. Instead, in order to understand how contextualized output embeddings shift under different circumstances, we utilize a unique source of data: paraphrases. Paraphrases give us the ability to control for word semantics, and we use this property to explore properties and assumptions of BERT, a popular contextualized output embedding algorithm. We find the BERT controls for the semantics of paraphrases reasonably well, though not perfectly, and we draw out insights about how BERT handles polysemy, stopwords, and punctuation.

- **Research Question #2. How does our knowledge of stability and other word embedding properties affect tasks where word embeddings are commonly used, both downstream applications and corpus-centered applications?**

  In our initial discussion of stability, we explore how instability affects two downstream applications, word similarity and part-of-speech tagging. We find that there is a relationship between stability and performance on word similarity tasks. For part-of-speech tagging, our LSTM architecture shifts unstable word embeddings more

than the stable embeddings, perhaps "compensating" for the instability of the original embeddings.

When expanding this to many languages around the world, rather than considering specific tasks in diverse languages, we pinpoint linguistic features in certain languages that are likely to lead to lower performance of word embeddings in a variety of scenarios.

Finally, in addition to stability, we also consider other word embedding properties, specifically batching and curriculum learning, and how methodological choices made for these properties affect downstream tasks. This is motivated by our initial study of stability, where we found that the curriculum (order of data given to an algorithm) is related to the stability of the resulting word embeddings. We systematically compare curriculum learning and batching strategies to understand how they affect performance on three downstream tasks, text classification, sentence and phrase similarity, and part-of-speech tagging. We find that certain curriculum learning and batching decisions do increase performance substantially for some tasks.

- **Research Question #3. How does our knowledge of stability and other word embedding properties affect our usage of embeddings?**

  Throughout this thesis, we discuss strategies for how to mitigate instability in both downstream applications and corpus-centered applications. In our discussion of English word embeddings, we give several basic suggestions to use embeddings more robustly, such as using in-domain embeddings whenever possible, preferring GloVe to word2vec, and learning a curriculum for word2vec.

  When we expand this discussion to multiple languages, we highlight particular linguistic properties that cause embeddings to be less stable in a variety of scenarios. Having an understanding of how embeddings change based on language properties will lay the groundwork for building new, stronger embedding methods in various languages.

  By considering curriculum learning and batching in addition to stability, we are able to suggest strategies to improve performance on the three downstream tasks that we consider, text classification, sentence and phrase similarity, and part-of-speech similarity.

In answering these research questions, we provide a deeper understanding of the strengths and limits of word embeddings, and we show how this knowledge can be used practically in multiple languages and applications.

1.3 Thesis Organization

This thesis is organized as follows.

Chapter 2 surveys recent work concerning word embeddings, their applications, and their properties.

Beginning in Chapter 3, we address our main research questions. Chapter 3 introduces the concept of stability and discusses several factors related to stability in English word embeddings. We show that even relatively high frequency words (100-200 occurrences) are often unstable. We provide empirical evidence for how various factors contribute to the stability of word embeddings, and we analyze the effects of stability on two downstream tasks, word similarity and part-of-speech tagging.

Chapter 4 extends our study of stability to a group of diverse languages around the world. We discuss linguistic properties that are related to stability, drawing out insights about correlations with affixing, language gender systems, and other features.

Building on the result in Chapter 3 that curriculum is related to stability, in Chapter 5, we consider how batching and curriculum learning affect the stability of word embeddings and their usage in downstream applications. In order to better understand the impact of these decisions, we present a systematic analysis of different curriculum learning strategies and different batching strategies. We consider multiple datasets for three diverse tasks: text classification, sentence and phrase similarity, and part-of-speech tagging. Our experiments demonstrate that certain curriculum learning and batching decisions do increase performance substantially for some tasks.

Finally, in Chapter 6, we shift our focus to English contextualized output algorithms. We use paraphrases as a unique source of data to analyze a popular algorithm, BERT. Because paraphrases naturally encode word and phrase semantics, we can use them to examine investigate properties of BERT, such as how BERT handles polysemous words and phrase similarity. We see that BERT does a reasonable, but not perfect job recognizing paraphrases, and that BERT correctly handles polysemy in paraphrases. However, BERT unexpectedly gives words that are farther apart lower cosine similarity scores. We also examine previously reported results about how BERT contextualizes certain words and find that BERT gives less contextualized representations to paraphrased words than non-paraphrased words.

We close with conclusions and final thoughts on the research questions in Chapter 7.

CHAPTER 2

Background and Related Work

In this chapter, we cover background on word embeddings, including different embedding algorithms and previous analysis done on embeddings. We also talk about a particular methodology, regression modeling, that we use throughout this dissertation.

2.1 Word Embeddings

Word embeddings are low-dimensional, dense vector representations that capture semantic and syntactic properties of words. Many different algorithms have been proposed for word embeddings, and these algorithms can be roughly divided into two categories: *context-free output* embeddings and *contextualized output* embeddings. Context-free output embedding algorithms produce one embedding per word, regardless of the context (surrounding words) that the word appears in. Contextualized output embedding algorithms produce separate embeddings for the same word, depending on the context of the word. Here, we survey some of the most common algorithms.

2.1.1 Context-Free Output Embedding Algorithms

Many context-free output embedding algorithms are based on the distributional hypothesis, the idea that words that occur in similar contexts tend to have similar meanings [52, 43].

2.1.1.1 Positive Pointwise Mutual Information (PPMI) Matrices

One traditional way to build embedding spaces (also denoted vector space models) is by using positive pointwise mutual information (PPMI) matrices. Let M be a matrix, where each row of M represents a word w, and each column of M represents a context (word) c. A word w appears in the context c if w and c occur within a window of some size. Each cell M_{wc} represents the association between a particular word w and a context c.

This association is measured using positive pointwise mutual information (PPMI) [23], a non-negative variant of pointwise mutual information (PMI). PPMI can be estimated as:

$$PMI(w,c) = \log \frac{\hat{P}(w,c)}{\hat{P}(w)\hat{P}(c)} = \log \frac{count(w,c) \times |D|}{count(w) \times count(c)} \tag{2.1}$$

$$PPMI(w,c) = \max(PMI(w,c), 0) \tag{2.2}$$

where $|D|$ is the number of $(word, context)$ pairs in the corpus, $count(w,c)$ is the number of times that word w and context c occur together in the corpus, $count(w)$ is the number of times that word w occurs in the corpus, and $count(c)$ is the number of times that context c occurs in the corpus. Bullinaria and Levy [17] show that a PPMI matrix outperforms a PMI matrix on a number of semantic tasks.

In this work, we use a PPMI matrix factored using singular value decomposition (SVD) to obtain a denser matrix [38]. This technique is also used in NLP in Latent Semantic Analysis (LSA) [30]. In SVD, the matrix M is decomposed into a product of three matrices:

$$M = U \cdot \Sigma \cdot V^\top \tag{2.3}$$

where U and V have orthonormal columns and Σ is diagonal, with positive diagonal elements (eigenvalues) ordered in decreasing order. As long as M is a real or a complex matrix, the decomposition shown in Equation 2.3 exists. To obtain a d-dimensional embedding matrix M_d using SVD, we keep the top d rows of Σ:

$$M_d = U_d \cdot \Sigma_d \cdot V_d^\top \tag{2.4}$$

2.1.1.2 word2vec

A more recent word embedding algorithm, word2vec [86, 83] uses a shallow neural network to learn word embeddings by predicting context words. There are two separate algorithms included in word2vec, the skip-gram model and the continuous bag-of-words (CBOW) model (see Figure 2.1). CBOW aims to predict a target word given its context words, while skip-gram aims to predict the context words given a target word. In this thesis, we primarily focus on the more common skip-gram model.

The skip-gram model is a one-layer feed-forward neural network seeking to optimize the log probability of a target word given its context words. Specifically, the distribution of a (word w, context c) pair appearing in the data D is modeled using the softmax equation:

$$P(D = 1|w,c) = \sigma(\vec{w} \cdot \vec{c}) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{c}}} \tag{2.5}$$

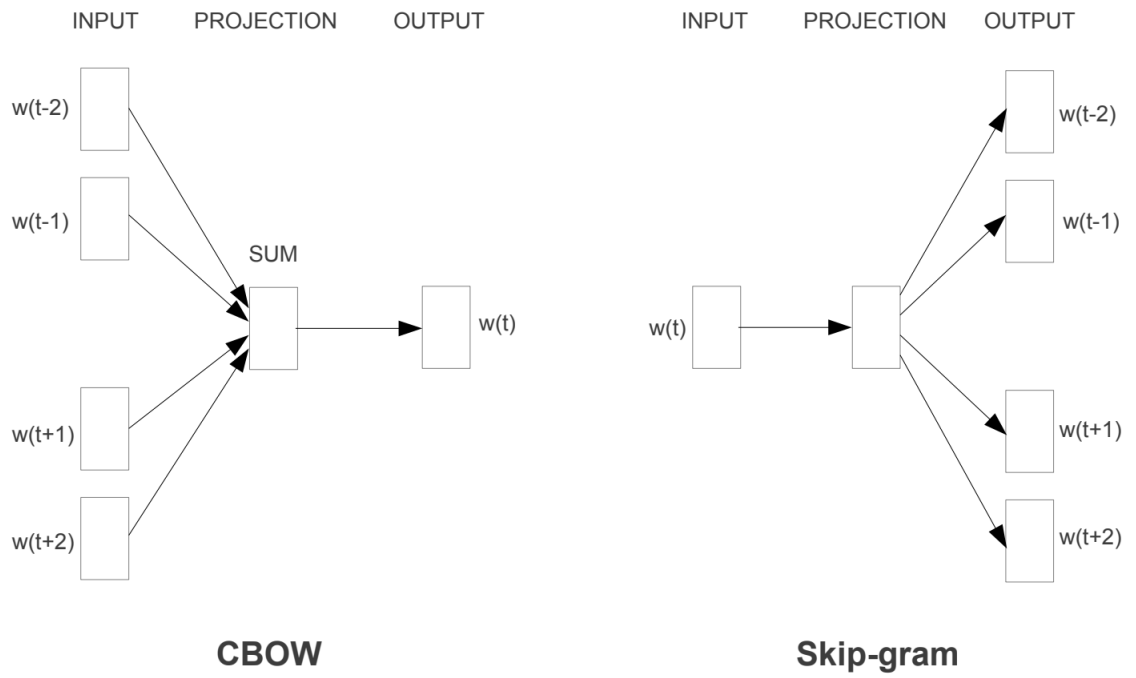


Figure 2.1: CBOW and Skip-gram architectures. $w(t)$ indicates the word at position t. Figure taken from Mikolov et al. [83].

$$P(D = 0|w, c) = 1 - P(D = 1|w, c) \tag{2.6}$$

where $\vec{w}$ represents a d-dimensional word embedding and $\vec{c}$ represents a d-dimensional context embedding, both parameters to be learned by the model. The skip-gram model optimizes $P(D = 1|w, c)$ for (w, c) pairs observed in the data, while maximizing $P(D = 0|w, c)$ for randomly selected pairs of (w, c). These randomly selected pairs are called "negative samples," and the assumption is made that most (w, c) pairs will be negative samples. Thus, the objective function of skip-gram for a single (w, c) pair is to maximize the following:

$$\log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D}[\log \sigma(-\vec{w} \cdot \vec{c_N})] \tag{2.7}$$

where k is the number of negative samples and c_N is the sampled context, drawn according to the following empirical distribution:

$$P_D(c) = \frac{count(c)}{|D|} \tag{2.8}$$

Here, $count(c)$ is the number of times that c appears in the data, and $|D|$ is the number of $(word, context)$ pairs (this formulation of the skip-gram objective function is from Levy and Goldberg [71]). Other optimizations, such as hierarchical softmax and subsampling of

frequent words, are used to speed up the computation of the skip-gram model. Throughout this thesis, we use the Python Gensim implementation of word2vec,[1] which implements standard optimizations, specifically negative sampling with five negative samples and no hierarchical softmax.

2.1.1.3 GloVe

Another recent method for creating word embeddings, GloVe, is based on factoring a matrix of ratios of co-occurrence probabilities [96]. Specifically, GloVe creates d-dimensional word vectors ($\vec{w}$) and context vectors ($\vec{c}$) that satisfy the following equation:

$$\vec{w} \cdot \vec{c} + b_w + b_c = count(w, c) \tag{2.9}$$

where b_w and b_c are bias terms that are learned in the model. To satisfy Equation 2.9, GloVe factors a log-count matrix shifted by the entire vocabularies' bias terms:

$$M^{\log(count(w,c))} \approx W \cdot C^\top + \vec{b^w} + \vec{b^c} \tag{2.10}$$

where W is a matrix of word embeddings, C is a matrix of context embeddings, and $\vec{b^w}$ and $\vec{b^c}$ are learned parameters.

2.1.1.4 Other Algorithms

Other word embedding algorithms have been proposed that incorporate additional information from a corpus. Levy and Goldberg [70] extend word2vec to use different contexts, particularly context from dependency parse trees, and Garimella et al. [47] introduce demographic information of the writer into the skip-gram model. Retro-fitting [40] after training has additional been proposed as a way to refine word embeddings by incorporating semantic information from outside sources such as WordNet [41].

We do not consider these methods in this thesis, because these methods are less common than the main word embedding algorithms discussed above.

2.1.1.5 Analysis

Various aspects of the embedding spaces produced by these algorithms have been previously studied. Particularly, the effect of parameter choices has a large impact on how word2vec, GloVe, and PPMI behave [72]. Further work shows that the parameters of word2vec influence the geometry of the created word vectors and context vectors [88].

[1] Available online at https://radimrehurek.com/gensim/index.html.

These parameters can be optimized; Hellrich and Hahn [54] posit optimal parameters for word2vec's negative sampling and the number of epochs to train for. They also demonstrate that in addition to parameter settings, word properties, such as word ambiguity, affect embedding quality.

Depending on the algorithm parameters and other design decisions, nearest neighbors vary in created embedding spaces [101]. Additional work has looked at how various word properties (both semantic and syntactic information) change as properties of word embeddings shift [8, 129].

At a higher level of granularity, Tan et al. [119] analyze word embedding spaces by comparing two spaces. They do this by linearly transforming one space into another space, and they show that words have different usage properties in different domains (in their case, Twitter and Wikipedia).

Finally, embeddings can be analyzed using second-order properties of embeddings, which are properties of a word embedding derived from nearest neighborhood topological features in an embedding space. Newman-Griffis and Fosler-Lussier [91] validate the usefulness of second-order properties by demonstrating that embeddings based on second-order properties perform as well as typical first-order embeddings.

2.1.1.6 Evaluation of Word Embeddings

There has been much interest in evaluating the quality of different word embedding algorithms. This is typically done extrinsically, by measuring performance on a downstream task, but there have also been efforts to build reliable intrinsic evaluation techniques.

The most common intrinsic evaluation techniques are using embeddings to measure word similarity and relatedness, as well as to complete analogies [48]. Another, linguistically inspired, intrinsic evaluation is using word embeddings to predict FrameNet [10] edges, which capture inheritance, causation, and other relationships [69]. Bakarov [9] provides a survey of additional intrinsic analysis techniques.

In this thesis, we use word similarity as a way to evaluate word embeddings intrinsically. This is a very similar evaluation to word analogies, and these two methods are the most common intrinsic evaluation methods.

2.1.2 Contextualized Output Embedding Algorithms

More recently, contextualized output word embeddings have been used in downstream tasks. These models provide different embeddings for the same word depending on the word's context.

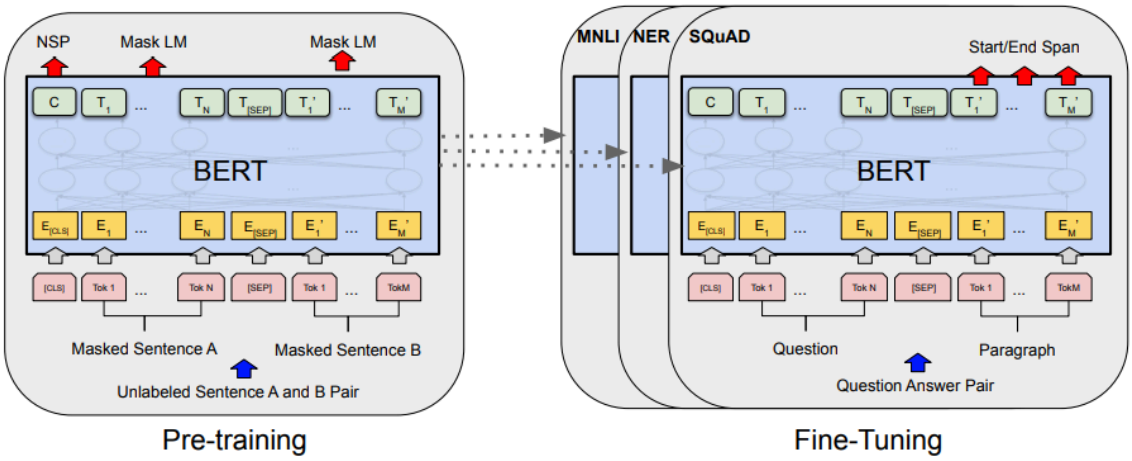


Figure 2.2: A typical pre-training and fine-tuning workflow for BERT. Figure taken from Devlin et al. [31].

One of the first major contextualized output embedding algorithms was ELMo [99]. ELMo is a bi-directional language model (biLM), implemented using forward and backwards LSTMs [56] and trained to maximize the log likelihood of seeing a target word given both the words before the target word (forward language model) and the words after the target word (background language model). To get embeddings, ELMo takes a weighted average of the layers of the biLM for a given word in context. These layer weights are typically learned by a downstream task architecture.

More recent contextualized output word embedding algorithms include BERT [31], ULMFiT [59], XLM [68], RoBERTa [74], and XLNet [130].

As noted in Chapter 1, contextualized output word embedding algorithms require tremendous amounts of computational resources and data to be effective. For instance, it takes 2.5 days to train XLNet with 512 TPU v3 chips, and BERT uses 3.3 billion words of training data. This computational cost makes contextualized output word embedding models unfeasible in some situations.

2.1.2.1 BERT

In Chapter 6, we focus primarily on BERT, since this has become one of the most popular contextualized output embedding algorithms, and many newer algorithms are variants of it (see [106]). BERT is composed of stacks of bi-directional Transformer [124] encoder layers.

Typically, the BERT workflow is broken down into two parts: pre-training and fine-tuning, visualized in Figure 2.2. Instead of a traditional language model training objective,

pre-training uses two unsupervised objectives: masked language modeling (MLM) and next sentence prediction (NSP). For the MLM, or cloze [121], objective, a certain percentage of input tokens are masked at random, and the model must predict these tokens. For the NSP objective, 50% of the time, sentences are presented in the correct order, and 50% of the time, sentences are presented randomly. The model must learn when they are in the correct order and when they are not. Pre-training provides a generic, task-agnostic model. BERT can then be fine-tuned to get embeddings for a particular task. In this step, one or more fully connected layers are added on top of the pre-trained model, and all the parameters are updated using task-specific inputs and outputs. Compared to pre-training, this is a relatively computationally inexpensive task.

BERT has shown impressive performance on a large number of tasks, such as the GLUE benchmark [125], the Stanford Question Answering Dataset (SQuAD) [104], and the Situations With Adversarial Generations (SWAG) dataset [132]. BERT has recently been expanded to 104 languages.[2] While this is a high number of languages, it is only a fraction of the 7,111 languages that exist in the world today [37]. There are still many scenarios where there is not enough data to train BERT for a particular language. BERT is also a general-purpose embedding algorithm and does not have domain-specific embeddings, where there is typically far less training data.

2.1.2.2 Analysis

Dubbed "BERTology," there has been a growing amount of research studying the inner workings of BERT and trying to quantify what this algorithm learns in various scenarios [107].

Of particular interest to this thesis is work that analyzes the embeddings output from BERT. Recent studies have found that BERT embeddings, created from the final layer of BERT, tend to cluster according to word senses [127], though this varies somewhat based on the position of a word in a sentence [82]. The final BERT layers also produce more contextualized word embeddings than the beginning layers [39].

2.2 Methodologies: Regression Analysis

Throughout this thesis, as part of our evaluation of word embedding properties, we will use regression analysis, a statistical method to measure the relationship between one or more variables of interest. Specifically, given N ground-truth data points with M extracted

[2] See https://github.com/google-research/bert/blob/master/multilingual.md.

features per data point, let $\mathbf{x}_n \in \mathbb{R}^{1\times M}$ be the features for sample n and let $\mathbf{y} \in \mathbb{R}^{1\times N}$ be the set of labels. Then, a regression model learns a set of weights $\mathbf{w} \in \mathbb{R}^{1\times M}$ by minimizing the least squares function

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{y}_n - \mathbf{w}^\top \mathbf{x}_n)^2 \tag{2.11}$$

Regression models have previously been used to measure the impact of individual features [112].

Ridge regression [57] is a form of regression modeling that regularizes the magnitude of the model weights, producing a more interpretable model than non-regularized linear regression. Ridge regression adds a regularization term, altering Equation 2.11 to

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{y}_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{2.12}$$

where λ is a regularization constant. In scenarios where there are two correlated features, both features will have similar, but reduced, coefficients. In highly collinear cases, both feature weights will be penalized the same amount.

The goodness of fit of a regression model is measured using the coefficient of determination R^2. This measures how much variance in the dependent variable $\mathbf{y}$ is captured by the independent variables $\mathbf{x}$. A model that always predicts the expected value of $\mathbf{y}$, regardless of the input features, will receive an R^2 score of 0. The highest possible R^2 score is 1, and the R^2 score can be negative.

CHAPTER 3

Factors Influencing the Surprising Instability of Word Embeddings

A version of the content in this chapter appeared in the Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies [126].

Although word embeddings are widely used across NLP, their stability has not yet been fully evaluated and understood. In this chapter, we explore the factors that play a role in the stability of English word embeddings, including properties of the data, properties of the algorithm, and properties of the words. We find that word embeddings exhibit substantial instabilities, which can have implications for downstream tasks.

Using the overlap between nearest neighbors in an embedding space as a measure of stability (see Section 3.1 below for more information), we observe that many common embedding spaces have large amounts of instability. For example, Figure 3.1[1] shows the instability of the embeddings obtained by training word2vec on the Penn Treebank (PTB) [77]. As expected, lower frequency words have lower stability, and higher frequency words have higher stability. What is surprising however about this graph is the medium-frequency words, which show huge variance in stability. This cannot be explained by frequency, so there must be other factors contributing to their instability.

In the following experiments, we explore which factors affect stability, as well as how this stability affects downstream tasks that word embeddings are commonly used for. To our knowledge, this is the first study comprehensively examining the factors behind instability.

[1]Throughout this dissertation, figures are plotted using the Python libraries matplotlib (https://matplotlib.org) [60] and seaborn (https://seaborn.pydata.org). We additionally organize our data using the Python pandas library (https://pandas.pydata.org) [81].

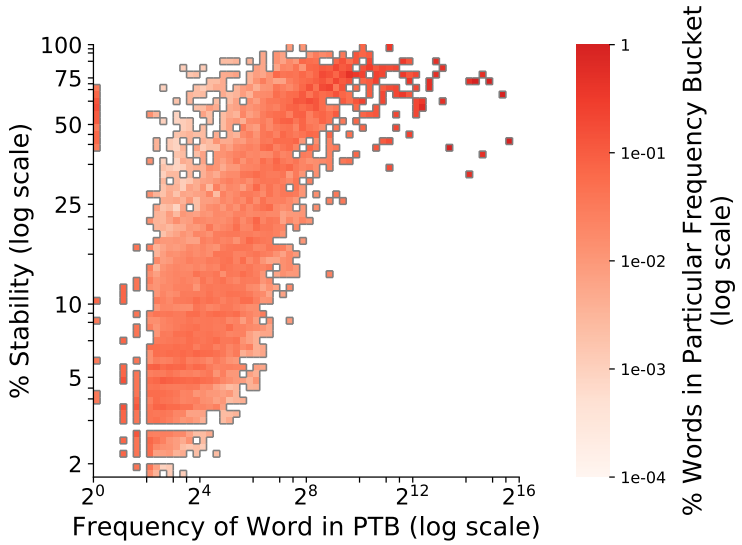


Figure 3.1: Stability of word2vec as a property of frequency in the PTB. Stability is measured across ten randomized embedding spaces trained on the training portion of the PTB (determined using language modeling splits [84]). Each word is placed in a frequency bucket (x-axis), and each column (frequency bucket) is normalized.

Concurrent work by Antoniak and Mimno [6] evaluates how document properties affect the stability of word embeddings. We also explore the stability of embeddings, but focus on a broader range of factors, and consider the effect of stability on downstream tasks. In contrast, Antoniak and Mimno focus on using word embeddings to analyze language (e.g., [46]), rather than to perform tasks.

3.1 Defining Stability

We define *stability* as the percent overlap between nearest neighbors in an embedding space.[2] Given a word W and two embedding spaces A and B, take the ten nearest neighbors of W in both A and B. Let the stability of W be the percent overlap between these two lists of nearest neighbors. 100% stability indicates perfect agreement between the two embedding spaces, while 0% stability indicates complete disagreement. In order to find the ten nearest neighbors of a word W in an embedding space A, we measure distance between words using cosine similarity.[3] This definition of stability can be generalized to more than two embedding spaces by considering the average overlap between two sets of embedding spaces. Let X and Y be two sets of embedding spaces. Then, for every pair of embedding

[2]This metric is concurrently explored in work by Antoniak and Mimno [6].

[3]We found comparable results for other distance metrics, such as l^1 norm, l^2 norm, and l^∞ norm, but we report results from cosine similarity to be consistent with other word embedding research.

Model 1	Model 2	Model 3
metropolitan	*ballet*	**national**
national	**metropolitan**	*ballet*
egyptian	bard	**metropolitan**
rhode	chicago	institute
society	**national**	glimmerglass
debut	*state*	*egyptian*
folk	*exhibitions*	intensive
reinstallation	*society*	jazz
chairwoman	whitney	*state*
philadelphia	*rhode*	*exhibitions*

Table 3.1: Top ten most similar words for the word *international* in three randomly intialized word2vec models trained on the NYT Arts Domain. Words in all three lists are in bold; words in only two of the lists are italicized.

spaces (x, y), where $x \in X$ and $y \in Y$, take the ten nearest neighbors of W in both x and y and calculate percent overlap. Let the stability be the average percent overlap over every pair of embedding spaces (x, y).[4]

Consider an example using this metric. Table 3.1 shows the top ten nearest neighbors for the word *international* in three randomly initialized word2vec embedding spaces trained on the NYT Arts domain (see Section 3.2.3 for a description of this corpus). These models share some similar words, such as *metropolitan* and *national*, but there are also many differences. On average, each pair of models has four out of ten words in common, so the stability of *international* across these three models is 40%.[5]

The idea of evaluating ten best options is also found in other tasks, like lexical substitution (e.g., [80]) and word association (e.g., [47]), where the top ten results are considered in the final evaluation metric. To give some intuition for how changing the number of nearest neighbors affects our stability metric, consider Figure 3.2. This graph shows how the stability of GloVe changes with the frequency of the word and the number of neighbors used to calculate stability; please see the figure caption for a more detailed explanation of how this graph is structured. Within each frequency bucket, the stability is consistent across varying numbers of neighbors. Ten nearest neighbors performs approximately as well as a

[4]Another possible way to define stability would be to align our embedding spaces using a non-linear transformation and then to measure the distance between words. The reason that we chose not to use this metric was because it is currently still an open question how to align embedding spaces well. Aligning the embedding spaces would need to include scaling the spaces so that distances are equivalent in multiple spaces, and it is inconclusive what the best way to do this would be. For these reasons, we choose to use the simpler definition presented here.

[5]Here, the two sets of embedding spaces X and Y are identical.

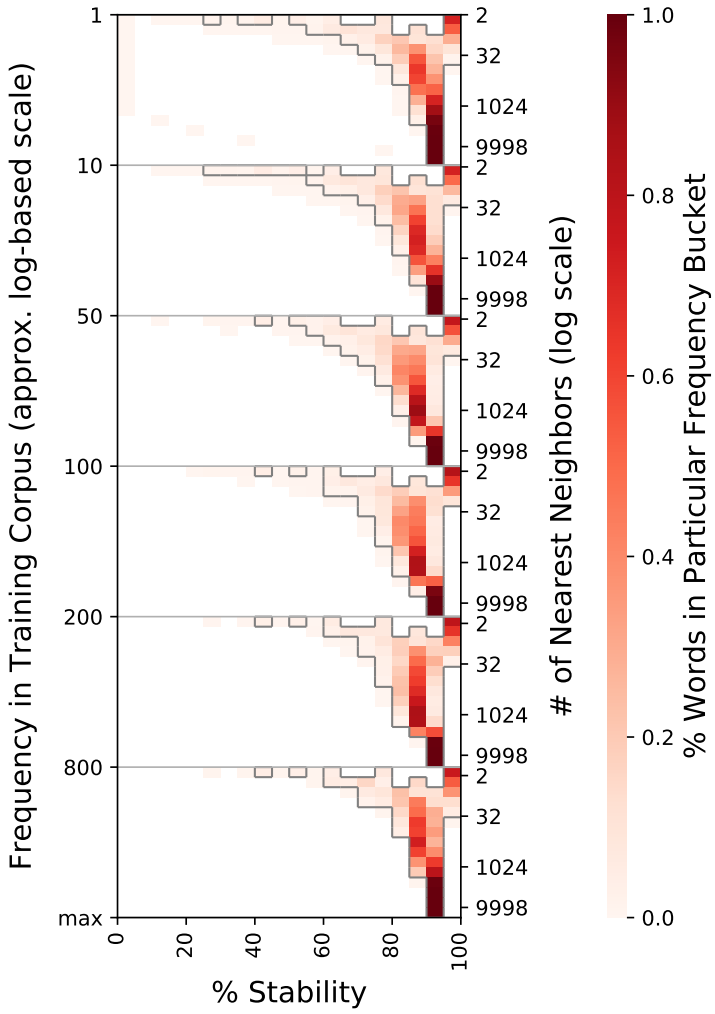


Figure 3.2: Stability of GloVe on the PTB. Stability is measured across ten randomized embedding spaces trained on the training data of the PTB (determined using language modeling splits [84]). Each word is placed in a frequency bucket (left y-axis) and stability is determined using a varying number of nearest neighbors for each frequency bucket (right y-axis). Each row is normalized, and boxes with more than 0.01% of the row's mass are outlined.

higher number of nearest neighbors (e.g., 100). We see this pattern for low frequency words as well as for high frequency words. Because the performance does not change substantially by increasing the number of nearest neighbors, it is computationally less intensive to use a small number of nearest neighbors. We choose ten nearest neighbors as our metric throughout the rest of the chapter.

3.2 Factors Influencing Stability

As we saw in Figure 3.1, embeddings are sometimes surprisingly unstable. To understand the factors behind the (in)stability of word embeddings, we build a regression model that aims to predict the stability of a word given: (1) properties related to the word itself; (2) properties of the data used to train the embeddings; and (3) properties of the algorithm used to construct these embeddings. Using this regression model, we draw observations about factors that play a role in the stability of word embeddings.

3.2.1 Methodology

We use ridge regression, measured using the coefficient of determination R^2, to model these various factors [57]. We set the regularization coefficient $\lambda = 1$. In addition to ridge regression, we tried non-regularized linear regression. We obtained comparable results, but many of the weights were very large or very small, making them hard to interpret.

Given this model, we create training instances by observing the stability of a large number of words across various combinations of two embedding spaces. Specifically, given a word W and two embedding spaces A and B, we encode properties of the *word* W, as well as properties of the *datasets* and the *algorithms* used to train the embedding spaces A and B. The target value associated with these features is the stability of the word W across embedding spaces A and B. We repeat this process for more than 2,500 words, several datasets, and three embedding algorithms.

Specifically, we consider all the words present in all seven of the data domains we are using (see Section 3.2.3), 2,521 words in total. Using the feature categories described below, we generate a feature vector for each unique word, dataset, algorithm, and dimension size, resulting in a total of 27,794,025 training instances. To get good average estimates for each embedding algorithm, we train each embedding space five times, randomized differently each time (this does not apply to PPMI, which has no random component). We then train a ridge regression model on these instances. The model is trained to predict the stability of word W across embedding spaces A and B (where A and B are not necessarily trained using the same algorithm, parameters, or training data). Because we are using this model to learn associations between certain features and stability, no test data is necessary. The emphasis is on the model itself, not on the model's performance on a specific task.

We describe next each of the three main categories of factors examined in the model. An example of these features is given in Table 3.2.

Word Properties	
Primary part-of-speech	`Adjective`
Secondary part-of-speech	`Noun`
# Parts-of-speech	`2`
# WordNet senses	`3`
# Syllables	`5`
Data Properties	
Raw frequency in corpus A	`106`
Raw frequency in corpus B	`669`
Diff. in raw frequency	`563`
Vocab. size of corpus A	`10,508`
Vocab. size of corpus B	`43,888`
Diff. in vocab. size	`33,380`
Overlap in corpora vocab.	`17%`
Domains present	`Arts, Europarl`
Do the domains match?	`False`
Training position in A	`1.02%`
Training position in B	`0.15%`
Diff. in training position	`0.86%`
Algorithm Properties	
Algorithms present	`word2vec, PPMI`
Do the algorithms match?	`False`
Embedding dimension of A	`100`
Embedding dimension of B	`100`
Diff. in dimension	`0`
Do the dimensions match?	`True`

Table 3.2: Consider the word *international* in two embedding spaces. Suppose embedding space A is trained using word2vec (embedding dimension 100) on the NYT Arts domain, and embedding space B is trained using PPMI (embedding dimension 100) on Europarl. This table summarizes the resulting features for this word across the two embedding spaces.

3.2.2 Word Properties

We encode several features that capture attributes of the word W. First, we use the primary and secondary parts-of-speech (POS) of the word. Both of these are represented as bags-of-words of all possible POS, and are determined by looking at the primary (most frequent) and secondary (second most frequent) POS of the word in the Brown corpus[6] [44]. If the word is not present in the Brown corpus, then all of these POS features are set to zero.

[6]Here, we use the universal tagset, which consists of twelve possible POS: adjective, adposition, adverb, conjunction, determiner / article, noun, numeral, particle, pronoun, verb, punctuation mark, and other [100].

Dataset	Sentences	Vocab. Size	Num. Tokens / Vocab. Size
NYT US	13,923	5,787	64.37
NYT NY	36,792	11,182	80.41
NYT Business	21,048	7,212	75.96
NYT Arts	28,161	10,508	65.29
NYT Sports	21,610	5,967	77.85
All NYT	121,534	24,144	117.98
Europarl	2,297,621	43,888	1,394.28

Table 3.3: Dataset statistics.

To get a coarse-grained representation of the polysemy of the word, we consider the number of different POS present. For a finer-grained representation, we use the number of different WordNet senses associated with the word [87, 41].

We also consider the number of syllables in a word, determined using the CMU Pronuncing Dictionary.[7] If the word is not present in the dictionary, then this is set to zero.

3.2.3 Data Properties

Data features capture properties of the training data (and the word in relation to the training data). For this model, we gather data from two sources: the New York Times (NYT) [109] and Europarl [67]. Overall, we consider seven domains of data: (1) NYT - U.S., (2) NYT - New York and Region, (3) NYT - Business, (4) NYT - Arts, (5) NYT - Sports, (6) All of the data from domains 1-5 (denoted "All NYT"), and (7) All of English Europarl. Table 3.3 shows statistics for these datasets. The first five domains are chosen because they are the top five most common categories of news articles present in the NYT corpus. They are smaller than "All NYT" and Europarl, and they have a narrow topical focus. The "All NYT" domain is more diverse topically and larger than the first five domains. Finally, the Europarl domain is the largest domain, and it is focused on a single topic (European Parliamentary politics). These varying datasets allow us to consider how data-dependent properties affect stability.

We use several features related to domain. First, we consider the raw frequency of word W in both the domain of data used for embedding space A and the domain of data for space B. To make our regression model symmetric, we effectively encode three features: the higher raw frequency (between the two), the lower raw frequency, and the absolute difference in raw frequency.

[7] Available online at http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

We also consider the vocabulary size of each corpus (again, symmetrically) and the percent overlap between corpora vocabulary, as well as the domain of each of the two corpora, represented as a bag-of-words of domains. Finally, we consider whether the two corpora are from the same domain.

Our final data-level features explore the role of curriculum learning in stability. It has been posited that the order of the training data affects the performance of certain algorithms, and previous work has shown that for some neural network-based tasks, a good training data order (curriculum learning strategy) can improve performance [12]. Curriculum learning has been previously explored for word2vec, where it has been found that optimizing training data order can lead to small improvements on common NLP tasks [122]. We will further explore the effects of curriculum on word embeddings in Chapter 5. In this chapter, of the embedding algorithms we consider, curriculum learning only affects word2vec. Because GloVe and PPMI use the data to learn a complete matrix before building embeddings, the order of the training data will not affect their performance. To measure the effects of training data order, we include as features the first appearance of word W in the dataset for embedding space A and the first appearance of W in the dataset for embedding space B (represented as percentages of the total number of training sentences).[8] We further include the absolute difference between these percentages.

3.2.4 Algorithm Properties

In addition to word and data properties, we encode features about the embedding algorithms. These include the different algorithms being used, as well as the different parameter settings of these algorithms. Here, we consider three embedding algorithms, word2vec, GloVe, and PPMI. The choice of algorithm is represented in our feature vector as a bag-of-words.

For each algorithm, we choose common parameter settings. For word2vec, two of the parameters that need to be chosen are window size and minimum count. Window size refers to the maximum distance between the current word and the predicted word (e.g., how many neighboring words to consider for each target word). Any word appearing less than the minimum count number of times in the corpus is discarded and not considered in the word2vec algorithm. For both of these features, we choose standard parameter settings, namely, a window size of 5 and a minimum count of 5. For GloVe, we also choose standard parameters. We use 50 iterations of the algorithm for embedding dimensions less than 300, and 100 iterations for higher dimensions.

[8] All word2vec experiments reported here are run in a multi-core setting, which means that these percentages are approximate. However, comparable results were achieved using a single-core version of word2vec.

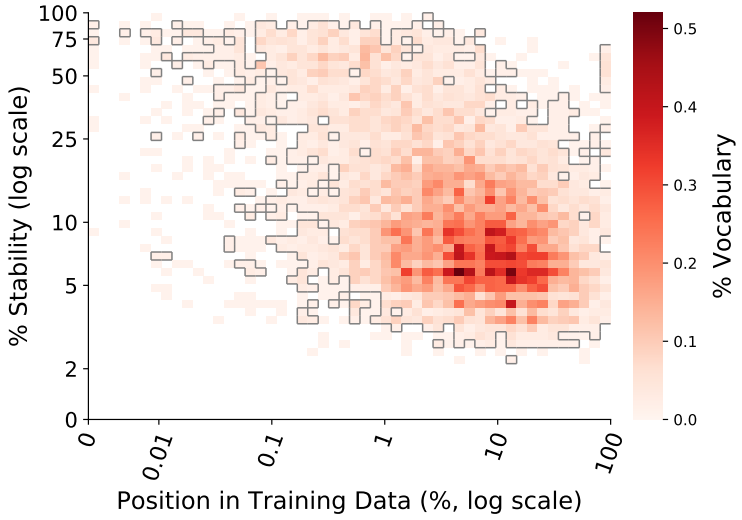
Feature	Weight
Lower training data position of word W	-1.52
Higher training data position of W	-1.49
Primary POS = Numeral	1.12
Primary POS = Other	-1.08
Primary POS = Punctuation mark	-1.02
Overlap between corpora vocab.	1.01
Primary POS = Adjective	-0.92
Primary POS = Adposition	-0.92
Do the two domains match?	0.91
Primary POS = Verb	-0.88
Primary POS = Conjunction	-0.84
Primary POS = Noun	-0.81
Primary POS = Adverb	-0.79
Do the two algorithms match?	0.78
Secondary POS = Pronoun	0.62
Primary POS = Determiner	-0.48
Primary POS = Particle	-0.44
Secondary POS = Particle	0.36
Secondary POS = Other	0.28
Primary POS = Pronoun	-0.26
Secondary POS = Verb	-0.25
Number of word2vec embeddings	-0.21
Secondary POS = Adverb	0.15
Secondary POS = Determiner	0.14
Number of NYT Arts Domain	-0.14
Number of NYT All Domain	0.14
Number of GloVe embeddings	0.13
Number of syllables	-0.11

Table 3.4: Regression weights with a magnitude greater than 0.1, sorted by magnitude.

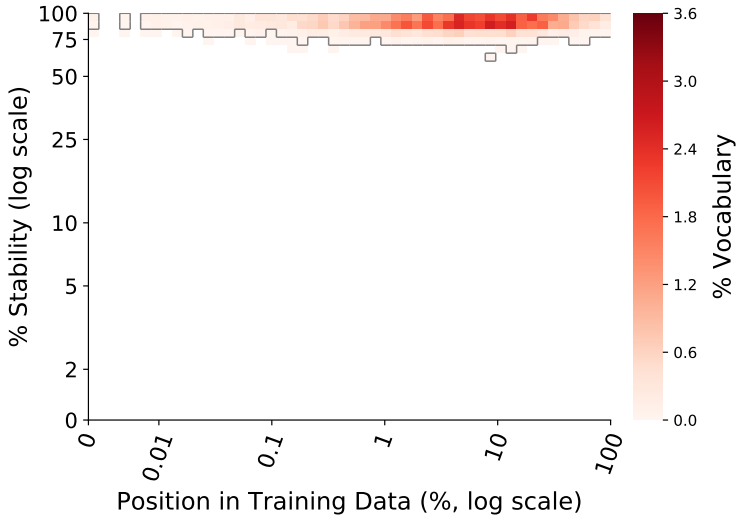
We also add a feature reflecting the embedding dimension, namely one of five embedding dimensions: 50, 100, 200, 400, or 800.

3.3 Lessons Learned: What Contributes to the Stability of an Embedding

Overall, the regression model achieves a coefficient of determination R^2 score of 0.301 on the training data, which indicates that the regression has learned a linear model that reasonably fits the training data given. Using the regression model, we can analyze the



(a) word2vec.



(b) GloVe.

Figure 3.3: Stability of both word2vec and GloVe as properties of the starting word position in the training data of the PTB. Stability is measured across ten randomized embedding spaces trained on the training data of the PTB (determined using language modeling splits [84]). Boxes with more than 0.02% of the total vocabulary mass are outlined.

weights corresponding to each of the features being considered, shown in Table 3.4. These weights are difficult to interpret, because features have different distributions and ranges. However, we make several general observations about the stability of word embeddings.

Observation 1. Curriculum learning is important. This is evident because the top two features (by magnitude) of the regression model capture where the word first appears in the training data. Figure 3.3 shows trends between training data position and stability in the

Primary POS	Avg. Stability
Numeral	47%
Verb	31%
Determiner	31%
Adjective	31%
Noun	30%
Adverb	29%
Pronoun	29%
Conjunction	28%
Particle	26%
Adposition	25%
Punctuation mark	22%

Table 3.5: Percent stability broken down by part-of-speech, ordered by decreasing stability.

PTB. This figure contrasts word2vec with GloVe (which is order invariant).

To further understand the effect of curriculum learning on the model, we train a regression model with all of the features except the curriculum learning features. This model achieves an R^2 score of 0.291 (compared to the full model's score of 0.301). This indicates that curriculum learning is a factor in stability.[9]

Observation 2. Part-of-speech is one of the biggest factors in stability. Table 3.4 shows that many of the top weights belong to POS-related features (both primary and secondary POS). Table 3.5 compares average stabilities for each primary POS. Here we see that the most stable POS are numerals, verbs, and determiners, while the least stable POS are punctuation marks, adpositions, and particles.

Observation 3. Stability within domains is greater than stability across domains. Table 3.4 shows that many of the top factors are domain-related. Figure 3.4 shows the results of the regression model broken down by domain. This figure shows the highest stabilities appearing on the diagonal of the matrix, where the two embedding spaces both belong to the same domain. The stabilities are substantially lower off the diagonal.

Figure 3.4 also shows that "All NYT" generalizes across the other NYT domains better than Europarl, but not as well as in-domain data ("All NYT" encompasses data from US, NY, Business, Arts, and Sports). This is true even though Europarl is much larger than "All NYT".

Observation 4. Overall, GloVe is the most stable embedding algorithm. This is partic-

[9]When looking at all of the words in the training data of the PTB, frequency and the position where a word first appears in the training data are negatively correlated, with a Spearman's correlation [115] of -0.60. This indicates that Figure 3.3 is showing a different trend from Figure 3.1.

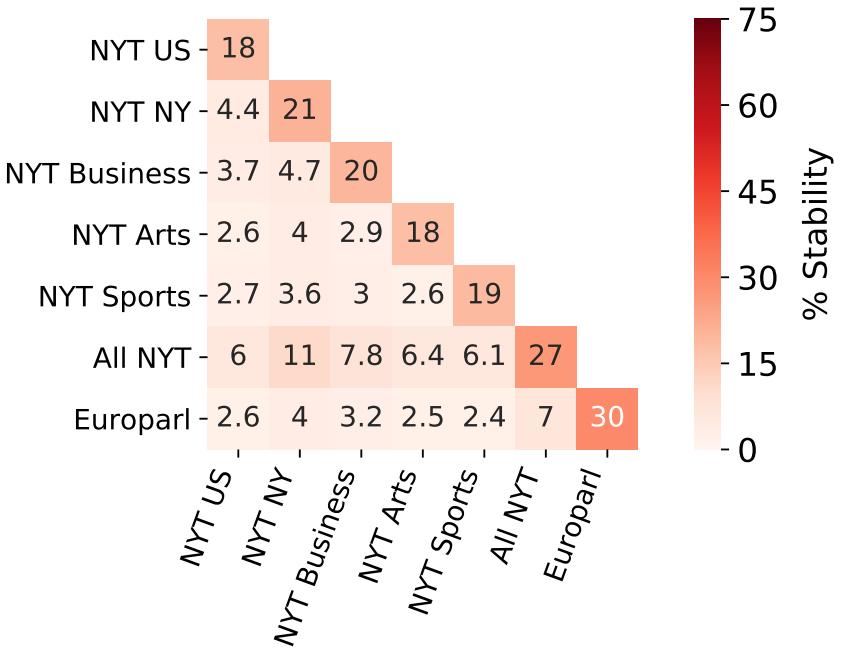


Figure 3.4: Percent stability broken down by domain.

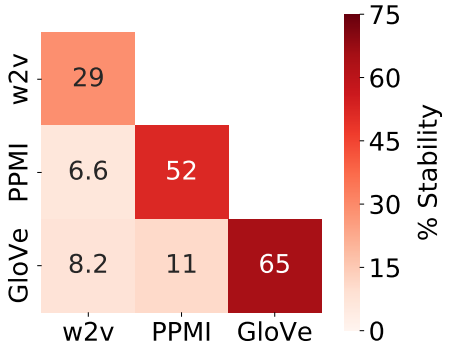


Figure 3.5: Percent stability broken down by algorithm (in-domain data only).

ularly apparent when only in-domain data is considered, as in Figure 3.5. PPMI achieves similar stability, while word2vec lags considerably behind.

To further compare word2vec and GloVe, we look at how the stability of word2vec changes with the frequency of the word and the number of neighbors used to calculate stability. This is shown in Figure 3.6 and is directly comparable to Figure 3.2. Surprisingly, the stability of word2vec varies substantially with the frequency of the word. For lower-frequency words, as the number of nearest neighbors increases, the stability increases approximately exponentially. For high-frequency words, the lowest and highest number of nearest neighbors show the greatest stability. This is different than GloVe, where stability remains reasonably constant across word frequencies, as shown in Figure 3.2. The behav-

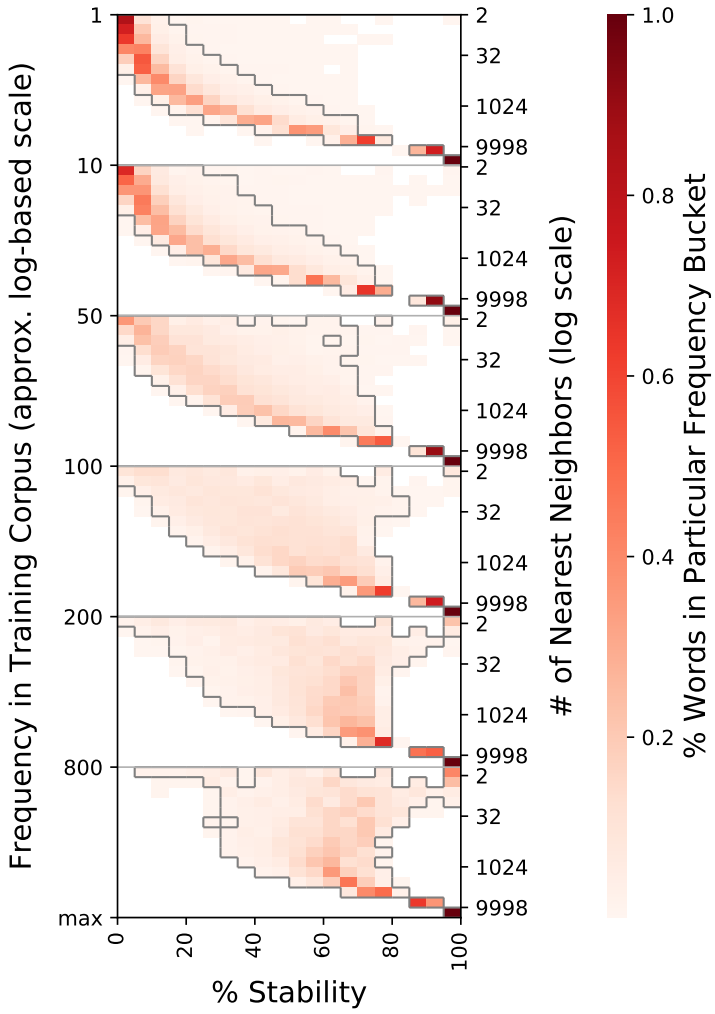


Figure 3.6: Stability of word2vec on the PTB. Stability is measured across ten randomized embedding spaces trained on the training data of the PTB (determined using language modeling splits [84]). Each word is placed in a frequency bucket (left y-axis) and stability is determined using a varying number of nearest neighbors for each frequency bucket (right y-axis). Each row is normalized, and boxes with more than 0.01% of the row's mass are outlined.

ior we see here agrees with the conclusion of Mimno and Thompson [88], who find that GloVe exhibits more well-behaved geometry than word2vec.

Observation 5. Frequency is not a major factor in stability. To better understand the role that frequency plays in stability, we run separate ablation experiments comparing regression models with frequency features to regression models without frequency features. Our current model (using raw frequency) achieves an R^2 score of 0.301. Comparably, a model using the same features, but with normalized instead of raw frequency, achieves a

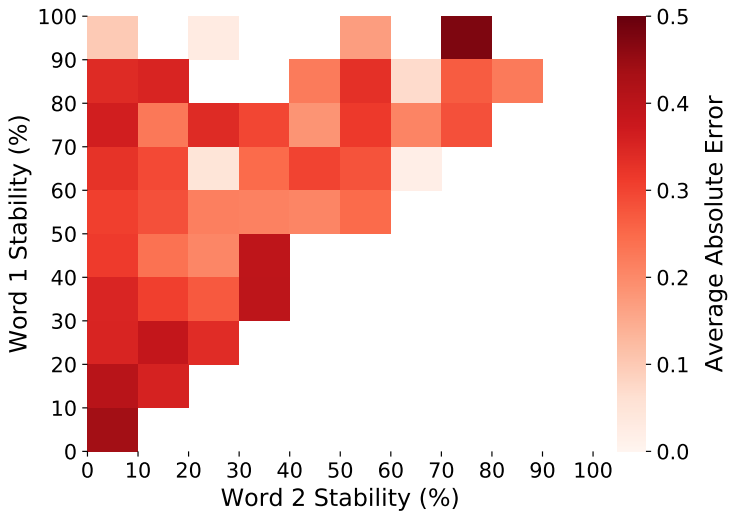


Figure 3.7: Absolute error for word similarity. Word 1 is the more stable word in the pair of words being compared.

score of 0.303. Removing frequency from either regression model gives a score of 0.301. This indicates that frequency is not a major factor in stability, though normalized frequency is a larger factor than raw frequency.

Finally, we look at regression models using only frequency features. A model using only raw frequency features has an R^2 score of 0.008, while a model with only normalized frequency features has an R^2 score of 0.0059. This indicates that while frequency is not a major factor in stability, it is also not negligible. As we pointed out in the introduction, frequency does correlate with stability (Figure 3.1). However, in the presence of all of these other features, frequency becomes a minor factor.

3.4 Impact of Stability on Downstream Tasks

Word embeddings are used extensively as the first stage of neural networks throughout NLP. Typically, embeddings are initalized based on a vector trained with word2vec or GloVe and then further modified as part of training for the target task. We study two downstream tasks to see whether stability impacts performance.

Since we are interested in seeing the impact of word vector stability, we choose tasks that have an intuitive evaluation at the word level: word similarity and POS tagging.

3.4.1 Word Similarity

To model word similarity, we use 300-dimensional word2vec embedding spaces trained on the PTB. For each pair of words, we take the cosine similarity between those words averaged over ten randomly initialized embedding spaces.

We consider three datasets for evaluating word similarity: WS353 (353 pairs) [42], MTurk287 (287 pairs) [103], and MTurk771 (771 pairs) [50]. For each dataset, we normalize the similarity to be in the range [0,1], and we take the absolute difference between our predicted value and the ground-truth value. Figure 3.7 shows the results broken down by stability of the two words (we always consider Word 1 to be the more stable word in the pair). Word similarity pairs where one of the words is not present in the PTB are omitted.

We find that these word similarity datasets do not contain a balanced distribution of words with respect to stability; there are substantially more unstable words than there are stable words. However, we still see a slight trend: As the combined stability of the two words increases, the average absolute error decreases, as reflected by the lighter color of the cells in Figure 3.7 while moving away from the (0,0) data point.

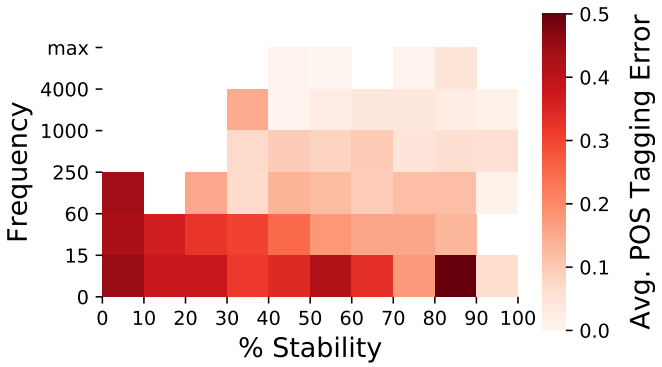
3.4.2 Part-of-Speech Tagging

Part-of-speech (POS) tagging is a substantially more complicated task than word similarity. We use a bidirectional LSTM implemented using DyNet [90]. We train nine sets of 128-dimensional word embeddings with word2vec using different random seeds. The LSTM has a single layer and 50-dimensional hidden vectors. Outputs are passed through a *tanh* layer before classification. To train, we use SGD with a learning rate of 0.1, an input noise rate of 0.1, and recurrent dropout of 0.4.

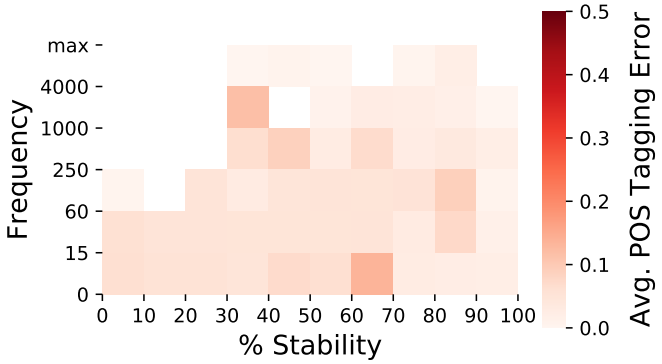
This simple model is not state-of-the-art, scoring 95.5% on the development set, but the word vectors are a central part of the model, providing a clear signal of their impact. For each word, we group tokens based on stability and frequency. Figure 3.8 shows the results.[10] Fixing the word vectors provides a clearer pattern in the results, but also leads to much worse performance: 85.0% on the development set. Based on these results, it seems that training appears to compensate for stability. This hypothesis is supported by Figure 3.8c, which shows the similarity between the original word vectors and the shifted word vectors produced by the training. In general, lower stability words are shifted more during training.

Understanding how the LSTM is changing the input embeddings is useful information

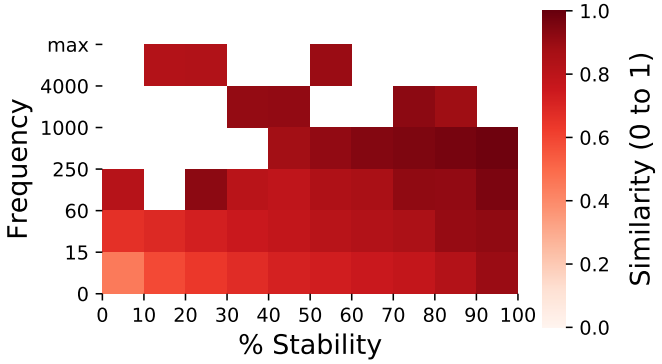
[10]The unusual dark spot that occurs at medium-high stability and low frequency is caused primarily by words that have a substantially different POS distribution in the test data than in the training data.



(a) POS error probability with fixed vectors.



(b) POS error probability when vectors may shift in training.



(c) Cosine similarity between original word vectors and shifted word vectors.

Figure 3.8: Results for POS tagging. The top two graphs show average POS tagging error divided by the number of tokens (darker is more errors) while either keeping word vectors fixed or not during training. The bottom graph shows word vector shift, measured as cosine similarity between initial and final vectors. In all graphs, words are bucketed by frequency and stability.

for tasks with limited data, and it could allow us to improve embeddings and LSTM training for these low-resource tasks.

3.5 Conclusion and Recommendations

Word embeddings are surprisingly variable, even for relatively high frequency words. Using a regression model, we show that domain and part-of-speech are key factors of instability. Downstream experiments show that stability impacts tasks using embedding-based features, though allowing embeddings to shift during training can reduce this effect. In order to use the most stable embedding spaces for future tasks, we recommend either using GloVe or learning a good curriculum for word2vec training data. We also recommend using in-domain embeddings whenever possible.

The code used in the experiments described in this chapter is publicly available from http://lit.eecs.umich.edu/downloads.html, under the heading "Embedding Stability (code)".

CHAPTER 4

Analyzing the Surprising Variability in Word Embedding Stability Across Languages

As we have seen, common English embedding spaces are surprisingly unstable, which has implications for work that uses embeddings as features in downstream tasks, and work that uses embeddings to study specific properties of language [6, 126]. However, research to date on word embedding stability has been exclusively done on English and is not representative of all languages. In this chapter, we explore the stability of word embeddings in languages around the world.

Specifically, we consider how stability varies for different languages and how linguistic properties are related to stability, a previously understudied relationship. Using regression modeling, we capture relationships between linguistic properties and average stability of a language, and we draw out insights about how linguistic features relate to stability. For instance, we find that languages with more affixing tend to be less stable. Our findings provide crucial context for research that uses word embeddings to study language properties and trends (e.g., [53, 55, 1]), which often rely on raw embeddings created by GloVe or word2vec. If these embeddings are unstable, then research using them needs to take this into account in terms of methodologies and error analysis.

4.1 Data

In order to explore the stability of word embeddings in different languages, we work with two datasets, Wikipedia and the Bible. While Wikipedia has more data, the Bible covers more languages. Wikipedia is a comparable corpus, whereas the Bible is a parallel corpus.

4.1.1 Wikipedia Corpus

We use pre-processed Wikipedia dumps in 40 languages taken from Al-Rfou et al. [4].[1] Specifically, the Wikipedia texts that we use have been previously segmented using an OpenNLP probabilistic tokenizer whenever possible,[2] and a Unicode text segmentation model offered by Lucene when no model is available[3] [4]. In order to verify that this word segmentation is reasonable, we asked speakers of several of the languages[4] to look over a subset of the data and describe any errors that they saw. All languages that we checked were confirmed to have reasonable word segmentation, though a few small inconsistencies were observed. In Finnish, several word cases were handled inconsistently, and in Italian and French, determiners followed by words beginning with a vowel were not segmented correctly. However, speakers of these languages confirmed that these inconsistencies are relatively minor and most of the text is well-segmented.

The size of these Wikipedia corpora varies from 329,136 sentences (Tagalog) to 75,241,648 sentences (English), with an average of 9,292,394 sentences. For all of our experiments, we downsample each corpus to work with comparably sized data (details in Section 4.2.2).

4.1.2 Bible Corpus

We consider 97 languages from the Bible corpus [79]:[5] all languages for which at least 75% of the Bible ($\geq$ 23,326 verses) is present.[6] This excludes many languages for which there is only a partial Bible, e.g., just the New Testament, which would be insufficient for training word vectors. We use the pre-processing already done by McCarthy et al. [79]. We consider two sets of languages with the Bible corpus: languages that overlap with the set of Wikipedia languages (26 languages), and all languages in the Bible corpus (97 languages).

4.1.3 WALS

To gain linguistic properties of these languages, we use the World Atlas of Language Structures (WALS),[7] a database of phonological, lexical, and grammatical properties for over

[1] Available online at https://sites.google.com/site/rmyeid/projects/polyglot.

[2] Danish, German, English, French, Dutch, Portuguese

[3] See http://www.unicode.org/reports/tr29/.

[4] Finnish, German, Romanian, Italian, French, English, Arabic

[5] Available by contacting McCarthy et al. [79].

[6] To work with a maximum number of languages, we only consider the complete Protestant Bible (i.e., all of the verses that appear in the English King James Version of the Bible).

[7] Available online at https://wals.info.

2,000 languages [36]. This expert-curated resource contains 192 language features. For example, WALS records subject, object, and verb word order for various languages.

4.2 Calculating Stability in Many Languages

The first part of this chapter is a comparison of stability across languages. Before presenting our measurements, we analyze some important methodological decisions.

4.2.1 The Effect of Downsampling on Stability

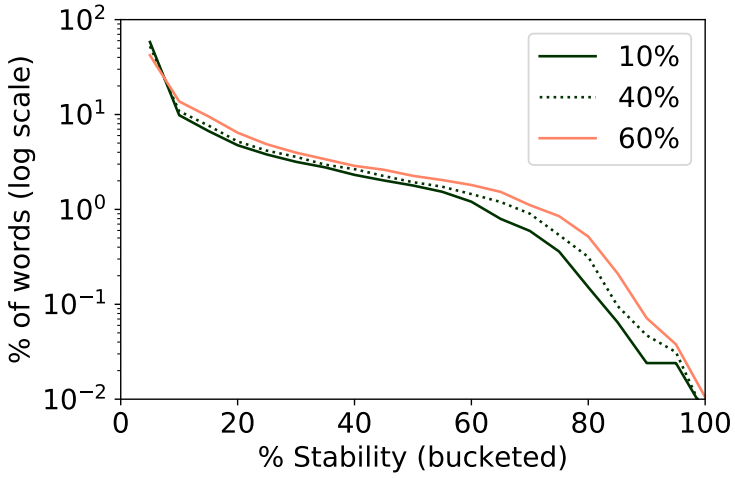
Stability measures how changes to the input data or training algorithm affect the resulting embeddings. Sometimes we make changes with the goal of shifting the embeddings, such as increasing the context window size to try to get embeddings that capture semantics more than syntax. In other cases, we would hope a change would not substantially change our embeddings, such as changing the random seed for the algorithm. For our experiments, we consider a previously unstudied source of instability: different data samples from the same distribution. This is a case where we hope embeddings remain stable, given a sufficiently large sample.

We generate data samples by downsampling a corpus to create multiple smaller corpora; we then measure stability across these downsamples. The choice of sampling with or without replacement, and the size of the sample are subtle methodological choices. In this section, we consider whether stability across downsamples produces consistent results that we can compare across languages.

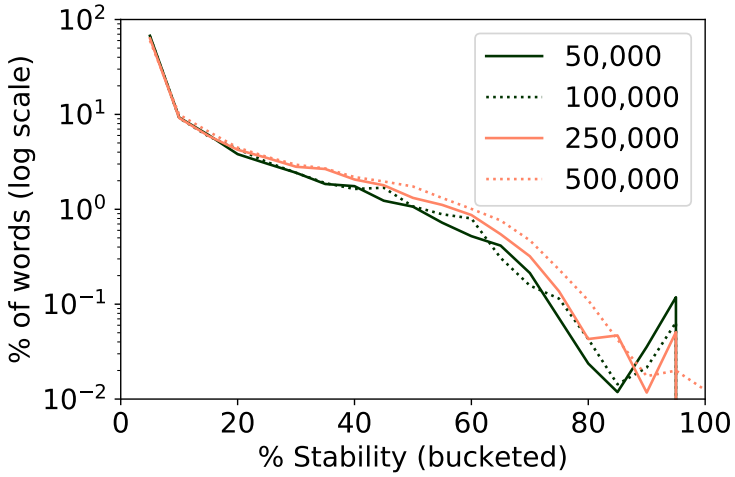
First, we consider downsampling with replacement, shown in Figure 4.1a. We sample five sets of 500,000 sentences multiple times, controlling the amount of overlap between downsamples (from 10% to 60%).[8] For a specific overlap amount $X\%$, $X\%$ of 500,000 sentences is randomly sampled and included in all of the five downsamples. The remaining $(100 - X)\%$ sentences are randomly sampled for each downsample. Stability is calculated using GloVe embeddings and the words that occur in every downsample for every overlap percentage. For GloVe, we use 100 iterations, 300 dimensions, a window size of 5, and a minimum word count of 5.

In Figure 4.1a, we group stability into buckets, allowing us to see patterns in stability that are not visible from a single statistic, such as the overall average. Stability is grouped into buckets of width 5%, e.g., 0-5%, 5-10%, ..., 95-100%. The number of words that fall

[8]Data drawn from an English Wikipedia corpus of 5,269,686 sentences (denoted "Large English Wikipedia"). This data was used in Tsvetkov et al. [122] and is available by contacting the authors of that paper.



(a) Sampling *with* replacement, varying percentage overlap between samples.



(b) Sampling *without* replacement, varying sample size.

Figure 4.1: Measuring the impact of data sampling parameters on stability measurements. Results when sampling *with* replacement consistently increase as overlap increases (a). This poses a problem, as results may reflect corpus size rather than intrinsic stability. Results when sampling *without* replacement do show a consistent pattern, even when the sample is only 50,000 sentences, a tenth of the largest sample size (b).

into each stability bucket are counted, and then normalized by the total number of words. We see that while stability trends are similar for different overlap amounts, stability is consistently higher as the overlap amount increases. This means that if we use downsampling with replacement, we cannot reliably compare stability across multiple corpora of varying sizes (e.g., Wikipedia and the much smaller Bible corpus). The overlap amount would change depending on the size of the corpus, changing our stability measurement.

Instead of downsampling with replacement, we consider downsampling without re-

placement, shown in Figure 4.1b for different downsample sizes. We see that varying the size of the downsample does not have a large effect on the patterns of stability. Particularly when looking at lower stability, the trends are remarkably consistent, even when the downsample size varies from 50,000 sentences to 500,000 sentences. The pattern grows less consistent when looking at higher stability, especially with smaller downsample sizes.

This comparison (Figures 4.1a and 4.1b) shows that downsampling without replacement produces more consistent (and thus comparable) stability results than downsampling with replacement. Thus, we only consider downsampling without replacement.

4.2.2 Stability for Wikipedia and the Bible

Our first study, shown in Figure 4.2, considers stability across the 26 languages included in both Wikipedia and the Bible. These results show three settings for Wikipedia: (1) Stability of GloVe embeddings across five downsampled corpora, (2) Stability of word2vec embeddings across five downsampled corpora, and (3) Stability of word2vec using five different random seeds across one downsampled corpus. For the Bible, we only show the third case, since it is too small for downsampling.

Each downsampled corpus is 100,000 sentences, and words that occur with a frequency less than five are ignored. Previous work [101, 126] has indicated that words that appear this infrequently will be very unstable. We use standard parameters for both embedding algorithms. For GloVe [96], we use 100 iterations, 300 dimensions, a window size of 5, and a minimum word count of 5; these parameters led to good performance in Chapter 3. For word2vec (w2v) [86], we use 300 dimensions, a window size of 5, and a minimum word count of 5. For each embedding, we calculate the ten nearest neighbors of every word using FAISS[9] [63]. Finally, for each language, we calculate the stability for every word in that language across all five embedding spaces.[10]

Figure 4.2 shows bucketed stability for both Wikipedia and the Bible. Visualizing stability in buckets allows us to see patterns in stability that are obscured in a single overall average. Of note in Figure 4.2 is that we do see differences in stability among languages. When considering GloVe downsamples on Wikipedia, Vietnamese is the most stable language (avg. 2.46%), while Korean is the least stable (avg. 0.58%). In general, we see that most of the words (in all languages) are relatively unstable, though Vietnamese has a

[9] We use exact, not approximate, search.

[10] All experiments were run on a machine with four Intel(R) Xeon(R) CPU E5-1603 v3 @ 2.80 GHz processors, each with four cores. Timings: Training one w2v embedding on one Wikipedia corpus, 13 sec.; training one GloVe embedding on one Wikipedia corpus, 12 min., 13 sec.; calculating stability on one Wikipedia corpus, 17 sec.; training one w2v embedding on one Bible corpus, 5 sec.; calculating stability on one Bible corpus, 12 sec.

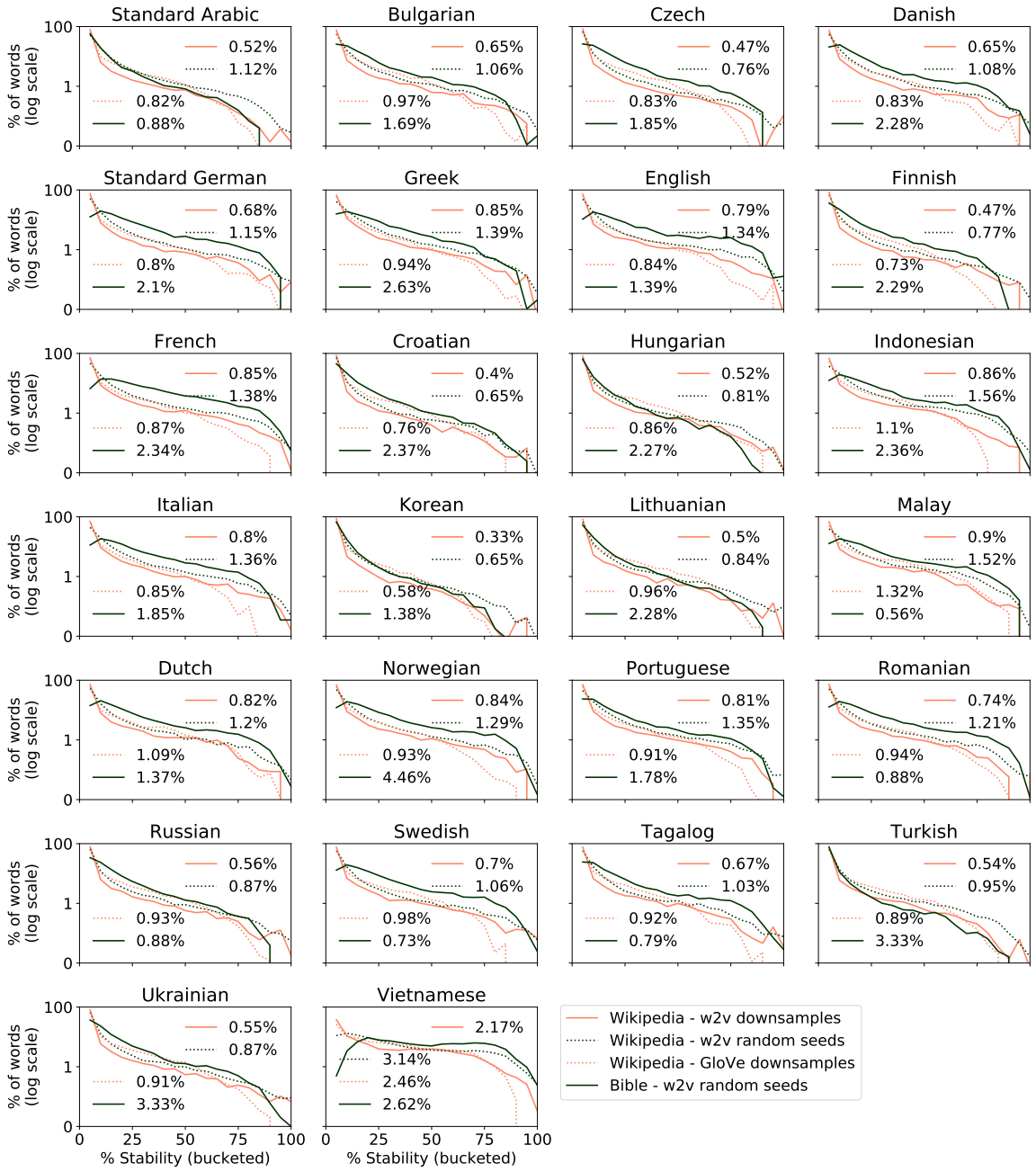
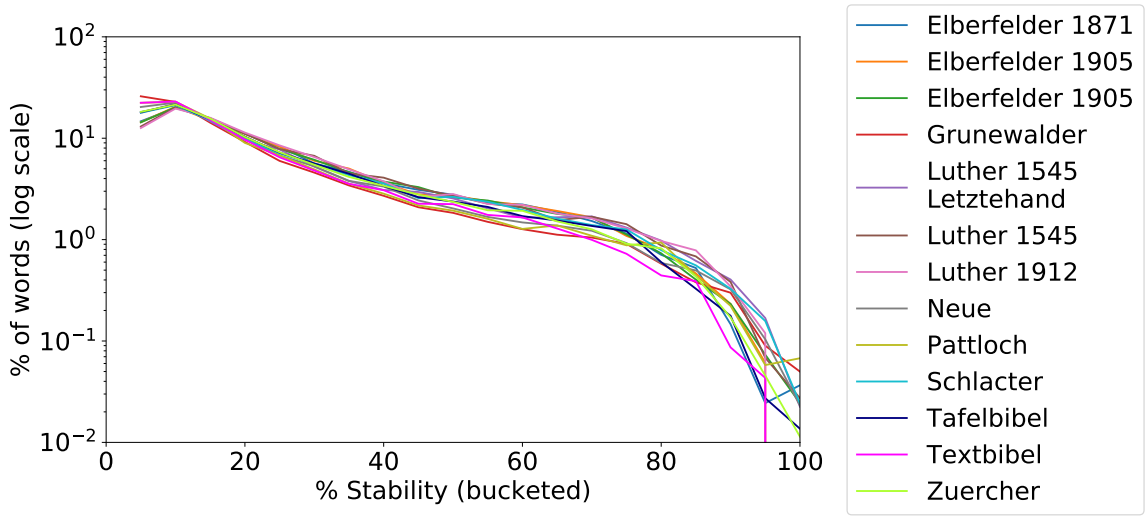


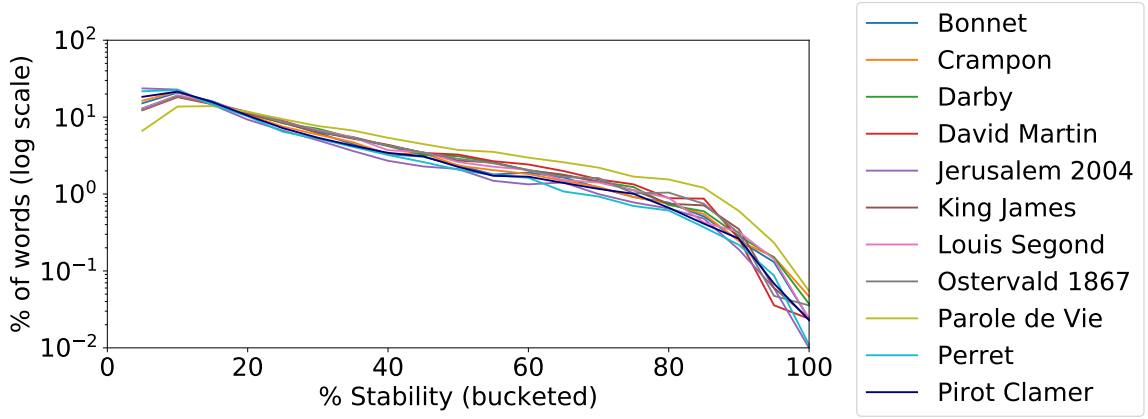
Figure 4.2: Percentage of words that occur in each stability bucket for four different methods, three on Wikipedia and one on the Bible. The 26 languages in common are shown here. The average stability for each method is shown on the individual graphs.

noticeably higher percentage of words that are stable (particularly for the Bible).

Figure 4.2 also shows that all three methods on Wikipedia generally show similar behavior. For further experiments, we use stability with GloVe embeddings across five downsampled corpora. Because the Bible corpus is substantially smaller than the Wikipedia corpus, downsampling to calculate stability is not a feasible option. We see that word2vec



(a) German.



(b) French.

Figure 4.3: Percentage of words that occur in each stability bucket for different Bible translations in German and French.

with a single downsample and five different random seeds gives comparable stability results to using GloVe across five downsamples. Based on those results, we choose to use w2v with varying seeds to calculate stability for the Bible.

Several languages have multiple Bible translations. This provides us with a way to check the consistency of our results. Figure 4.3 shows that stability patterns are very consistent. The French Parole de Vie translation (top line in yellow in Figure 4.3b) intentionally uses simpler, everyday language, which could explain why this line follows a different pattern than the other French translations. For further experiments on languages with multiple Bible translation, we choose the Bible translation with the highest average stability.

Here, we have compared trends in stability across 26 languages and explored the con-

sistency of results using various methods. For the rest of this paper, we use results from GloVe across five downsampled corpora for Wikipedia, and results across five random seeds of word2vec for the Bible.

4.3 Regression Modeling

To further explore the results from the previous section, we now examine linguistic factors that correlate with stability. In order to draw conclusions about specific linguistic features, we use a ridge regression model [57][11] to predict the average stability of all words in a language given features reflecting language properties. We experiment with different regularization strengths and use the best-performing value ($\lambda = 10$).[12] We choose to use a linear model here because of its interpretability. While more complicated models might yield additional insight, we show that there are interesting connections to be drawn from a linear model.

4.3.1 Model Input and Output

Our model takes linguistic features of a language as input and predicts stability as output. Since WALS properties are categorical, we turn each property into a set of binary features. If a particular language does not have a known value for a given WALS property, then all of these binary features are marked zero. To prevent overfitting, we remove all binary features with fewer than five languages. Note that because all of our input features are binary, all weights are easily comparable.

For each model, we bootstrap over the input features $N = 1,000$ times, allowing us to calculate standard error for both the R^2 score and the model weights. Calculating significance for each feature allows us to discard highly variable weights and focus on features that consistently contribute to the regression model, giving us more confidence in the results.

In order to draw out important correlations between linguistic features and stability, we filter the languages and WALS properties that we consider. We only include languages that have at least 25% of all WALS properties. Then, we only consider WALS properties that cover at least 25% of the filtered languages. Each WALS property has a subset of binary

[11] Run using the Python package `sklearn.linear.model.Ridge` [95] with default parameters except $\alpha = 10$.

[12] We run leave-one-language-out cross-validation, described in Section 4.3.2, using the regularization strength α values of 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, and 1000, choosing the α value with the lowest average absolute error.

features. We remove all WALS properties that do not have at least two features that include at least five languages. After this filtering, we end up with 37 languages,[13] and 97 WALS properties.

We also group highly correlated WALS features together. We create the groupings by combining features that have a Pearson correlation greater than 0.8. A feature is included in a particular grouping if it correlates highly with any of the features already in the group. Each grouped feature is marked as one if *any* of the included features are marked as one.

The output of our model is the average stability of a language, which is calculated by averaging together the stability of all of the words in a language. If a language is present in both corpora, we average the stabilities from the two corpora.

4.3.2 Evaluation

We evaluate our model in two ways. First, we measure goodness of fit using the coefficient of determination R^2.[14] Second, we run leave-one-out cross-validation across all languages, and report absolute error on the left-out language. We compare this to a baseline of choosing the average stability over all training languages.

We use the individual feature weights to measure how much a particular feature contributes to the overall model. When reporting weights, we train the model using all 37 languages. Because we are primarily using regression modeling to learn associations between certain features and stability, no test data are necessary. The emphasis is on the model itself and the feature weights it learns, not on the model's performance on a task.

4.4 Results and Discussion

Our regression model has a high R^2 score of 0.96 ± 0.00, indicating that this model fits the data well.[15] Significant weights with the highest magnitude are shown in Table 4.1. Running leave-one-out cross-validation across all languages, we get an average absolute error of 0.62 ± 0.53.[16] This is substantially better than the baseline of choosing the average stability over all languages, which gives an average absolute error of 0.86 ± 0.55.

[13] Bengali, Bulgarian, Cherokee, Comanche, English, Estonian, Finnish, Haitian, Haitian Creole, Hebrew, Hindi, Hmong Njua, Hungarian, Indonesian, Italian, Japanese, Korean, Latin, Latvian, Linda, Lithuanian, Ma'di, Mam, Mandarin, Maybrat, Norwegian, Persian, Pohnpeian, Polish, Portuguese, Russian, Somali, Spanish, Swedish, Thai, Turkish, Ukrainian, Vietnamese

[14] Measured using the Python package `sklearn.linear.model.Ridge.score`.

[15] Run on a 2.9 GHz Dual-Core Intel Core i5 in < 7 sec.

[16] Run on a 2.9 GHz Dual-Core Intel Core i5 in < 4 sec. Cross-validation has an average R^2 score of 0.92 on the training data.

Cat.	WALS Attribute	Weight
VC, M	*Suffixing Grouping:* ·Prefixing vs. Suffixing in Inflectional Morphology: Strongly Suffixing; ·Position of Tense-Aspect Affixes: Tense-aspect suffixes	-0.14 ± 0.0
L	Hand and Arm: Different	-0.11 ± 0.0
CS	Relativization on Obliques: Gap	-0.10 ± 0.0
VC	Overlap between Situational & Epistemic Modal Marking: Overlap for both possibility & necessity	-0.09 ± 0.0
NC	Ordinal Numerals: First, second, three-th	-0.08 ± 0.0
NC	Comitatives and Instrumentals: Differentiation	-0.08 ± 0.0
P	Rhythm Types: Trochaic	-0.08 ± 0.0
WO	Order of Adjective and Noun: Adjective-Noun	-0.07 ± 0.0
WO	Order of Adposition and Noun Phrase: Postpositions	-0.07 ± 0.0
NC	*No Gender Grouping:* · Systems of Gender Assignment: No gender; · Sex-based and Non-sex-based Gender Systems: No gender; · Gender Distinctions in Independent Personal Pronouns: No gender distinctions; · Number of Genders: None	0.05 ± 0.0
P	Voicing and Gaps in Plosive Systems: Other	0.06 ± 0.0
M	Prefixing vs. Suffixing in Inflectional Morphology: Little affixation	0.06 ± 0.0
CS	'Want' Complement Subjects: Subject is expressed overtly	0.06 ± 0.0
VC	The Morphological Imperative: No second-person imperatives	0.06 ± 0.0
CS	Purpose Clauses: Balanced	0.06 ± 0.0
WO	*Prepositions Grouping:* ·Order of Adposition and Noun Phrase: Prepositions; ·Relationship between the Order of Object and Verb and the Order of Adposition and Noun Phrase: VO and Prepositions	0.06 ± 0.0
WO	Order of Demonstrative and Noun: Noun-Demonstrative	0.07 ± 0.0
NC	Position of Case Affixes: No case affixes or adpositional clitics	0.11 ± 0.0

Table 4.1: Weights with the highest magnitude in the regression model. Negative weights correspond with low stability, and positive weights correspond with high stability.

Table 4.2 breaks down the regression results by broad WALS category, listing both the number of binary features per category, as well as the average magnitude of weights for features in that category. The two most important groups of features are Nominal Categories and Verbal Categories. Both of these categories have a large number of features and a high average magnitude. While the Lexicon category has a high average magnitude, it contains very few features. To further explore these results, we highlight a few WALS property in more detail.

WALS Category	Num. Features	Avg. Magnitude
Simple Clauses (SC)	30	0.019
Nominal Syntax (NS)	2	0.021
Other (O)	2	0.023
Complex Sentences (CS)	11	0.028
Morphology (M)	18	0.031
Word Order (WO)	32	0.031
Phonology (P)	21	0.032
Nominal Categories (NC)	40	0.036
Verbal Categories (VC)	27	0.036
Lexicon (L)	6	0.039

Table 4.2: Number of binary features and average magnitude of weights in the regression model for different WALS categories. Grouped features are included in each category that they cover.

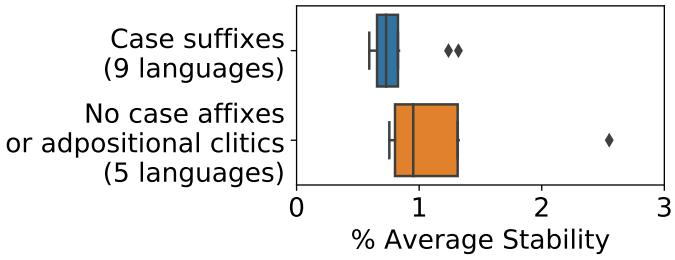
4.4.1 Suffixes and Prefixes

Table 4.1 shows that three of the top features are related to affixes (suffixes and prefixes). Specifically, we include three main WALS properties that deal with affixes: Position of Case Affixes [33], Prefixing vs. Suffixing in Inflectional Morphology [35], and Position of Tense-Aspect Affixes [34]. Distributions of these features in the 37 languages used for the regression model are shown in Figure 4.4 (categories with fewer than five languages are not shown).

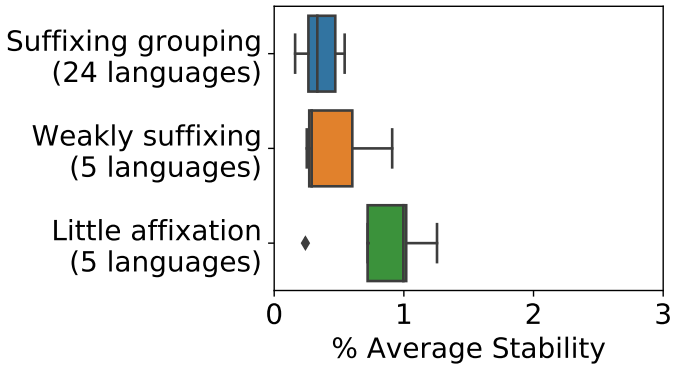
For all three of these properties, more affixing is associated with lower stability. When considering word embeddings, this result makes intuitive sense. Affixes cause there to be many different word variations (e.g., walk, walked, walking, walker), which may not be handled consistently by the embedding algorithm, leading to lower average stability.

4.4.2 Gendered Languages

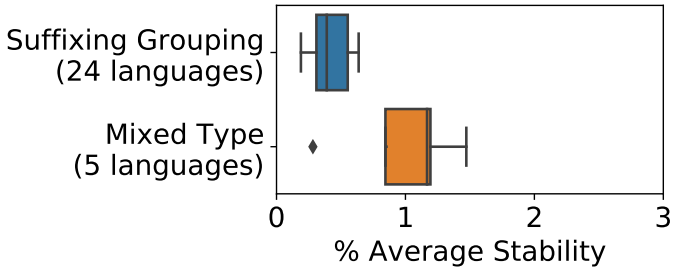
Table 4.1 also highlights a grouping of WALS properties related to whether a language is gendered or not. Four WALS properties are relevant to this: Systems of Gender Assignment [28], Sex-based and Non-sex-based Gender Systems [27], Gender Distinctions in Independent Personal Pronouns [111], and Number of Genders [26]. In general, a language is considered to have a gender system if different parts-of-speech are required to agree in gender (as opposed to simply having gendered nouns). Distributions of these features are shown in Figure 4.5.



(a) Position of Case Affixes.



(b) Prefixing vs. Suffixing in Inflectional Morphology.



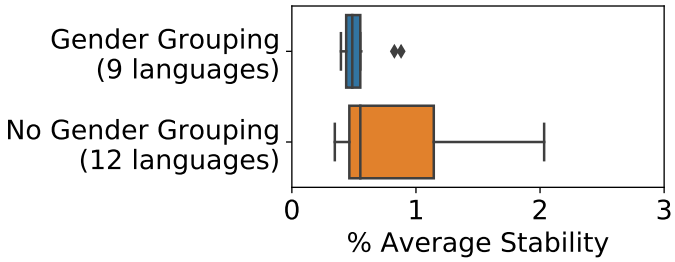
(c) Position of Tense-Aspect Affixes.

Figure 4.4: Affixing properties compared using box-and-whisker plots.

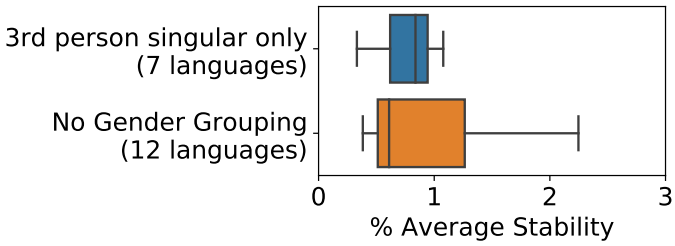
For all of these properties, languages with no gender system tend to have higher average stability. Again, this result makes sense in the context of word embeddings. Languages with gender systems will have more word forms (e.g., both male and female word forms), which may not be handled consistently by the embedding algorithm.

4.5 Conclusion

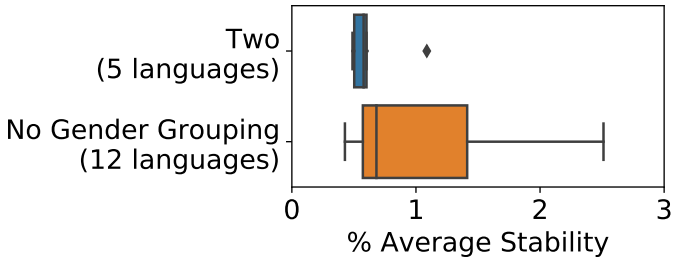
In this chapter, we considered how stability varies across different languages. This work is important because algorithms such as GloVe and word2vec continue to be effective methods in a wide variety of scenarios [7], particularly the computational humanities and lan-



(a) Systems of Gender Assignment; Sex-based and Non-sex-based Gender Systems (*Gender Grouping:* No gender; Sex-based).



(b) Gender Distinctions in Independent Personal Pronouns.



(c) Number of Genders.

Figure 4.5: Gender properties compared using box-and-whisker plots.

guages where large corpora are not available. We have shown that most languages have unstable word embeddings. Because of this, when using embeddings to study language, it is important to not rely on a single embedding space, but rather to consider multiple embedding spaces, with slight variations in the training of the embedding spaces.

We studied the relationship between linguistic properties and stability, something that has been previously understudied. We drew out several aspects of this relationship, including that languages with more affixing tend to be less stable, and languages with no gender systems tend to be more stable. These insights can be used in future work to inform the design of embeddings in many languages. For instance, future architectures could be designed specifically for gendered languages, or for languages with high amounts of affixing, taking into account the linguistic properties particular to these languages. As we saw with languages with high affixing, the morphology of a language is important for designing word

embeddings. There has already been some current research on designing morphologically-aware word embeddings (e.g., [22, 29, 5]), and our analyses in this chapter suggest that this is a fruitful avenue of research to continue. Particularly for languages that are highly unstable, the existing word representations that we use are not effective, and there is a need to come up with new representations that will work well for these languages.

The code used in the experiments described in this chapter will be publicly available from http://lit.eecs.umich.edu/downloads.html.

CHAPTER 5

To Batch or Not to Batch? Comparing Batching and Curriculum Learning Strategies Across Tasks and Datasets

When designing architectures for tasks in natural language processing (NLP), certain methodological questions arise, such as how to order the data during training (curriculum learning), what batching method to use, and what batch size to use. In Chapter 3, we first looked at how the curriculum of a dataset affects the stability of word embeddings, and we saw that there is a trend between where a word first appears in the training data and its stability. In this chapter, we expand on this by analyzing curriculum and batching choices for the training of word vectors, evaluated through three downstream tasks: text classification, sentence and phrase similarity, and part-of-speech tagging. We consider a variety of datasets in order to understand how these strategies work across diverse tasks and data. We show that for some tasks, these decisions do not have a significant impact, but for other tasks, such as text classification on small datasets, these decisions are important consider-

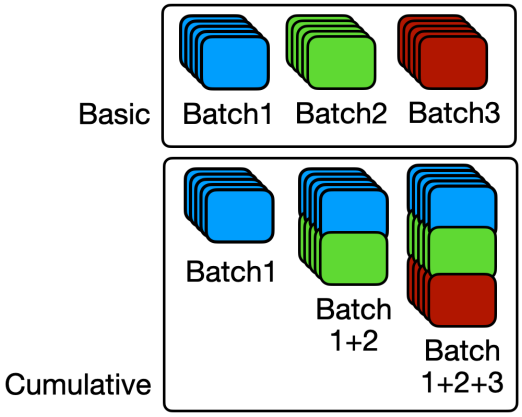


Figure 5.1: Basic v. cumulative batching. Rectangles represent chunks of the training data, with different colors representing different sections of the data.

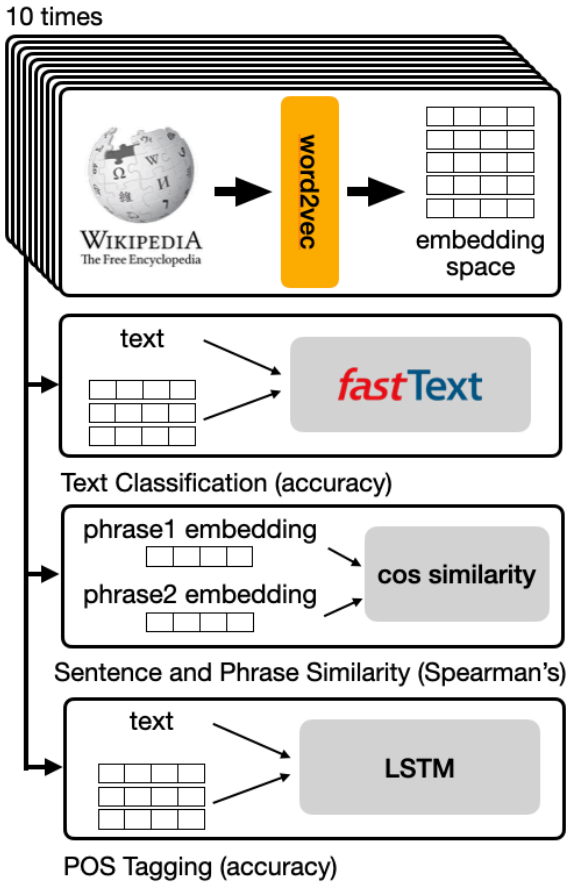


Figure 5.2: Experimental setup for text classification, sentence and phrase similarity, and POS tagging.

ations. Building on our analysis in Chapter 3, we additionally consider how batching and curriculum learning strategies affect word embedding stability.

Batching. The first methodological decision we consider is batching. We focus on two techniques shown in Figure 5.1 and described in more detail in Section 5.1.1. These techniques were first proposed for synthetic vision and word representation learning tasks [12] and unsupervised dependency parsing [116]. Varying the number of batches has also been studied; Smith et al. [113] show that choosing a good batch size can decrease the number of parameter updates to a network.

Curriculum Learning. The second decision we study is curriculum learning. Curriculum refers to the order that the data is presented to the embedding algorithm, which has some effect on the performance of the created embeddings [122]. Curriculum learning has also been applied to other tasks in NLP, including relation extraction [76] and sentiment analysis [24].

5.1 Experimental Setup

We consider the downstream tasks of text classification, sentence and phrase similarity, and part-of-speech (POS) tagging, shown in Figure 5.2. These were chosen because they have varying degrees of complexity - from word similarity, where word embeddings are compared using cosine similarity, to POS tagging, where embeddings are input to an LSTM. We also analyze the stability of the resulting embedding spaces.

5.1.1 Initial Embedding Spaces

For each task, we begin with a dataset of sentences from Wikipedia (5,269,686 sentences; 100,003,406 words; 894,044 tokens)[1] and create word2vec skip-gram embeddings [86].[2] These embeddings are then input to an architecture suited for the particular task that we are considering. In order to account for variability in embeddings [6, 126], we train ten embedding spaces using different random seeds[3] for word2vec. We use these ten spaces to calculate average performance and standard deviation.

5.1.2 Curriculum Learning

When creating embeddings, we consider three different curriculums for the data: the *default* order of Wikipedia sentences, *descending* order by sentence length (longest to shortest), and *ascending* order by sentence length (shortest to longest). Note that curriculum learning only applies to the Wikipedia dataset used to create the word embeddings, rather than the task-specific datasets used for training.

Qualitatively looking at Wikipedia sentences ordered by length, both the shortest and the longest sentences tend to be unnatural sounding. The shortest sentences are only a single token, such as a single word or a single punctuation mark. Some of these are most likely the result of incorrect sentence tokenization. The longest sentences tend to be either run-on sentences, or lists of a large number of items. For instance, the longest sentence is 725 tokens long, and it lists numerical statistics for different countries. This unnaturalness may adversely affect the embedding algorithm when using either the ascending or descending curriculum. It is possible that a more complex ordering of the data would achieve better performance; we leave this exploration to future work.

[1] This data was used in Tsvetkov et al. [122] and is available by contacting the authors of that paper.
[2] We use 300-dimension embeddings and a context window size of 5.
[3] 2518, 2548, 2590, 29, 401, 481, 485, 533, 725, 777

Dataset	# Train Sentences	# Test Sentences	# Classes
Amazon Review (polarity)	$3.6e6$	$4e5$	2
Amazon Review (full)	$3e6$	$6.5e5$	5
Yahoo! Answers	$1.4e6$	$6e4$	10
Yelp Review (full)	$6.5e5$	$5e4$	5
Yelp Review (polarity)	$5.6e5$	$3.8e4$	2
DBPedia	$5.6e5$	$7e4$	14
Sogou News	$4.5e5$	$6e4$	5
AG News	$1.2e5$	$7,600$	4
Open Domain Deception	$5,733$	$1,435$	2
Personal Email	260	89	2
Real Life Deception	96	25	2

Table 5.1: Data statistics for text classification. The first eight datasets are from Zhang et al. [133].

5.1.3 Batching

As part of creating word2vec embeddings, we also consider batching the Wikipedia dataset input to the embedding algorithm. We batch words (and their contexts) using two strategies: basic batching and cumulative batching, visualized in Figure 5.1. For each batching strategy, we consider different numbers of batches ranging exponentially between 2 and 200.

Basic Batching. As described in Bengio et al. [12], we split the data up into X disjoint batches. Each batch is processed sequentially, and each batch is run for n epochs (Batch 1 runs for n epochs, then Batch 2 runs for n epochs, etc.). Once a batch is finished processing, it is discarded and never returned to. Both X and n are hyperparameters.

Cumulative Batching. Our second batching strategy [116] begins in the same way, with the data split up into X disjoint batches. In this strategy, the batches are processed cumulatively (Batch 1 is run for n epochs, then Batches 1 and 2 combined are run for n epochs, etc.).

5.1.4 Task 1: Text Classification

Given the input embedding spaces, the first task we consider is text classification: deciding what category a particular document falls into. We evaluate on eleven datasets, shown in Table 5.1.[4] These datasets span a wide range of sizes (from 96 sentences to 3.6 million training sentences), as well as number of classes to be categorized (from 2 to 14).

Of particular note are three datasets that are at least an order of magnitude smaller

[4]For all tasks, sentences are tokenized using NLTK's Tokenizer.

Dataset	# Pairs (Train)	# Pairs (Test)	# Tokens (Train)
Human Activity	1,373	1,000	1,446
STS Benchmark	5,749	1,379	14,546
SICK	4,439	4,906	2,251

Table 5.2: Data statistics for sentence and phrase similarity.

than the other datasets. These are the Open Domain Deception Dataset [97],[5] and the Real Life Deception Dataset [98],[6] both of which classify statements as truthful or deceptive, as well as the Personal Email Dataset [75], which classifies e-mail messages as personal or non-personal.[7]

After creating embedding spaces, we use fastText [65] for text classification.[8] Performance is measured using accuracy.

5.1.5 Task 2: Sentence and Phrase Similarity

The second task that we consider is sentence and phrase similarity: determining how similar two sentences or phrases are. We consider three evaluation datasets, shown in Table 5.2. The Human Activity Dataset [128] consists of pairs of human activities with four annotated relations each (similarity, relatedness, motivational alignment [MA], and perceived actor congruence [PAC]).[9] The STS Benchmark [19] has pairs of sentences with semantic similarity scores,[10] and the SICK dataset [13] has pairs of sentences with relatedness scores.[11]

For each pair of phrases or sentences in our evaluation set, we average the embeddings for each word, and take the cosine similarity between the averaged word vectors from both phrases or sentences. We compare this with the ground truth using Spearman's correlation [115].

[5]Data available from https://lit.eecs.umich.edu/downloads.html, under "Open-Domain Deception."
[6]Data available from https://lit.eecs.umich.edu/downloads.html, under "Real-life Deception."
[7]Data available from https://lit.eecs.umich.edu/downloads.html, under "Summarization and Keyword Extraction from Emails."
[8]Available online at https://fasttext.cc/.
[9]Data available from https://lit.eecs.umich.edu/downloads.html, under "Human Activity Phrase Data."
[10]Data available from https://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark.
[11]Data available from https://wiki.cimec.unitn.it/tiki-index.php?page=CLIC.

Dataset	# Sentences (Train)	# Sentences (Test)
UD Answers	2,631	438
UD Email	3,770	606

Table 5.3: Data statistics for POS tagging.

5.1.6 Task 3: Part-of-Speech Tagging

Our final task is part-of-speech (POS) tagging: determining the correct part-of-speech for a given word in a sentence. For evaluation, we use two datasets, the email and answers datasets from the English Universal Dependencies Corpus (UD) [92],[12] shown in Table 5.3.

After creating embedding spaces, we use a bi-directional LSTM implemented using DyNet [90] to perform POS tagging. The LSTM has 1 layer with hidden dimension size of 50, and a multi-layer perceptron on the output. Performance is measured using accuracy.

5.1.7 Stability

After looking at performance on these three downstream tasks, we additionally measure the stability of the resulting embedding spaces. We look at the stability distribution across all the words in the embedding space.

5.2 Results

We now consider how each task performs with different curriculum and batching strategies, in order to determine which strategies are most effective.

5.2.1 Task 1: Text Classification

For results on the larger text classification datasets ($>$ 120,000 training sentences), there are no substantial differences between different curriculum and batching strategies. However, we do see differences for the three smallest datasets, shown in Figure 5.3. To compare across datasets of different sizes, we show the number of sentences per batch, rather than number of batches. Because these graphs show many combinations of curriculum and batching strategies, we report numbers on each dataset's dev set.

On the smallest dataset, Real Life Deception (96 training sentences), we see that above approximately ten batches, ascending curriculum with cumulative batching outperforms the

[12]Data available from https://universaldependencies.org/.

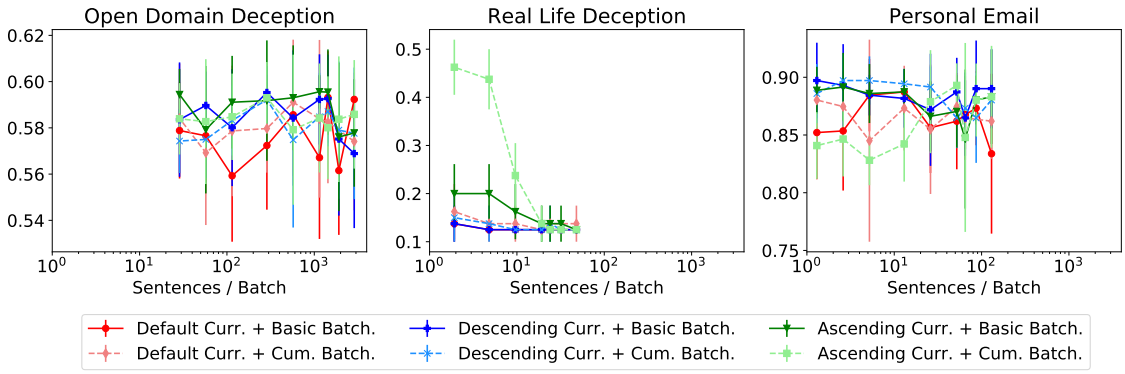


Figure 5.3: Accuracy scores on the development set for three text classification datasets. Different lines indicate models trained with different curriculums and batching strategies (basic, cumulative). Datasets span different ranges of the x-axis because they are different sizes.

other methods. On the test set, we compare our best strategy (ascending curriculum with cumulative batching) with the baseline setting (default curriculum with basic batching), both with 100 batches, and we see no significant difference. This is most likely because the test set is so small (25 sentences).

5.2.2 Task 2: Sentence and Phrase Similarity

Next, we consider results from the sentence and phrase similarity task, shown in Figure 5.4. Because these graphs show many combinations of curriculum and batching strategies, we report numbers on each dataset's train set (we use the train sets rather than the dev sets because there is no training or hyperparemeter tuning).

First, we note that the relative performance of different strategies remains consistent across all three datasets and across all six measures of similarity. An ascending curriculum with cumulative batching performs the worst by a substantial amount, while a descending curriculum with cumulative batching performs the best by a small amount. As the number of sentences per batch increases, the margin between the different strategies decreases. On the test set, we compare our best strategy (descending curriculum with cumulative batching) with the baseline setting (default curriculum with basic batching), and we see in Table 5.4 that the best strategy significantly outperforms the baseline with five batches.

For all six measures, we observe a time v. performance trade-off: The fewer sentences are in a batch, the better the performance is, but the more computational power and time it takes to run.

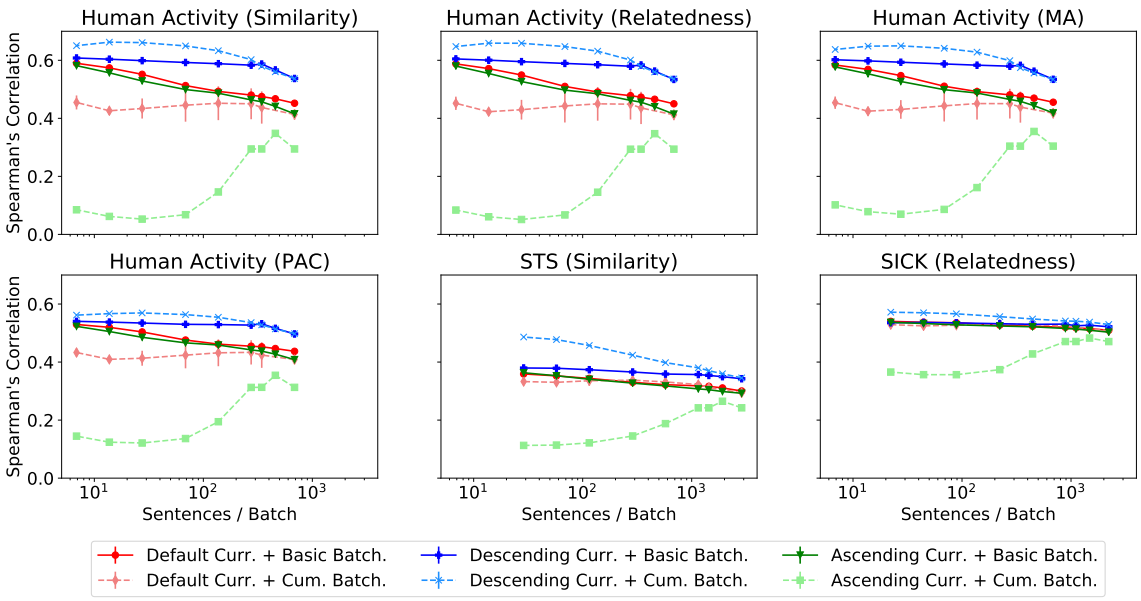


Figure 5.4: Spearman's correlation scores on the train set for sentence and phrase similarity tasks. Different lines indicate models trained with different curriculums and batching strategies (basic, cumulative). Datasets span different ranges of the x-axis because they are different sizes.

	Human Activity					
Dataset	Sim.	Rel.	MA	PAC	STS	SICK
Baseline (Default Curr. + Basic Batch.)	0.36	0.33	0.33	0.22	0.27	0.51
Best (Descending Curr. + Cum. Batch.)	0.43	0.41	0.41	0.29	0.32	0.53

Table 5.4: Spearman's correlation on the test set for similarity tasks (all have a standard deviation of 0.0). Both the baseline method and the best method have a batch size of five.

5.2.3 Task 3: Part-of-Speech Tagging

Finally, there are no significant differences in POS tagging between batching and curriculum learning strategies.

5.2.4 Stability

In addition to these three downstream tasks, we also consider the stability of the embedding spaces produced by different curriculum learning and batching combinations, shown in Figure 5.5. We see very little difference in stability for different curriculums, for either batching strategy. For basic batching, there are also no differences in stability, but for cumulative batching, we see that stability consistently increases as we increase the number of batches. This is consistent with what we saw for text classification and sentence similarity;

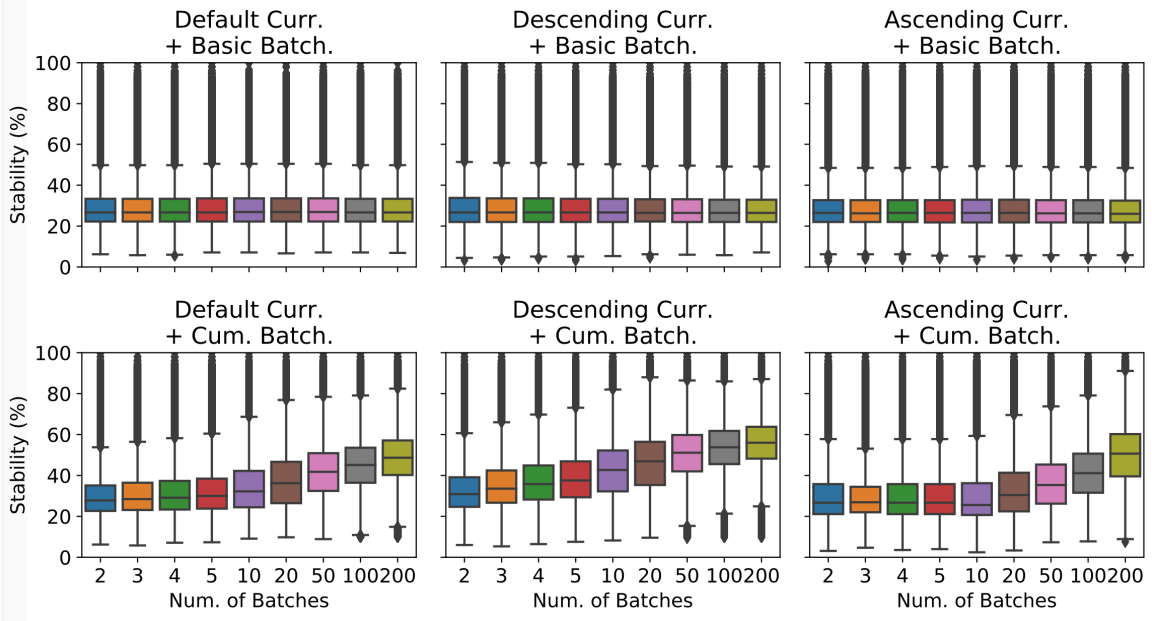


Figure 5.5: Box-and-whisker plots showing stability scores on the embedding spaces produced by different curriculum learnings and batching strategies (basic, cumulative).

cumulative batching performs better than basic batching.

5.3 Discussion and Conclusion

One strategy does not perform equally well on all tasks. On same tasks, such as POS tagging, the curriculum and batching strategies that we tried have no effect at all. Simpler tasks that rely most heavily on word embeddings, such as sentence similarity and text classification with very small datasets, benefit the most from fine-tuned curriculum learning and batching.

In general, cumulative batching outperforms basic batching. This is intuitive because cumulative batching sees the same training example more times than basic batching, and overall sees the training data more times. As the number of sentences per batch increases, the differences between cumulative and basic batching shrink.

It is inconclusive what the best curriculum is. For text classification, the ascending curriculum works best, while for sentence and phrase similarity, the descending curriculum works best. The three curriculums that we experimented with in this chapter (default, ascending, and descending) are relatively simple ways to order data, and more work is needed to investigate more complex orderings. Taking into account properties such as the readability of a sentence, the difficulty level of words, and the frequency of certain part-

of-speech combinations could create a better curriculum that consistently works well on a large variety of tasks. Additionally, artificially simplifying sentences (e.g., substituting simpler words or removing unnecessary clauses) at the beginning of the curriculum, and then gradually increasing the difficulty of the sentences, could be helpful for "teaching" the embedding algorithm to recognize more and more complex sentences.

We have explored different combinations of curriculum learning and batching strategies across three different downstream tasks, as well as looked at the stability of the resulting embedding spaces. We have shown that for different tasks, different strategies are appropriate, but that overall, cumulative batching performs better than basic batching. Since our experiments demonstrate that certain curriculum learning and batching decisions do increase performance substantially for some tasks, for future experiments we recommend that practitioners experiment with different strategies, particularly when the task at hand relies heavily on word embeddings.

The code used in the experiments described in this chapter will be publicly available from http://lit.eecs.umich.edu/downloads.html.

CHAPTER 6

Using Paraphrases to Understand Properties of Contextualized Output Embeddings

Chapters 3, 4, and 5 have focused on context-free output embedding algorithms, such as word2vec and GloVe, both in English and approximately 100 world languages. In this chapter, we shift our focus to English contextualized output algorithms, which have risen in popularity recently. We concentrate on BERT [31], an algorithm which has achieved impressive performance on a wide variety of tasks and has been incorporated into the Google search algorithm.[1] BERT has achieved such rapid success that many current research efforts are focused on improving its architecture further (e.g., [134, 73, 74]).

Because this algorithm has received widespread adoption, we are interested in analyzing it further to better understand its properties and behaviors. Similar to looking at stability in context-free output embeddings, we want to understand how contextualized output embeddings shift under different circumstances. However, our definition of stability does not work in a contextualized output embedding space, since it relies on a single word having a fixed set of nearest neighbors. Instead, in order to understand how and when contextualized output embeddings change, we utilize a unique source of data: paraphrases, which naturally capture properties of word and phrase semantics. As far as we know, we are the first to use paraphrases to better understand contextualized output word embeddings.

Paraphrases encode rich information about the words and phrases in the paraphrase. If two phrases paraphrase each other and we know the word alignment between these phrases, we can conclude that aligned words in these phrases have similar semantics. For example, consider the paraphrases `the goals of the world summit` and `the objectives of the world summit`, where `goals` and `objectives` are aligned. We can conclude that these two words have similar meanings in these contexts. Paraphrases also place constraints on the semantics of polysemous words.

[1] See https://blog.google/products/search/search-language-understanding-bert.

In the paraphrases `it notes with satisfaction that` and `the committee notes with appreciation that`, the word `notes` in both phrases is aligned. This is a highly polysemous word (13 WordNet synsets), but because these two phrases are paraphrases, we can assume that the same sense of the word is being used in both phrases. If we input both of these phrases to an embedding algorithm, we would expect the embeddings of `notes` in the first and second phrase to be close to each other. If they are not, then the embedding algorithm has not properly understood the semantics of that word.

We begin this chapter by examining BERT's ability to correctly understand the semantics of paraphrases. We then use the unique properties of paraphrases to investigate other properties of BERT. We focus on specific groups of words, including polysemous words, stopwords, and punctuation; we also examine previously reported results about how BERT contextualizes certain words.

6.1 Data

We use the Paraphrase Database as our primary source of paraphrases, additionally using the Microsoft Research Paraphrase Corpus to validate some results.

6.1.1 The Paraphrase Database

The Paraphrase Database (PPDB) [45, 93] is an automatically constructed database of paraphrases collected using the bilingual pivoting method [11]. The intuition behind this collection method is that two English strings that can be translated to the same foreign language string are paraphrases of each other (this can be extended to languages other than English). PPDB 2.0 contains 100m+ English paraphrases, each with word alignment information, an automatically generated quality rating, and, for a subset, a human quality rating.[2] Word alignment is a by-product of the machine translation process used in the bilingual pivoting. Formally, a weighted synchronous context-free grammar [3, 21] is used to align words in translation. Example paraphrases with their average human annotations and automatically generated scores are shown in Table 6.1. In general, the phrases in this dataset are short, as shown in Figure 6.1. The longest phrases have six tokens, and the majority have fewer than six.

The manual quality ratings are produced by sampling 26,455 paraphrase pairs from the PPDB and gathering five human annotations for the quality of the paraphrase. Agreement

[2] Available online at http://paraphrase.org.

Phrase 1	Phrase 2	Human Score	PPDB Score
are you talking	do n't they	1.0	2.7
see what 's happening	're seeing	2.0	3.2
internal and foreign	internal and external	3.4	2.1
what 's this all about ?	what 's she saying ?	4.2	3.9
where did they come from ?	where are they from ?	4.8	4.4

Table 6.1: Example tokenized paraphrases from the PPDB, with their average human annotations and automatic PPDB scores.

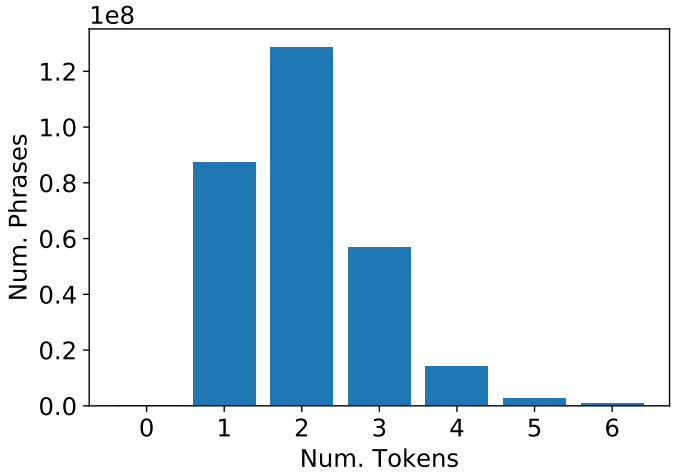


Figure 6.1: Distribution of phrase lengths in the PPDB.

is measured using Spearman's ρ [115]; the average ρ between two workers is 0.57, and the average ρ between each worker with the other four annotators is 0.65.

The automatic quality ratings (denoted "PPDB score") are generated by using the human annotations to fit a supervised ridge regression model. The input to the model consists of 209 hand-crafted paraphrase features, including WordNet features [41], distributional similarity features, and cosine similarities of generated Multiview Latent Semantic Analysis embeddings [105]; the output is the human-annotated paraphrase score. The PPDB score is evaluated by comparing it with the human annotations, achieving Spearman's ρ of 0.71. In comparison, Pavlick et al. [93] report that using the word2vec embedding of the rarest word in each paraphrase obtains Spearman's ρ of 0.46.

Sent. 1	Albertson's and Kroger's Ralphs stores locked out their workers in response.
Sent. 2	Kroger's Ralphs chain and Albertsons immediately locked out their grocery workers in a show of solidarity.
Sent. 1	Electronic Data Systems Corp. Thursday said the Securities and Exchange Commission has asked the company for documents related to its large contract with the U.S. Navy.
Sent. 2	In a regulatory filing, EDS said the SEC had asked for information related to its troubled IT outsourcing contract with the US Navy.
Sent. 1	Nigeria and other African oil producers are increasingly important in U.S. plans to lessen dependence on Middle Eastern suppliers for its energy security.
Sent. 2	Nigeria and other African producers are increasingly important in the former Texas oilman's plans to lessen dependence on Middle Eastern suppliers for energy security.

Table 6.2: Examples of paraphrased sentences from the MRPC.

6.1.2 The Microsoft Research Paraphrase Corpus

In addition to the PPDB, we utilize paraphrases from the Microsoft Research Paraphrase Corpus (MRPC) [102, 32].[3] This dataset contains 5,801 paired sentences collected from news articles. Each pair of sentences was judged by at least two annotators, and 67% of the paired sentences are labeled as paraphrases. The remaining 33% of the paired sentences range from completely unrelated to almost semantically equivalent; for experiments in this chapter, we ignore these sentences, because of their wide range of semantic equivalence. Inter-annotator agreement averaged to 83%. Unlike the PPDB, we do not have word alignment information, so we only consider sentence-level experiments. Example paraphrases are shown in Table 6.2. With an average length of 19.3 ± 5.1 words, the MRPC sentences are much longer than the PPDB phrases.

6.2 Paraphrase Semantics in BERT

Using the PPDB and the MRPC, we examine BERT's ability to correctly understand paraphrase semantics. We consider both phrase-level and word-level embeddings. All experiments are run using the uncased base model of BERT, using a maximum sequence length of 128 and a batch size of 8.[4]

[3] Available online at https://www.microsoft.com/en-us/download/details.aspx?id=52398.

[4] Pre-trained model available online at https://github.com/google-research/bert.

6.2.1 Phrase-Level Embeddings for the Paraphrase Database

First, we consider phrase-level embeddings that capture aggregate information about all of the words in a given phrase. These embeddings give us information about whether BERT can distinguish between two paraphrases and two unrelated phrases. Specifically, we consider a set of paraphrases annotated by humans for paraphrase quality. After generating BERT phrase-level embeddings for each phrase, we measure Spearman's ρ between the cosine similarities of two phrase-level embeddings and the human annotations.

Our human annotations are from 25,736 phrase pairs in the PPDB.[5] We run each phrase through the pre-trained BERT model. For each pair of phrases, we average together the embeddings for each word to get a phrase embedding; then, we take the cosine similarity between the two phrase embeddings. Using Spearman's ρ, we compare all cosine similarities to the ground truth annotations. We create phrase embeddings and measure Spearman's ρ for all twelve BERT layers, as well as for all of the layers concatenated together into a single embedding.

We also compare BERT to a context-free output embedding method, word2vec (w2v) [85]. We train w2v on an English Wikipedia corpus of 5,269,686 sentences,[6] using dimension size 200, a window size of 5, and a minimum count of 5. We train five w2v models, using five different random seeds.[7] For each pair of phrases, we average together the embeddings for each word to get a phrase embedding and take the cosine similarity between the embeddings of the two phrases.[8] We report the average and standard deviation of Spearman's ρ over the five models.

BERT and w2v are compared in Table 6.3, where results are broken down by the average length of each paraphrase. For all layers, BERT improves on longer paraphrases. This is intuitive, because BERT is trained on longer sentences, and the longer the phrase, the more it will be able to leverage contextual information. We discuss this trend more in the next section. The last layer of BERT behaves markedly different than the other layers. While it continues to perform better on longer paraphrases, it does substantially worse

[5]This is marginally smaller than the entire human-annotated subset, which includes 26,455 pairs. We were unable to map all of the human-annotated pairs back to the full PPDB, in order to obtain the automatically generated PPDB score, which is relevant for experiments in this section. We have been in touch with the authors of the PPDB paper and have been unable to resolve this issue. This may also explain why our reported correlations between w2v and human annotations are lower than those reported by Pavlick et al. [93].

[6]This data was used in Tsvetkov et al. [122] and is available by contacting the authors of that paper. We additionally ran these experiments on the Europarl corpus [67] and got lower correlations. This is possibly because Wikipedia covers a wider range of topics than Europarl.

[7]2518, 2548, 2590, 29, 401

[8]For both BERT and w2v, we additionally tried using the embedding of only the rarest word (with frequency measured using the full PPDB), as reported in Pavlick et al. [93], but this gave us consistently lower correlations.

Method	Avg. Length 1-2.5	Avg. Length 2.5-4	Avg. Length 4-6
BERT Layer 1	0.18	0.35	0.47
BERT Layer 2	0.18	0.35	0.49
BERT Layer 3	0.18	0.37	0.48
BERT Layer 4	0.18	0.38	0.48
BERT Layer 5	0.18	0.39	0.48
BERT Layer 6	0.19	0.39	0.49
BERT Layer 7	0.20	0.40	0.49
BERT Layer 8	0.21	0.40	0.50
BERT Layer 9	0.21	0.40	0.50
BERT Layer 10	0.22	0.38	0.48
BERT Layer 11	0.22	0.36	0.46
BERT Layer 12	0.10	0.35	0.51
BERT Concatenated	0.20	0.40	0.51
w2v Average	0.35 ± 0.0	0.32 ± 0.0	0.41 ± 0.0
PPDB score	0.41	0.50	0.51
Num. paired phrases	17,517	5,349	2,870
Avg. human annotation	2.40 ± 1.0	2.94 ± 1.1	3.26 ± 1.1

Table 6.3: Spearman's ρ between human-annotated PPDB paraphrases and different embedding methods (BERT and w2v), broken down by average paraphrase length (the average number of words in each of the two phrases in the paraphrase). At the bottom of the table, we include the length distribution of the human-annotated paraphrases, as well as the average human annotation for each set of grouped lengths.

on short paraphrases, slightly worse on medium paraphrases, and slightly better on long paraphrases.

Similarly, w2v also improves on longer paraphrases, though it underperforms BERT for all but the shortest paraphrases. While w2v does not produce contextualized embeddings, it still has more information to incorporate into its phrase embeddings for longer paraphrases. We also see that the automatic PPDB score does better on longer paraphrases. This could be because it incorporates distributional information, which is richer when there are more words. Finally, the quality of the paraphrases goes up as the paraphrases get longer, as evidenced by the increasing human annotation score.

From Table 6.3, we conclude that the final layer of BERT outperforms w2v and performs comparably to the PPDB score on the longest paraphrases. This is not a completely fair comparison; the PPDB score has access to outside information that BERT does not, such as WordNet features and additional features derived from the translation process used to create the PPDB. These results give us confidence that BERT is successfully learning to

Phrase 1	Phrase 2	Cos. Sim.
laboratoires	laboratories	0.51
completly	totally	0.51
fervor	enthusiasm	0.52
79.0	seventy-nine	0.53
approximatly	around	0.54
-mom	-mother	0.91
1.350	1.35	0.92
characterises	characterizes	0.92
km	kilometres	0.92
garbage	trash	0.96

Table 6.4: Cosine similarity scores for the last layer of BERT for one-word paraphrases with the highest human annotation score.

distinguish between paraphrases and unrelated phrases.

6.2.2 One-Word Paraphrases in the PPDB

The case where both phrases in a paraphrase are a single word is an interesting subset of our experiments; in the previous experiment, this equates to asking BERT to identify the synonym level between two words, rather than whether or not two phrases are paraphrases. In the above section, we saw that BERT does not do as well on this task as it does at identifying longer paraphrases.

Among the subset of one-word paraphrases, there is a wide range of human annotations. The average annotation is 2.27 ± 0.99. We focus on one-word paraphrases with a human annotation of 5, the highest annotation score, indicating that these are the strongest synonyms among the group of one-word paraphrases. Among these high-quality synonyms, there is a wide range of cosine similarities for the last layer of BERT; the average similarity is 0.76 ± 0.12. Table 6.4 shows synonyms with both the highest and lowest BERT similarities. We observe that misspelled words (e.g., `completly`, `approximatly`) and foreign words (e.g., French `laboratoires`) have low cosine similarities. Numbers appear on both the low end (e.g., `79.0` and `seventy-nine`) and the high end (e.g., `1.350` and `1.35`) of the similarity spectrum. One difference between the similar and dissimilar cases of numbers is that in the similar case they both use digits, while in the dissimilar case one uses digits while the other uses words.

To further explore how BERT handles individual words, we use the WordSimilarity-353 dataset, which contains 353 pairs of words with human-assigned similarity judgments

[42].[9] We remove 24 word pairs where BERT tokenizes a single word into more than one token; in these cases, we are unable to do a single cosine similarity comparison. For the remaining 329 word pairs, we run each word through BERT and take the cosine similarity between the word embeddings for each word in the pair. We use Spearman's correlation to compare BERT cosine similarities with the human annotations. In addition to using BERT embeddings, we use word embeddings produced with the five word2vec embedding spaces described in the previous section. In the 329 word pairs used, 80 words are unknown in the word2vec embedding spaces; we use an all-zero embedding for these words.

BERT word embeddings achieve a Spearman's correlation of 0.32, while word2vec embeddings give a substantially higher Spearman's of 0.56 ± 0.0. Part of the reason for BERT's poor performance on this task could be that BERT needs contextual information about a word in order to produce an informed embedding, while word2vec does not require any contextual information. BERT was specifically designed for sentences, and running individual words through BERT is not how the algorithm is intended to be used. However, it is interesting that before any fine-tuning, BERT does a poor job of recognizing synonyms, as observed with both one-word paraphrases in the PPDB and the WordSimilarity-353 dataset.

6.2.3 Phrase-Level Embeddings for the Microsoft Research Paraphrase Corpus

To confirm that BERT can recognize paraphrases, we verify our experiments using paraphrases from the MRPC. We follow the same procedure as with the PPDB, running each sentence through the pre-trained BERT model and averaging together the embeddings from the last layer of BERT for each word to get sentence embeddings. We do not have human annotations for this dataset, so we instead compare cosine similarities of the sentence embeddings for two sets of sentences: sentences marked as paraphrases in the MRPC, and randomly paired sentences from the MRPC, shown in Figure 6.2. We see that the similarity scores are substantially higher for paraphrases, confirming our previous experiments and showing that BERT is able to recognize paraphrases.

6.2.4 Word-Level Embeddings for the Paraphrase Database

Phrase-level embeddings allow us to look at whether BERT is able to distinguish between paraphrases and unrelated phrases. Word-level embeddings give us information about whether BERT is capturing the semantics of individual words. Here, we consider four

[9] Available online at http://www.gabrilovich.com/resources/data/wordsim353/wordsim353.html.

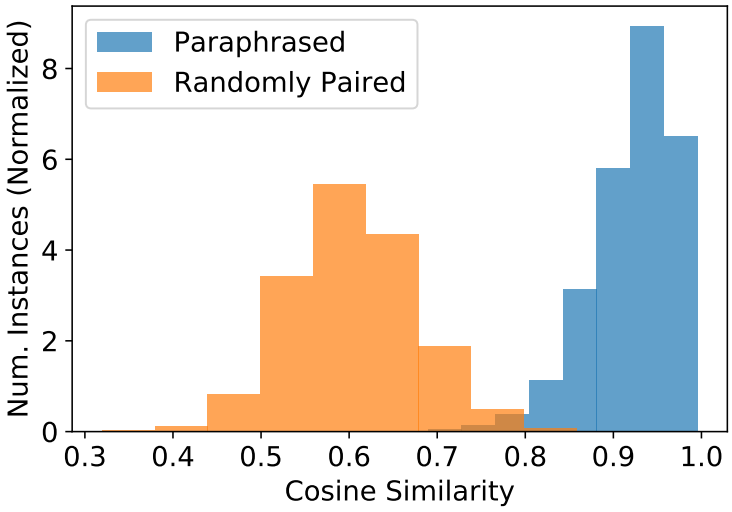


Figure 6.2: Distribution of cosine similarities using the last layer of BERT for paraphrases and randomly paired sentences in the MRPC. Both histograms are normalized to form probability densities.

different sets of words. First, we make a distinction between words that are *aligned* and those that are *unaligned*. Second, we make a distinction between words that are the *same* in both phrases and those that are *different*. Intuitively, aligned different words are likely synonyms, and unaligned same words are polysemous words being used in different senses. Unaligned different words are not completely unrelated words; they are a part of the same paraphrase, and they will be more related than words from two different paraphrases.

Because the MRPC does not provide word alignment information, we use the PPDB for all word-level experiments. We also restrict our focus to the most relevant section of the PPDB, long paraphrases (where one of the phrases has at least six tokens) with a relatively good PPDB score[10] and no syntactic placeholders. From this set, we randomly sample 4,000 paraphrases. Our sample yields 22,751 aligned same words, 25,973 aligned different words, 2,782 unaligned same words, and 163,474 unaligned different words. We randomly sample 2,500 words from each category.

To generate word-level embeddings, we run each phrase through the pre-trained BERT model and for each pair of words, we take the cosine similarity between the embeddings of the two words. We do this for all twelve layers of BERT, as well as for all layers concatenated together.

First, we consider the set of same aligned words and compare Spearman's ρ between BERT word-level cosine similarities and the PPDB scores for all layers of BERT, shown in

[10]Using the automatic PPDB score, the PPDB is divided into six sizes, from S up to XXXL. We use size S, those paraphrases with the highest PPDB scores for the highest precision.

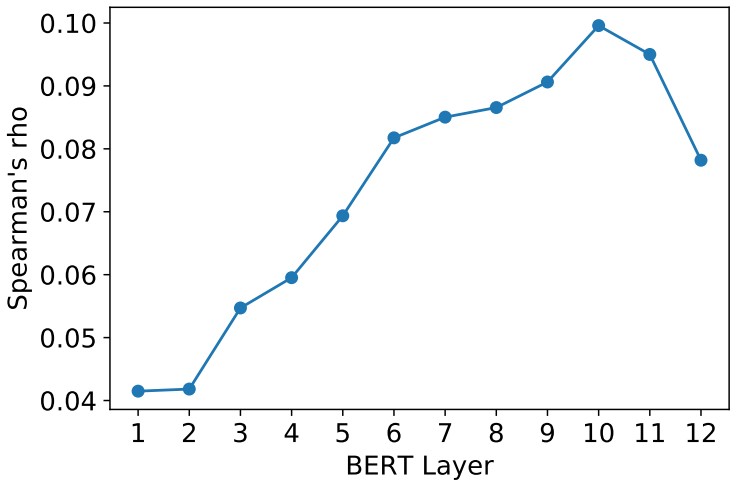


Figure 6.3: Spearman's ρ between BERT cosine similarities and PPDB scores for all aligned same words, broken down by BERT layer.

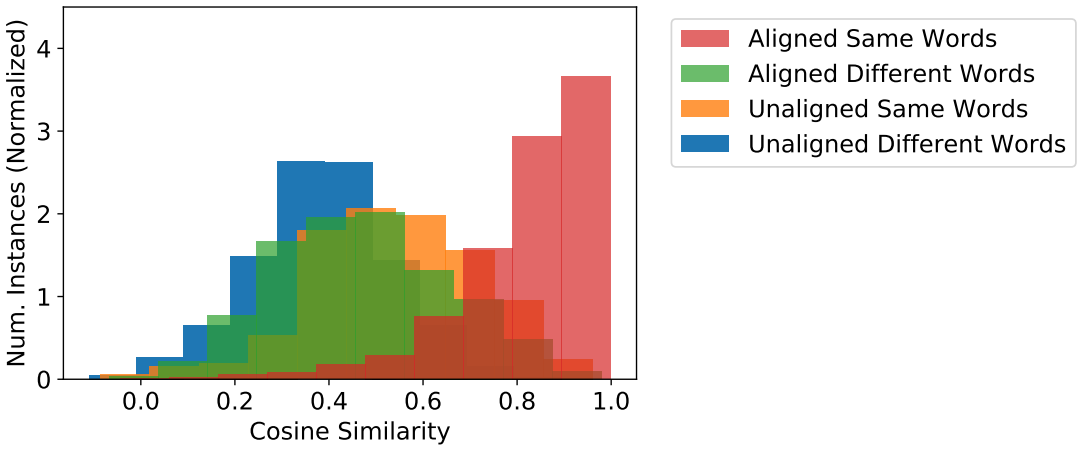


Figure 6.4: Distributions of cosine similarities using the last layer of BERT for different groups of words in the PPDB. All histograms are normalized to form probability densities (best seen in color).

Figure 6.3. We see a similar pattern to Table 6.3; ρ gradually increases in the later layers, before dropping slightly in the final two layers.

Considering all four sets of words, Figure 6.4 compares cosine similarity distributions for the last layer of BERT. The highest category consists of aligned same words. Because these words are identical and play the same role in the paraphrase, we would expect their BERT embeddings to have a high similarity. The remaining three groups of words have very similar distributions. We would expect to see a difference between aligned different words and unaligned words, and not seeing this difference indicates that BERT does not always understand when two different words have the same meaning in a paraphrase. This

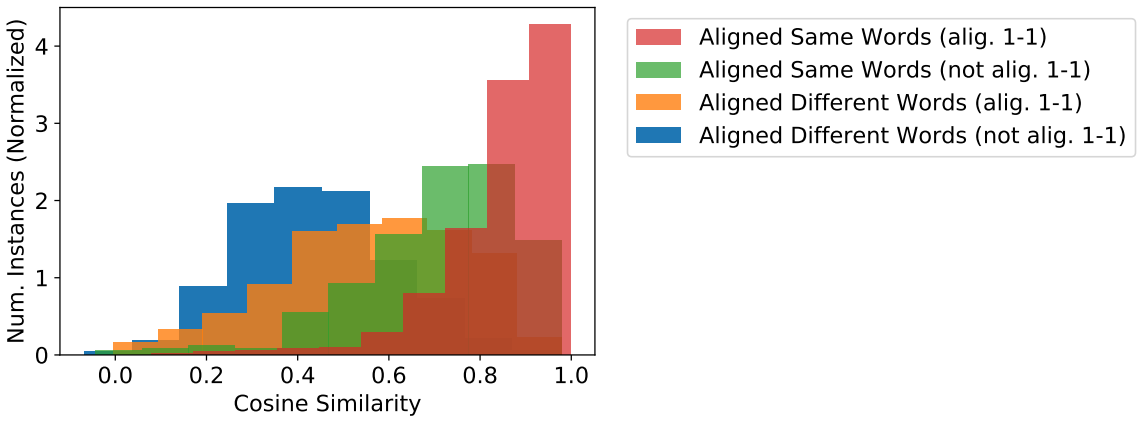


Figure 6.5: Distributions of cosine similarities using the last layer of BERT for same and different aligned words, broken down by whether the word is in a one-to-one alignment with another word. All histograms are normalized to form probability densities (best seen in color).

is a failure of BERT to completely control for semantics in paraphrases. We do see that unaligned same words have a slightly higher similarity than unaligned different words, which is intuitive because different words often have different connotations or senses.

Aligned words can be further divided into words that have a one-to-one alignment with another word, and words that are a part of a one-to-many or many-to-one alignment. For instance, in the paraphrases `members of the panel` and `members of the panel of experts`, the word `panel` in the first phrase is aligned to the set of words `panel of experts` in the second phrase; this is a one-to-many word alignment. In contrast, the word `contribute` in the paraphrases `contribute to improving the` and `contribute to the improvement of the` is a one-to-one word alignment.

Same aligned words are more likely to be in a one-to-one alignment than different aligned words; 80% of same aligned words are one-to-one, while only 26% of different aligned words are one-to-one. We compare the cosine distributions of these sets of words in Figure 6.5. For both same and different words, one-to-one aligned words have slightly higher similarities. As in Figure 6.4, we see that same aligned words (both one-to-one and not one-to-one) have higher similarity distributions than different aligned words. Looking at alignment information adds an important nuance to the comparison of same and different words, but it does not reverse the trends of Figure 6.4.

To further understand BERT's word-level embeddings, we examine two cases of particular interest: words with low BERT similarity but high PPDB score, and words with high BERT similarity but low PPDB score; examples are shown in Table 6.5. In several cases where there is a mismatch between the PPDB score and the cosine similarity, BERT

Token	Phrase 1	Phrase 2	PPDB Score (Norm.)	Cos. Sim.
members	members of the security council	the members of the council ,	0.81	0.46
of	and protect the human rights of	the protection of human rights	0.80	0.40
.	- i am sorry .	well , i 'm sorry .	0.80	0.44
regard	, including with regard	, in particular with regard to	0.79	0.43
the	contained in the annex to the	annexed to the	0.79	0.62
board	board of referees	members of the board of referees	0.79	0.54
,	- yes , i am .	all right , yeah .	0.78	0.57
i	yeah , i 'm sure	- yeah , i got it	0.78	0.60
consideration	consideration of the matter at	its consideration of the item on	0.77	0.59
action	plans of action for the implementation	action plan for the implementation	0.77	0.28
the	adjournment of the debate on the	to adjourn the debate on the	0.70	0.96
-	- what 's the matter ?	- what 's happened ?	0.70	0.97
of	note of the information provided by	note of the information contained in	0.70	0.96
,	president , i am very pleased	president , i am pleased	0.70	0.98
-rrb-	-rrb- to continue to make available	-rrb- to continue to provide	0.69	0.97
to	permit me to introduce myself .	allow me to introduce myself .	0.69	0.98
?	is there someone there ?	is there anybody out there ?	0.69	0.96
for	: draft programme budget for the	: proposed programme budget for the	0.69	0.97
contained	6.1 before considering any claim contained	5.1 before considering any claim contained	0.69	1.00
before	6.1 before considering any claims contained	5.1 before considering any claim contained	0.69	0.99

Table 6.5: Examples of words and paraphrases with either high PPDB scores and low cosine similarities, or low PPDB scores and high cosine similarities. All examples are randomly sampled from the top or bottom 10% of each relevant category. PPDB scores are normalized to be between 0 and 1. Cosine similarities are using the last layer of BERT. The aligned tokens are underlined.

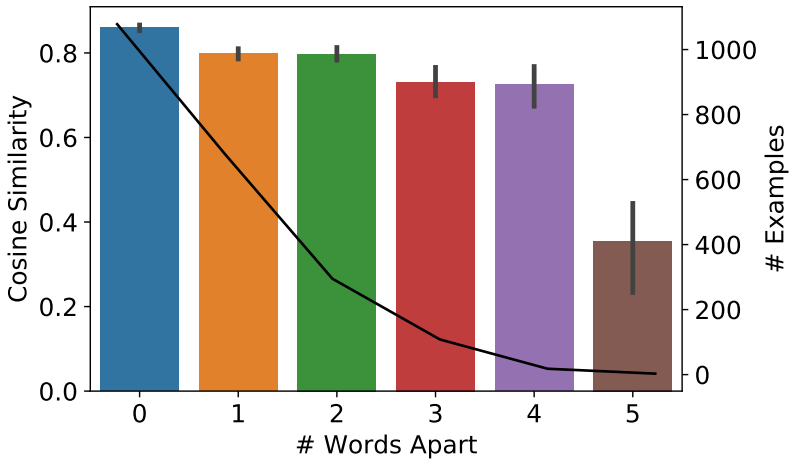
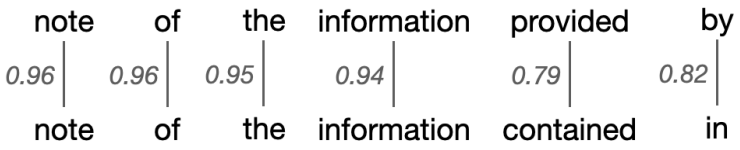


Figure 6.6: Cosine similarity using the last layer of BERT for aligned same words broken down by the number of words apart that the words are in the two phrases (shown in bar plot and left y-axis). Error bars indicate confidence intervals. The line graph and the right y-axis show how many examples we have for each category.

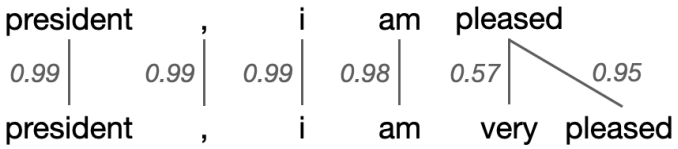
appears to be detecting appropriate nuances. For instance, the word `for` in the phrases : `draft programme budget for the` and : `proposed programme budget for the` is given a very high similarity score (0.97), even though the overall paraphrase has a lower PPDB score (0.69). In this example, the difference between the two phrases is entirely contained in the words `draft` and `proposed`, so it is intuitive that the other tokens of the paraphrase would have high similarities. Both stopwords and punctuation appear frequently in this table, and we discuss these in more detail later in the chapter.

Qualitatively looking at these examples, we notice that when a token appears in a different position in the paraphrase, the similarity tends to be lower (e.g., the word `action` in the phrases `plans of action for the implementation` and `action plan for the implementation` has a similarity of 0.28). To explore this at a larger scale, we consider 2,181 aligned same words where we have information about how many tokens apart the two words are. Figure 6.6 shows the cosine similarity of the last layer of BERT broken down by how far apart the two words are. Spearman's $\rho = -0.29$ (p-value $< 10e - 42$), indicating that the farther away two words are in the paraphrase, the lower cosine similarity they will have. This supports the conclusion drawn in Mickus et al. [82] that the position of the sentence where the word appears in BERT affects the embeddings of the word, most likely because of the next sentence prediction objective used in BERT.

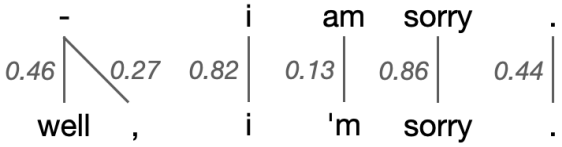
To better understand how cosine similarities of words in the same paraphrase interact, Figure 6.7 shows three detailed examples of paraphrases with word alignment and cosine similarity information. In the first example (Figure 6.7a), the two phrases are identical



(a) PPDB score: 4.3.



(b) PPDB score: 4.3.



(c) PPDB score: 5.2.

Figure 6.7: Example paraphrases from the PPDB with word alignment and word cosine similarities using the last layer of BERT shown.

except for one word (`provided` in Phrase 1 v. `contained` in Phrase 2). Intuitively, all of the words have high similarities except for the word that has been changed, which has a slightly lower similarity of 0.79. This is still fairly high, because `provided` and `contained` are highly related words. In the second example (Figure 6.7b), the second phrase adds a word (`very`) that is not present in the first phrase. Again, we see that all the word pairs except for the pair containing this added word have high similarity scores. In this example, `pleased` in Phrase 1 is aligned with `very pleased` in Phrase 2, and the similarity between `pleased` and `very` is intuitively low (0.57), since these are not related words and are different parts of speech. We also saw in Figure 6.5 that words that are not in a one-to-one alignment (such as `very` and `pleased`) tend to have lower similarities.

The third example (Figure 6.7c) is the most complex, with one word being swapped (`am` in Phrase 1 and `'m` in Phrase 2), and the dash in Phrase 1 being aligned to both the word `well` and a comma in Phrase 2. Surprisingly, BERT fails to recognize that `'m` is a contraction for `am` and gives this word pair a very low cosine similarity (0.13). This could be because of a difference in tokenization between the PPDB and the data that BERT is trained on. We also see that even though the period occurs at the end of both paraphrases following the word `sorry`, it has a fairly low cosine similarity (0.44). Finally, we see that the dash in Phrase 1 has a higher cosine similarity score with the word `well` (0.46) than

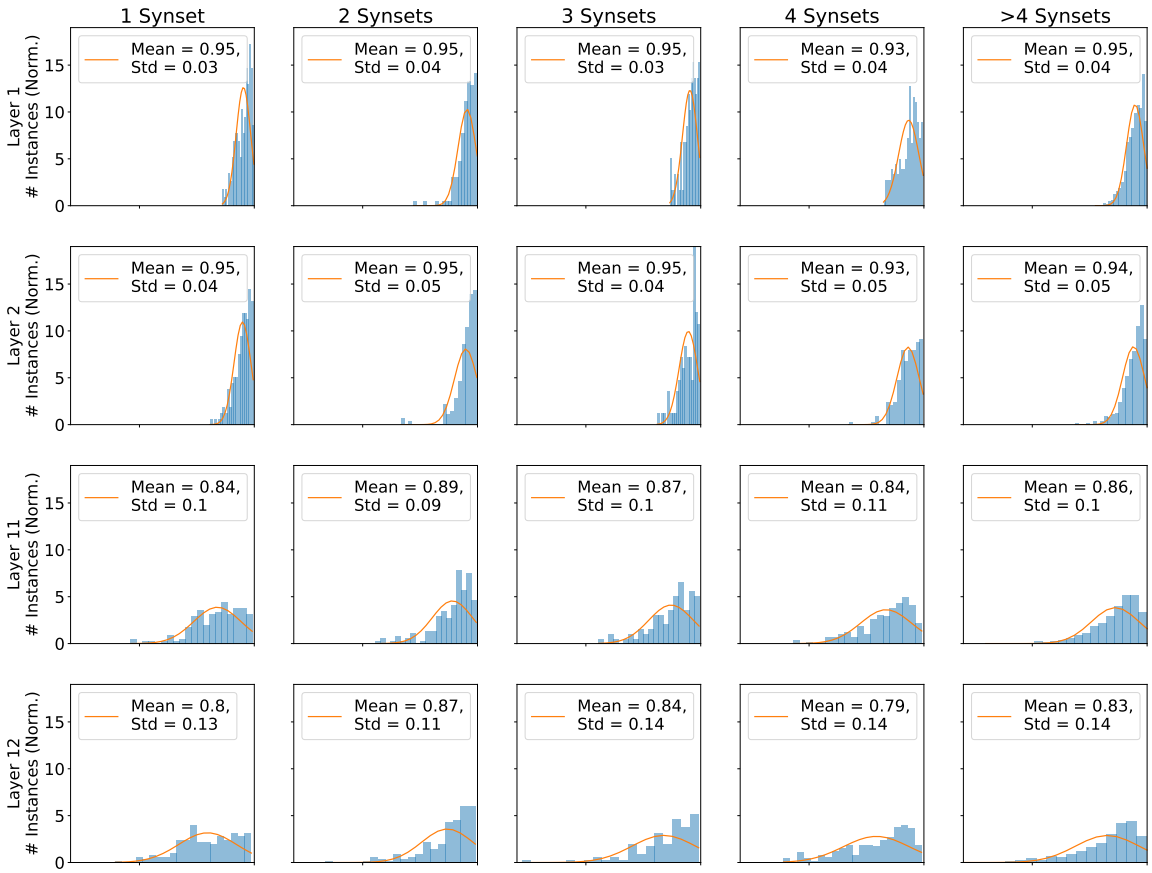


Figure 6.8: Distributions of cosine similarities for aligned same words across the first two and the last two layers of BERT for words with different polysemy. All histograms are normalized to form probability densities, and each is shown with a fitted Gaussian distribution (mean and standard deviation in legends).

with another punctuation mark, the comma (0.27). This is somewhat unintuitive, and we explore these punctuation marks more later in the chapter.

6.3 Using Paraphrases to Understand Properties of BERT

In the previous section, we showed that BERT is successfully able to identify paraphrases, and it controls well for the semantics of aligned words in paraphrases. We now use this information to examine other properties of BERT.

6.3.1 Polysemy

First, we look closer at how the polysemy of a word affects its representation in BERT. Previous work has shown that BERT embeddings form clusters based on word senses [127].

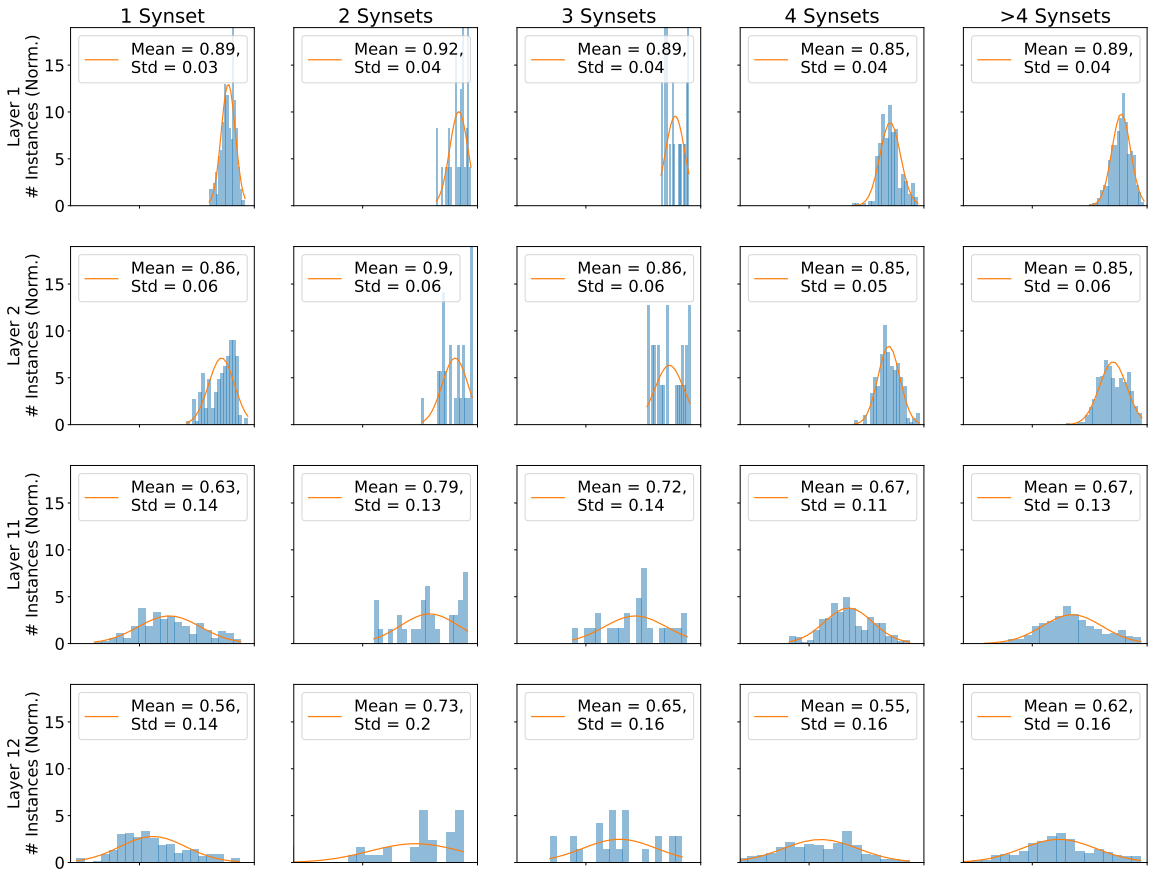


Figure 6.9: Distributions of cosine similarities for unaligned same words across the first two and the last two layers of BERT for words with different polysemy. All histograms are normalized to form probability densities, and each is shown with a fitted Gaussian distribution (mean and standard deviation in legends).

However, in the context of aligned words in a paraphrase, we would expect even a highly polysemous word to have similar embeddings in the two phrases.

To measure polysemy, we consider the number of WordNet synsets of a word, and we focus on words that are the same in both phrases, both aligned and unaligned. In order to have enough data to make a good comparison, we use the 4,000 sampled paraphrases from the previous section, as well as an additional random sample of long paraphrases with at least one unaligned same word. We then downsample the aligned same words to get 1,597 instances of both unaligned and aligned same words that are present in WordNet, with up to 52 synsets.[11]

In Figures 6.8 and 6.9, we show the cosine similarity distributions for both aligned and unaligned words with different levels of polysemy across the first two and the last two layers of BERT. There is not a substantial difference between words with different synsets,

[11] We look up WordNet synsets using the Python NLTK library [14].

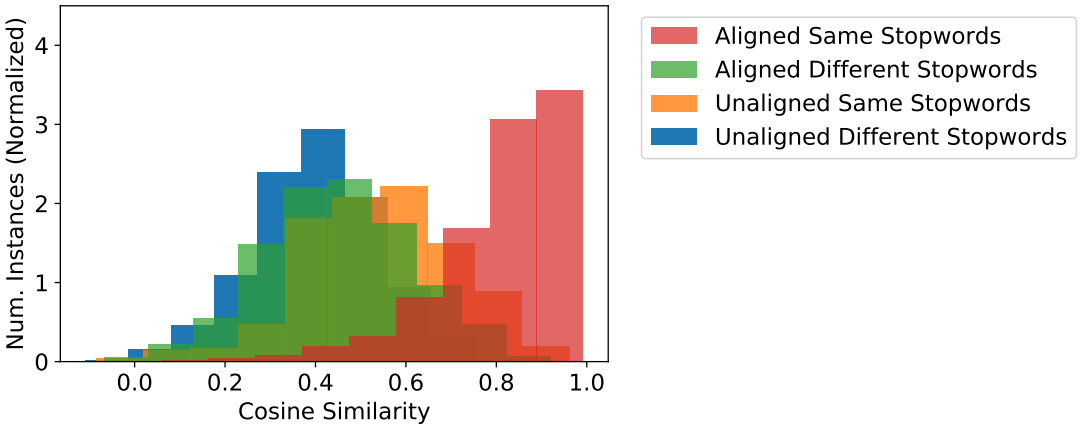


Figure 6.10: Distributions of cosine similarities using the last layer of BERT for different groups of stopwords in the PPDB. All histograms are normalized to form probability densities (best seen in color).

which supports our conclusion that BERT successfully controls for the semantics of aligned same words in paraphrases. We do see a difference between aligned (Figure 6.8) and unaligned words (Figure 6.9). Aligned words peak at a high cosine similarity, while unaligned words roughly follow a normal distribution centered around 0.5. Note that for unaligned words with two or three synsets, there is not enough data to draw conclusions about the cosine similarity distributions. Overall, these plots show that even highly polysemous aligned same words have very similar embeddings in the context of a paraphrase.

6.3.2 Stopwords

Earlier, we saw that Table 6.5 had several examples of stopwords in it. Part of the reason stopwords appear frequently in that table is that stopwords are common in our dataset of paraphrases.[12] Of the same aligned words, 54% are stopwords; 62% of the different aligned words are stopwords. Stopwords make up an even greater percentage of unaligned words; 78% of same unaligned words and 77% of different unaligned words are stopwords.

To further explore stopwords, we look at the cosine similarity distribution for different sets of stopwords for the last layer of BERT in Figure 6.10. Comparing this figure to Figure 6.4, we do not see a substantially different distribution of cosine similarity scores for stopwords than for all words. Ethayarajh [39] found that stopwords have some of the most context-specific representations, but we do not see that trend in the context of paraphrases.

[12]We use the set of English stopwords from NLTK.

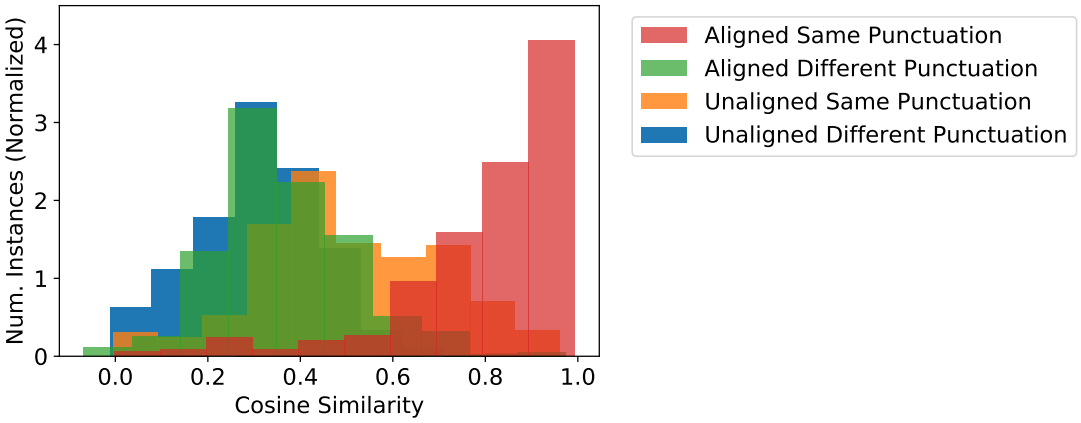


Figure 6.11: Distributions of cosine similarities using the last layer of BERT for different groups of punctuation in the PPDB. All histograms are normalized to form probability densities (best seen in color).

6.3.3 Punctuation

In addition to stopwords, we also consider punctuation. This appeared frequently in the examples in Table 6.5; we also saw unintuitive behavior around punctuation marks in Figure 6.7. Looking at punctuation has also proved to be useful for certain embedding-related tasks. For instance, embeddings are used to generate punctuation for text that is lacking punctuation, such as recorded transcripts [131]. Punctuation also plays an important role in distinguishing between different types of text, such as texts by different authors [114] or texts produced by different Twitter communities [120].

To understand how BERT handles punctuation differently than other words, we look at the cosine similarity distribution for different sets of punctuation tokens for the last layer of BERT in Figure 6.11.[13] Comparing this figure to Figure 6.4, we see that punctuation in general has a broader distribution of cosine similarities, indicating that punctuation embeddings are highly dependent on the surrounding contexts.

In Figure 6.12, we break these trends down by individual punctuation marks, focusing only on aligned same words. We look at the four most common punctuation marks, the comma, period, question mark, and dash. Of these four, the comma and period show the widest distributions. Even when they play the same role in the paraphrase, they can be given very different embeddings, indicating how highly contextualized these punctuation marks are. The question mark and dash are less contextualized; this is most likely because these punctuation marks are used in more prescribed circumstances. In this dataset, they are almost always used at the beginning or the end of the paraphrase. In all but one example,

[13]Each category of tokens contains 335 punctuation marks.

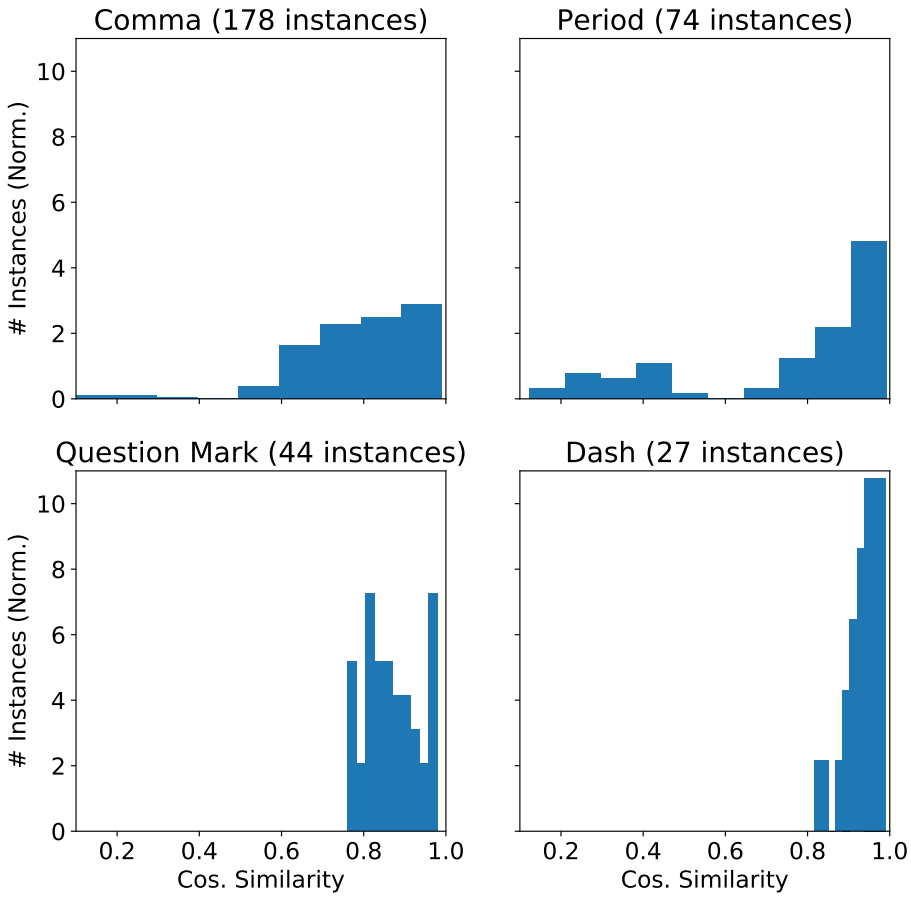


Figure 6.12: Distribution of cosine similarities using the last layer of BERT for aligned same words, broken down by punctuation mark. The four most common punctuation marks are shown here. All histograms are normalized to form probability densities.

the question mark is the last token of both phrases; the dash is the first token of both phrases in all but two examples. Table 6.6 shows examples in context of each of these punctuation marks.

6.3.4 Contextualization in BERT Layers

After looking at special groups of words (polysemous words, stopwords, punctuation), we consider how context-specific the words in a paraphrase are in general. Previously, Ethayarajh [39] showed that BERT word embeddings are more context-specific in higher layers. They measure this using the self-similarity of words, defined as the average cosine similarity between a word's contextualized representations across its unique contexts, and show that self-similarity consistently decreases with later layers of BERT, indicating that the contextualization of words is increasing.

Token	Phrase 1	Phrase 2	Cos. Sim. (Last Layer)
,	, especially small	including , in particular , the	-0.00
,	note , however	however , it should be noted	0.13
,	oh , i am so sorry	i 'm sorry , doctor	0.69
,	- no , it wo n't	- no , he does n't	0.94
.	yes , i am .	well , that 's true .	0.20
.	- i am sorry .	well , i 'm sorry .	0.44
.	- yes , i am .	- i 'm ready .	0.74
.	, that 's all right .	, this is good .	0.93
?	what 's wrong ?	- what is your problem ?	0.77
?	where have you come from ?	where are you from ?	0.94
-	news - politics - world -	news - international politics -	0.77
-	- yes , thank you .	- yeah , thank you .	0.99

Table 6.6: Examples of aligned punctuation marks with varying cosine similarities. The aligned tokens are underlined.

We compare this observation to the paraphrase setting that we have been exploring in this chapter. Because there are only two phrases in a paraphrase, we cannot implement the full self-similarity metric. Instead, we measure the cosine similarity between words in these two phrases, shown in Figure 6.13, which captures the same concept as self-similarity. We see two trends reflected in this graph. The first, decreasing cosine similarity, is seen with same words, whether aligned or unaligned, and is similar to what Ethayarajh [39] report with decreasing self-similarity scores. This trend is stronger with unaligned words than with aligned words, indicating that more contextualization is happening with unaligned words and BERT is somewhat controlling for the semantics in paraphrases. The second trend that we see is the opposite, increasing cosine similarity, and we see this trend with different words, both aligned and unaligned. This indicates decreasing contextualization. Surprisingly, we see that different aligned words have lower cosine similarities than unaligned same words. This reflects the patterns in Figure 6.4, where we noted that BERT does not completely control for the semantics of paraphrases.

6.3.5 Intra-Sentence Similarity

Another way that Ethayarajh [39] measures contextualization is by looking at intra-sentence similarity, defined as the average cosine similarity between each word's embedding and the overall sentence embedding (the mean of all word embeddings). In order to

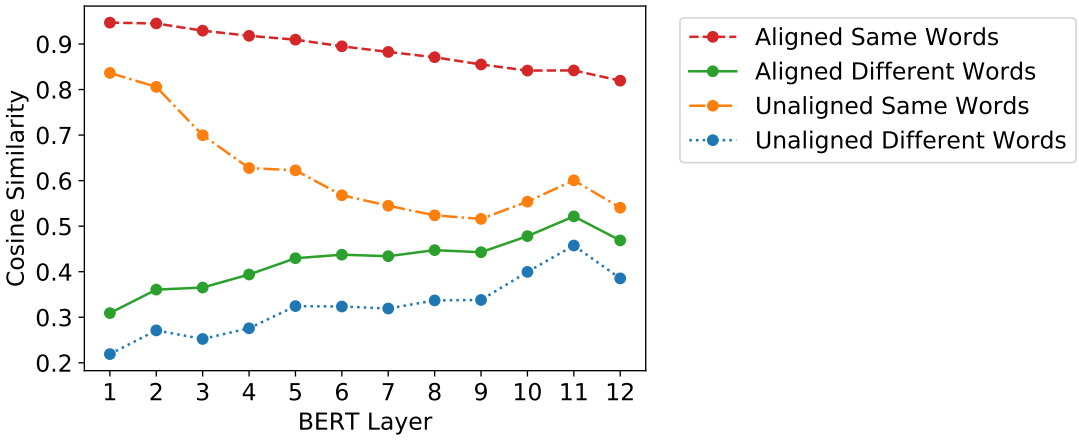


Figure 6.13: Cosine similarity for different groups of words in the PPDB across all layers of BERT.

take into account the geometry of the final word embedding space, this measure is adjusted for anisotropy (lack of directional uniformity in the embedding space). Specifically, when measuring intra-sentence similarity, for each layer of BERT, they subtract the average cosine similarity between the representations of 5,000 uniformly randomly sampled words from different paraphrases.

We measure intra-sentence similarity for our sample of 4,000 paraphrases in the same way and compare our results, shown in Figure 6.14. Ethayarajh [39] found that in general, words in the same sentence are more dissimilar to one another in upper layers. In contrast, once we adjust for anisotropy, we see that words in the same sentence are becoming more similar to each other in later layers of BERT. We attribute this to the fact that two phrases which are paraphrases will have overall similar phrase embeddings. However, like Ethayarajh [39], we find the average similarity of words in a sentence is consistently higher than the anisotropy baseline (e.g., two words are not necessarily close to each other because they are in the same sentence). This indicates a nuanced contextualization.

6.4 Lessons Learned

From our analyses using paraphrases, we draw out several lessons about contextualized word embeddings.

1. BERT does a reasonable, but not perfect job recognizing paraphrases. At the phrase level, BERT does a good job separating paraphrases from unrelated phrases. On the PPDB, it outperforms word2vec and does as well as the feature-based PPDB score (Table 6.3). On the MRPC, there is a large gap in cosine similarities between paraphrases and non-

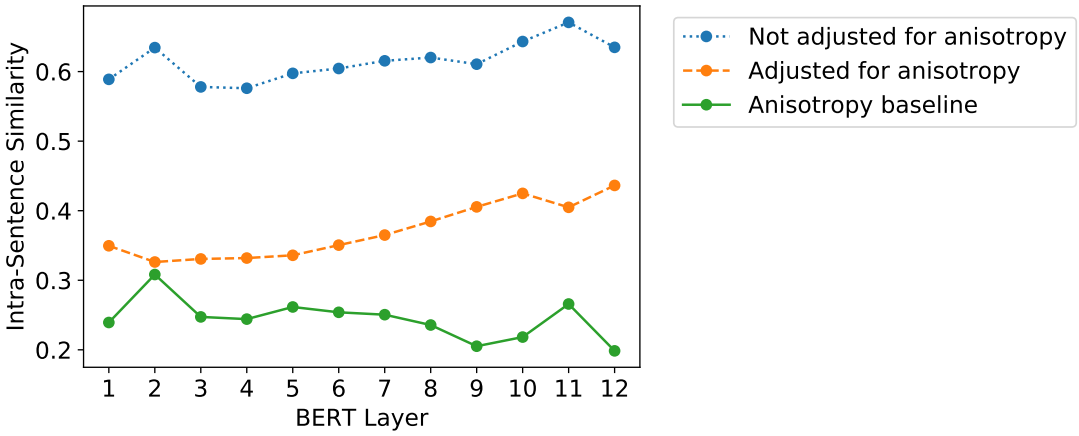


Figure 6.14: Intra-sentence similarity, both adjusted and not adjusted for anisotrophy, measured for all BERT layers. We also include the anisotropy baseline used in the adjustment.

paraphrases (Figure 6.2).

Looking at word-level, rather than phrase-level cosine similarities, BERT gives high similarities for aligned same words (Figure 6.4). However, BERT fails to capture the proper semantics of words when different words are aligned. Because these words are aligned in the paraphrase, they have similar semantics, and we would expect BERT to give them similar embeddings. Yet, there is no substantial difference in similarity between aligned different words and unaligned words. We even see that aligned different words have lower cosine similarities than unaligned same words (Figure 6.13).

2. BERT correctly handles polysemy in paraphrases. If two highly polysemous words are the same and aligned, BERT gives them very similar embeddings. This contrasts with aligned different words, which have less similar embeddings. This supports the previous takeaway, that BERT correctly recognizes paraphrases, even when there are polysemous words involved.

3. Unexpectedly, BERT gives words that are farther apart lower cosine similarity scores. Confirming observations by Mickus et al. [82], we see that aligned same words that are farther apart from each other in the paraphrase have lower cosine similarities than words that are in the same position in the paraphrase. This is not an intuitive result, as the distance of words in the paraphrase does not affect the semantics of the paraphrase.

4. BERT correctly gives less contextualized representations to paraphrased words than non-paraphrased words. The exception to this is punctuation, which has highly contextual representations. Looking at the cosine similarity of words across layers, we see that for aligned same words in paraphrases, contextualization decreases as the BERT layer increases. This is intuitive, because BERT is controlling for semantics in paraphrases.

We do however see one group of words that tends to be more contextualized than other words: punctuation. We can attribute this to the fact that the same punctuation mark can be used in many different contexts.

6.5 Implications

While the analyses in this chapter focus on a single algorithm, BERT, the ideas and techniques used here could also be applied to other contextualized output embedding algorithms. For the many different variants of BERT (such as compressed versions of BERT, e.g., [110, 136]), we would expect to see similar results. Understanding the behavior of contextualized output embedding algorithms in different contexts and for different groups of words has various practical implications for how we use these algorithms.

One part of training that is not addressed in these analyses is fine-tuning. When BERT is used for a specific downstream task, the base model is often fine-tuned for that particular task (e.g., [118]). Fine-tuning creates a specific model for each task, so fine-tuned models for different tasks can vary substantially. To fully understand how fine-tuned models handle paraphrases, we would need to consider a large number of diverse models. However, pre-training and fine-tuning serve different purposes; pre-training aims to capture general-purpose properties of language that apply across a wide variety of tasks, while fine-tuning tweaks the architecture to perform a very specific downstream task. In this chapter, we are investigating the similarity of paraphrases, and the properties that we look at are primarily general properties of language (synonymy, polysemy, stopwords, etc.). Because of this, we expect that many of the trends we see with the pre-trained BERT model would continue to be relevant in fine-tuned models.

One observation we have made is that BERT's output embeddings are affected by irrelevant factors, such as how far apart two words are in a pair of phrases. All other things being equal, the position of a word in a sentence should not affect the resulting embedding. We have also seen that when words do not have any context (one-word phrases), BERT does not produce good embeddings. These observations are important for tasks that compare individual words or parts of sentences, such as sentiment analysis between specific entities. For a task like that, it is important to make sure that the entities or words being compared are appropriately contextualized in sentences, and that if pairs of sentences are being considered, the location of the entities in the sentence needs to be consistent. Making small adjustments like this could lead to substantially better performance on these tasks.

Finally, we have seen that paraphrases are a valuable source of data for investigating patterns in BERT's behavior. BERT is able to reasonably identify paraphrases versus unre-

lated phrases. This indicates that there is potential for BERT to be used in the generation of paraphrases, which have proved to be a valuable source of data for many NLP tasks (e.g., [16, 61]). Using BERT as a source of paraphrases could prove to be an relatively easy way to generate large amounts of usable data.

6.6 Conclusion

Throughout this chapter, we have used paraphrases as a way to understand the properties and behaviors of BERT. We have shown that BERT does a good job controlling for the semantics of paraphrases, and we have drawn out lessons for specific groups of words (polysemous words, stopwords, and punctuation).

We have also compared our observations with previous research on BERT. Like Mickus et al. [82], we observe that the position of the sentence where the word appears in BERT affects the embeddings of the word (Section 6.2.4). When looking at polysemy, we compare to Wiedemann et al. [127], who find that BERT embeddings form clusters based on word senses. Unlike them, we observe that in the context of a paraphrase, even highly polysemous aligned same words have similar embeddings (Section 6.3.1).

Finally, we compare to several observations made by Ethayarajh [39]. They found that stopwords have some of the most context-specific representations, but in the context of a paraphrase, we do not see a substantially different distribution of similarity scores for stopwords than for all words (Section 6.3.2). Ethayarajh [39] also show that BERT word embeddings are more context-specific in higher layers. We notice a similar trend for same words in a paraphrase, whether aligned or unaligned, but we see decreasing contextualization for different words in a paraphrase (Section 6.3.4). Similarly, Ethayarajh [39] found that words in the same sentence are more dissimilar to one another in upper layers. In contrast, we see that words in the same paraphrase are becoming more similar to each other in later layers of BERT (Section 6.3.5).

This chapter has looked at how and when contextualized output embeddings change, using paraphrases as a source of data. This builds on previous work in Chapters 3 and 4 looking at the concept of stability in context-free output embedding spaces. We have used the unique properties of paraphrases to investigate assumptions and properties of BERT.

CHAPTER 7

Conclusion

In this thesis, we explore properties and limitations of word embeddings. We show how this affects usage of word embeddings both in downstream applications and corpus-centered applications, where embeddings are used to study language directly. We examine a key assumption that researchers often make about embeddings, that they learn meaningful relations, and that because these relations are present in the text, they are always present in embeddings.

We conclude this thesis by revisiting the research questions posed in the introduction.

- **Research Question #1. Are word embeddings stable across variations in data, algorithmic parameter choices, words, and linguistic typologies?**

  We introduce a definition of stability, and show that by this definition, common English word embedding spaces are surprisingly unstable. We show how data properties, word properties, and algorithm properties are related to this instability. Particularly, we observe for English embeddings that curriculum learning is important, part-of-speech is one of the biggest factors in stability, stability within domains is greater than stability across domains, GloVe is the most stable embedding algorithm, and frequency is not a major factor in stability.

  We extend this work to look at approximately 100 languages other than English, and we draw out several aspects of the relationship between linguistic properties and stability, including that languages with more affixing tend to be less stable, and languages with no gender systems tend to be more stable.

  Finally, we extend our analysis to contextualized output embedding algorithms. We use paraphrases as a way to control for the semantics of individual words, and we show that BERT is able to reasonably, though not perfectly, control for semantics. We also highlight insights about how BERT handles polysemous words, stopwords, and punctuation.

- **Research Question #2. How does our knowledge of stability and other word embedding properties affect tasks where word embeddings are commonly used, both downstream applications and corpus-centered applications?**

  We show that word stability affects the English tasks of word similarity and part-of-speech tagging. In particular, we note that an LSTM used for part-of-speech tagging shifts unstable word embeddings more than stable embeddings. When expanding this to multiple languages, we pinpoint several linguistic properties that are related to instability in word embeddings in various scenarios.

  In addition to stability, we consider batching and curriculum learning, and we systematically study different batching and curriculum learning decisions on three tasks, text classification, sentence and phrase similarity, and part-of-speech tagging. We find that certain methodological decisions increase performance substantially.

- **Research Question #3. How does our knowledge of stability and other word embedding properties affect our usage of embeddings?**

  We give practical suggestions on how to mitigate instability in English word embeddings: use in-domain embeddings whenever possible, choose GloVe over word2vec, and consider learning a good curriculum for word2vec. For multilingual word embeddings, we bring out certain linguistic properties that affect word embeddings, and this can be used as a starting point for future research in building more robust multilingual embedding methods.

  We also suggest ways to improve batching and curriculum learning for the three tasks that we specifically consider these properties for, text classification, sentence and phrase similarity, and part-of-speech tagging.

Word embeddings have proven to be a powerful and versatile tool for natural language processing, and the work in this thesis will help researchers use embeddings more robustly and effectively.

7.1 Future Work

These analyses and experiments primarily impact two groups of researchers. First, this research affects scientists conducting corpus-centered research, those who use embeddings to study the language of a document and to draw conclusions from a corpus. Second, this work impacts practitioners who use embeddings as part of a pipeline to accomplish downstream tasks.

For corpus-centered embedding research, we have shown that analyzing raw embeddings can be potentially misleading, as embeddings are unstable across variations in data, algorithms, and parameters. We have made suggestions about how to mitigate this, such as looking at multiple embedding spaces and reporting the variation, but more research is needed into techniques for analyzing text using embeddings. Specifically, the community should pursue methods for comparing multiple sources of text in a consistent manner, in a way that would allow researchers to pull out important differences between the texts, rather than unimportant parameter or algorithm variations. These methods would benefit researchers in the digital humanities and social scientists and give them better computational tools to analyze texts of interest to their own communities.

The instability of word embeddings also impacts researchers who use embeddings to achieve better performance on downstream tasks. In this more computational direction, there are several interesting avenues of future research. First, more research is needed to see how the stability of embeddings affects downstream tasks. This is primarily of interest to experiments that use smaller datasets or less computationally expensive architectures. While huge pre-trained models are gaining popularity and being used for many tasks, we still see a need for research into smaller, task-specific architectures. These will continue to be important for domains with small amounts of data, as well as languages with few resources. This dissertation has presented evidence that when using smaller architectures, stability has an impact on downstream performance. Continuing to quantify this is an important first step. There is also work to be done in learning how to mitigate this instability. Perhaps certain architectures or design decisions will help to stabilize embeddings; this may look different for different tasks and languages.

We have begun to explore how word embedding stability affects languages other than English, but there is more work to be done in this area. As mentioned above, it would be interesting to look at how stability affects performance on a multilingual task. One direction of research coming out of this dissertation would be to use the linguistic analyses that we have presented here in order to design better word embeddings for low-resource languages. We have seen that properties such as how gendered a language is and whether a language has affixing affect the stability of word embeddings. These, and other linguistic properties, should be taken into account when designing new embedding algorithms specifically for low-resource languages. Integrating linguistic knowledge into these algorithms will be a fruitful direction to explore.

Finally, in this work, we have expanded the idea of stability to contextualized output embeddings using paraphrases. There are still many open research questions that could be analyzed here, such as how fine-tuning affects BERT's ability to recognize paraphrases, and

how other contextualized output embeddings process paraphrases. Additionally, it would be interesting to look at how BERT could be used in the generation of paraphrases, which have proved to be a valuable source of data for many NLP tasks. Our analysis of BERT also shows areas where BERT incorrectly handles certain scenarios. For instance, words that are farther apart in a pair of phrases will have lower cosine similarity. This is not a desirable characteristic of BERT, and more work needs to be done in order to mitigate this effect.

This dissertation builds on much previous research analyzing word embeddings, and in turn suggests directions for future research. The analyses and experiments presented here contribute to the growing science of word representation and will enable advances in natural language processing pipelines.

BIBLIOGRAPHY

[1] ABDULRAHIM, A. Z. Ideological drifts in the US constitution: Detecting areas of contention with models of semantic change. In *NeurIPS Joint Workshop on AI for Social Good* (2019).

[2] ADAMS, O., MAKARUCHA, A., NEUBIG, G., BIRD, S., AND COHN, T. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (2017), pp. 937–947.

[3] AHO, A. V., AND ULLMAN, J. D. *The theory of parsing, translation, and compiling*, vol. 1. Prentice-Hall Englewood Cliffs, NJ, 1972.

[4] AL-RFOU', R., PEROZZI, B., AND SKIENA, S. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (2013), Association for Computational Linguistics, pp. 183–192.

[5] ALABI, J., AMPONSAH-KAAKYIRE, K., ADELANI, D., AND ESPAÑA-BONET, C. Massive vs. curated embeddings for low-resourced languages: The case of Yorùbá and Twi. In *Proceedings of The 12th Language Resources and Evaluation Conference* (2020), European Language Resources Association, pp. 2754–2762.

[6] ANTONIAK, M., AND MIMNO, D. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics 6*, 0 (2018), 107–119.

[7] ARORA, S., MAY, A., ZHANG, J., AND RÉ, C. Contextual embeddings: When are they worth it? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), Association for Computational Linguistics, pp. 2650–2663.

[8] ARTETXE, M., LABAKA, G., LOPEZ-GAZPIO, I., AND AGIRRE, E. Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning* (2018), pp. 282–291.

[9] BAKAROV, A. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536* (2018).

[10] BAKER, C. F., FILLMORE, C. J., AND LOWE, J. B. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* (1998), Association for Computational Linguistics, pp. 86–90.

[11] BANNARD, C., AND CALLISON-BURCH, C. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics* (2005), Association for Computational Linguistics, pp. 597–604.

[12] BENGIO, Y., LOURADOUR, J., COLLOBERT, R., AND WESTON, J. Curriculum learning. In *Proceedings of the International Conference on Machine Learning* (2009), pp. 41–48.

[13] BENTIVOGLI, L., BERNARDI, R., MARELLI, M., MENINI, S., BARONI, M., AND ZAMPARELLI, R. SICK through the SemEval glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Language Resources and Evaluation 50*, 1 (2016), 95–124.

[14] BIRD, S., KLEIN, E., AND LOPER, E. *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[15] BOLUKBASI, T., CHANG, K.-W., ZOU, J. Y., SALIGRAMA, V., AND KALAI, A. T. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in Neural Information Processing Systems* (2016), pp. 4349–4357.

[16] BOTARLEANU, R.-M., DASCALU, M., CROSSLEY, S. A., AND MCNAMARA, D. S. Sequence-to-sequence models for automated text simplification. In *Artificial Intelligence in Education* (Cham, 2020), I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, and E. Millán, Eds., Springer International Publishing, pp. 31–36.

[17] BULLINARIA, J. A., AND LEVY, J. P. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods 39*, 3 (2007), 510–526.

[18] CALISKAN, A., BRYSON, J. J., AND NARAYANAN, A. Semantics derived automatically from language corpora contain human-like biases. *Science 356*, 6334 (2017), 183–186.

[19] CER, D., DIAB, M., AGIRRE, E., LOPEZ-GAZPIO, I., AND SPECIA, L. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation* (2017), Association for Computational Linguistics, pp. 1–14.

[20] CHEN, X., AND CARDIE, C. Unsupervised multilingual word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 261–270.

[21] CHIANG, D. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics* (June 2005), Association for Computational Linguistics, pp. 263–270.

[22] CHIMALAMARRI, S., SITARAM, D., AND JAIN, A. Morphological segmentation to improve crosslingual word embeddings for low resource languages. *ACM Trans. Asian Low-Resour. Lang. Inf. Process. 19*, 5 (June 2020).

[23] CHURCH, K. W., AND HANKS, P. Word association norms, mutual information, and lexicography. *Computational Linguistics 16*, 1 (1990), 22–29.

[24] CIRIK, V., HOVY, E., AND MORENCY, L.-P. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204* (2016).

[25] COLLOBERT, R., WESTON, J., BOTTOU, L., KARLEN, M., KAVUKCUOGLU, K., AND KUKSA, P. Natural language processing (almost) from scratch. *Journal of Machine Learning Research 12*, Aug (2011), 2493–2537.

[26] CORBETT, G. G. Number of genders. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[27] CORBETT, G. G. Sex-based and non-sex-based gender systems. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[28] CORBETT, G. G. Systems of gender assignment. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[29] COTTERELL, R., AND SCHÜTZE, H. Morphological word-embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2015), Association for Computational Linguistics, pp. 1287–1292.

[30] DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[31] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019), pp. 4171–4186.

[32] DOLAN, B., QUIRK, C., AND BROCKETT, C. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics* (2004), COLING, pp. 350–356.

[33] DRYER, M. S. Position of case affixes. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[34] DRYER, M. S. Position of tense-aspect affixes. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[35] DRYER, M. S. Prefixing vs. suffixing in inflectional morphology. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[36] DRYER, M. S., AND HASPELMATH, M., Eds. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[37] EBERHARD, D. M., SIMONS, G. F., AND FENNIG, C. D., Eds. *Ethnologue: Languages of the World*, 23 ed. SIL International, Dallas, Texas, 2020.

[38] ECKART, C., AND YOUNG, G. The approximation of one matrix by another of lower rank. *Psychometrika 1*, 3 (1936), 211–218.

[39] ETHAYARAJH, K. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (2019), Association for Computational Linguistics, pp. 55–65.

[40] FARUQUI, M., DODGE, J., JAUHAR, S. K., DYER, C., HOVY, E., AND SMITH, N. A. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2015), Association for Computational Linguistics, pp. 1606–1615.

[41] FELLBAUM, C. *WordNet*. Wiley Online Library, 1998.

[42] FINKELSTEIN, L., GABRILOVICH, E., MATIAS, Y., RIVLIN, E., SOLAN, Z., WOLFMAN, G., AND RUPPIN, E. Placing search in context: The concept revisited. In *Proceedings of the International Conference on World Wide Web* (2001), pp. 406–414.

[43] FIRTH, J. R. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis* (1957).

[44] FRANCIS, W. N., AND KUCERA, H. Brown corpus manual. *Brown University 2* (1979).

[45] GANITKEVITCH, J., VAN DURME, B., AND CALLISON-BURCH, C. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013), Association for Computational Linguistics, pp. 758–764.

[46] GARG, N., SCHIEBINGER, L., JURAFSKY, D., AND ZOU, J. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences* (2018).

[47] GARIMELLA, A., BANEA, C., AND MIHALCEA, R. Demographic-aware word associations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), Association for Computational Linguistics, pp. 2285–2295.

[48] GLADKOVA, A., AND DROZD, A. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP* (2016), Association for Computational Linguistics, pp. 36–42.

[49] GREENWALD, A. G., MCGHEE, D. E., AND SCHWARTZ, J. L. Measuring individual differences in implicit cognition: The implicit association test. *Journal of Personality and Social Psychology 74*, 6 (1998), 1464.

[50] HALAWI, G., DROR, G., GABRILOVICH, E., AND KOREN, Y. Large-scale learning of word relatedness with constraints. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (2012), pp. 1406–1414.

[51] HAMILTON, W. L., LESKOVEC, J., AND JURAFSKY, D. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), Association for Computational Linguistics, pp. 1489–1501.

[52] HARRIS, Z. S. Distributional structure. In *Papers in Structural and Transformational Linguistics*. Springer, 1970, pp. 775–794.

[53] HELLRICH, J., BUECHEL, S., AND HAHN, U. Modeling word emotion in historical language: Quantity beats supposed stability in seed word selection. In *Proceedings of the 3rd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature* (2019), Association for Computational Linguistics, pp. 1–11.

[54] HELLRICH, J., AND HAHN, U. Bad Company—Neighborhoods in neural embedding spaces considered harmful. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016), The COLING 2016 Organizing Committee, pp. 2785–2796.

[55] HEYMAN, T., AND HEYMAN, G. Can prediction-based distributional semantic models predict typicality? *Quarterly Journal of Experimental Psychology 72*, 8 (2019), 2084–2109. PMID: 30704340.

[56] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural Computation 9*, 8 (1997), 1735–1780.

[57] HOERL, A. E., AND KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*, 1 (1970), 55–67.

[58] HOTELLING, H. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology 24*, 6 (1933), 417.

[59] HOWARD, J., AND RUDER, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (2018), Association for Computational Linguistics, pp. 328–339.

[60] HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering 9*, 3 (2007), 90–95.

[61] JIA, X., ZHOU, W., SUN, X., AND WU, Y. How to ask good questions? Try to leverage paraphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), Association for Computational Linguistics.

[62] JIANG, C., YU, H.-F., HSIEH, C.-J., AND CHANG, K.-W. Learning word embeddings for low-resource languages by PU learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2018), pp. 1024–1034.

[63] JOHNSON, J., DOUZE, M., AND JGOU, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019), 1–1.

[64] JOSHI, I., KORINGA, P., AND MITRA, S. Word embeddings in low resource Gujarati language. In *2019 International Conference on Document Analysis and Recognition Workshops* (2019), vol. 5, pp. 110–115.

[65] JOULIN, A., GRAVE, E., BOJANOWSKI, P., AND MIKOLOV, T. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (2017), Association for Computational Linguistics, pp. 427–431.

[66] KENTER, T., AND DE RIJKE, M. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (New York, NY, USA, 2015), CIKM 15, Association for Computing Machinery, p. 14111420.

[67] KOEHN, P. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X* (2005), vol. 5, pp. 79–86.

[68] LAMPLE, G., AND CONNEAU, A. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291* (2019).

[69] LENGERICH, B., MAAS, A., AND POTTS, C. Retrofitting distributional embeddings to knowledge graphs with functional relations. In *Proceedings of the 27th International Conference on Computational Linguistics* (2018), Association for Computational Linguistics, pp. 2423–2436.

[70] LEVY, O., AND GOLDBERG, Y. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (2014), Association for Computational Linguistics, pp. 302–308.

[71] LEVY, O., AND GOLDBERG, Y. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems* (2014), pp. 2177–2185.

[72] LEVY, O., GOLDBERG, Y., AND DAGAN, I. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics 3* (2015), 211–225.

[73] LI, Z., DING, X., AND LIU, T. Story ending prediction by transferable BERT. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (2019), International Joint Conferences on Artificial Intelligence Organization, pp. 1800–1806.

[74] LIU, Y., OTT, M., GOYAL, N., DU, J., JOSHI, M., CHEN, D., LEVY, O., LEWIS, M., ZETTLEMOYER, L., AND STOYANOV, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[75] LOZA, V., LAHIRI, S., MIHALCEA, R., AND LAI, P.-H. Building a dataset for summarization and keyword extraction from emails. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation* (2014), European Languages Resources Association, pp. 2441–2446.

[76] LUO, B., FENG, Y., WANG, Z., ZHU, Z., HUANG, S., YAN, R., AND ZHAO, D. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (2017), Association for Computational Linguistics, pp. 430–439.

[77] MARCUS, M. P., MARCINKIEWICZ, M. A., AND SANTORINI, B. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics 19*, 2 (1993), 313–330.

[78] MAXWELL, M., AND HUGHES, B. Frontiers in linguistic annotation for lower-density languages. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006* (2006), Association for Computational Linguistics, pp. 29–37.

[79] MCCARTHY, A. D., WICKS, R., LEWIS, D., MUELLER, A., WU, W., ADAMS, O., NICOLAI, G., POST, M., AND YAROWSKY, D. The Johns Hopkins University Bible corpus: 1600+ tongues for typological exploration. In *Proceedings of The 12th Language Resources and Evaluation Conference* (2020), European Language Resources Association, pp. 2884–2892.

[80] MCCARTHY, D., AND NAVIGLI, R. SemEval-2007 task 10: English lexical substitution task. In *Proceedings of the International Workshop on Semantic Evaluations* (2007), pp. 48–53.

[81] MCKINNEY, W. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (2010), Stéfan van der Walt and Jarrod Millman, Eds., pp. 56 – 61.

[82] MICKUS, T., PAPERNO, D., CONSTANT, M., AND VAN DEEMETER, K. What do you mean, BERT? Assessing BERT as a distributional semantics model. *Proceedings of the Society for Computation in Linguistics 3* (2020).

[83] MIKOLOV, T., CHEN, K., CORRADO, G., AND DEAN, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[84] MIKOLOV, T., KARAFIÁT, M., BURGET, L., CERNOCKỲ, J., AND KHUDANPUR, S. Recurrent neural network based language model. In *Interspeech* (2010), vol. 2, p. 3.

[85] MIKOLOV, T., LE, Q. V., AND SUTSKEVER, I. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* (2013).

[86] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems* (2013), pp. 3111–3119.

[87] MILLER, G. A. WordNet: A lexical database for English. *Communications of the ACM 38*, 11 (1995), 39–41.

[88] MIMNO, D., AND THOMPSON, L. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), Association for Computational Linguistics, pp. 2873–2878.

[89] MOLINO, P., WANG, Y., AND ZHANG, J. Parallax: Visualizing and understanding the semantics of embedding spaces via algebraic formulae. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2019), pp. 165–180.

[90] NEUBIG, G., DYER, C., GOLDBERG, Y., MATTHEWS, A., AMMAR, W., ANASTASOPOULOS, A., BALLESTEROS, M., CHIANG, D., CLOTHIAUX, D., COHN, T., DUH, K., FARUQUI, M., GAN, C., GARRETTE, D., JI, Y., KONG, L., KUNCORO, A., KUMAR, G., MALAVIYA, C., MICHEL, P., ODA, Y., RICHARDSON, M., SAPHRA, N., SWAYAMDIPTA, S., AND YIN, P. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* (2017).

[91] NEWMAN-GRIFFIS, D., AND FOSLER-LUSSIER, E. Second-order word embeddings from nearest neighbor topological features. *arXiv preprint arXiv:1705.08488* (2017).

[92] NIVRE, J., DE MARNEFFE, M.-C., GINTER, F., GOLDBERG, Y., HAJIČ, J., MANNING, C. D., MCDONALD, R., PETROV, S., PYYSALO, S., SILVEIRA, N., TSARFATY, R., AND ZEMAN, D. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation* (2016), pp. 1659–1666.

[93] PAVLICK, E., RASTOGI, P., GANITKEVITCH, J., VAN DURME, B., AND CALLISON-BURCH, C. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), Association for Computational Linguistics, pp. 425–430.

[94] PEARSON, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2*, 11 (1901), 559–572.

[95] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[96] PENNINGTON, J., SOCHER, R., AND MANNING, C. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014), Association for Computational Linguistics, pp. 1532–1543.

[97] PÉREZ-ROSAS, V., ABOUELENIEN, M., MIHALCEA, R., AND BURZO, M. Deception detection using real-life trial data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (2015), ACM, pp. 59–66.

[98] PÉREZ-ROSAS, V., AND MIHALCEA, R. Experiments in open domain deception detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), pp. 1120–1125.

[99] PETERS, M., NEUMANN, M., IYYER, M., GARDNER, M., CLARK, C., LEE, K., AND ZETTLEMOYER, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2018), pp. 2227–2237.

[100] PETROV, S., DAS, D., AND MCDONALD, R. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation* (2012), European Languages Resources Association, pp. 2089–2096.

[101] PIERREJEAN, B., AND TANGUY, L. Towards qualitative word embeddings evaluation: Measuring neighbors variation. In *Proceedings of the 2018 Conference of the*

North American Chapter of the Association for Computational Linguistics: Student Research Workshop (2018), pp. 32–39.

[102] QUIRK, C., BROCKETT, C., AND DOLAN, W. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing* (Barcelona, Spain, July 2004), Association for Computational Linguistics, pp. 142–149.

[103] RADINSKY, K., AGICHTEIN, E., GABRILOVICH, E., AND MARKOVITCH, S. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the International Conference on World Wide Web* (2011), pp. 337–346.

[104] RAJPURKAR, P., ZHANG, J., LOPYREV, K., AND LIANG, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016), Association for Computational Linguistics, pp. 2383–2392.

[105] RASTOGI, P., VAN DURME, B., AND ARORA, R. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2015), Association for Computational Linguistics, pp. 556–566.

[106] ROGERS, A., DROZD, A., AND LI, B. The (too many) problems of analogical reasoning with word vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics* (2017), Association for Computational Linguistics, pp. 135–148.

[107] ROGERS, A., KOVALEVA, O., AND RUMSHISKY, A. A primer in BERTology: What we know about how BERT works. *arXiv preprint arXiv:2002.12327* (2020).

[108] RUDER, S., VULIĆ, I., AND SØGAARD, A. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research 65* (2019), 569–631.

[109] SANDHAUS, E. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia 6*, 12 (2008), e26752.

[110] SANH, V., DEBUT, L., CHAUMOND, J., AND WOLF, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[111] SIEWIERSKA, A. Gender distinctions in independent personal pronouns. In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[112] SINGH, A. D., MEHTA, P., HUSAIN, S., AND RAJAKRISHNAN, R. Quantifying sentence complexity based on eye-tracking measures. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity* (2016), pp. 202–212.

[113] SMITH, S., JAN KINDERMANS, P., YING, C., AND LE, Q. V. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations* (2018).

[114] SOLER-COMPANY, J., AND WANNER, L. On the relevance of syntactic and discourse features for author profiling and identification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (2017), Association for Computational Linguistics, pp. 681–687.

[115] SPEARMAN, C. Correlation calculated from faulty data. *British Journal of Psychology, 1904-1920 3*, 3 (1910), 271–295.

[116] SPITKOVSKY, V. I., ALSHAWI, H., AND JURAFSKY, D. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010), Association for Computational Linguistics, pp. 751–759.

[117] STRUBELL, E., GANESH, A., AND MCCALLUM, A. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), Association for Computational Linguistics, pp. 3645–3650.

[118] SUN, C., QIU, X., XU, Y., AND HUANG, X. How to fine-tune BERT for text classification? In *Chinese Computational Linguistics* (Cham, 2019), M. Sun, X. Huang, H. Ji, Z. Liu, and Y. Liu, Eds., Springer International Publishing, pp. 194–206.

[119] TAN, L., ZHANG, H., CLARKE, C., AND SMUCKER, M. Lexical comparison between Wikipedia and Twitter corpora by using word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), Association for Computational Linguistics, pp. 657–661.

[120] TATMAN, R., AND PAULLADA, A. Social identity and punctuation variation in the #BlueLivesMatter and #BlackLivesMatter Twitter communities. In *The 33rd Northwest Linguistics Conference* (2017).

[121] TAYLOR, W. L. "Cloze procedure": A new tool for measuring readability. *Journalism Bulletin 30*, 4 (1953), 415–433.

[122] TSVETKOV, Y., FARUQUI, M., LING, W., MACWHINNEY, B., AND DYER, C. Learning the curriculum with Bayesian optimization for task-specific word representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), Association for Computational Linguistics, pp. 130–139.

[123] TURNEY, P. D., AND PANTEL, P. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research 37* (2010), 141–188.

[124] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), pp. 5998–6008.

[125] WANG, A., SINGH, A., MICHAEL, J., HILL, F., LEVY, O., AND BOWMAN, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (2018), pp. 353–355.

[126] WENDLANDT, L., KUMMERFELD, J. K., AND MIHALCEA, R. Factors influencing the surprising instability of word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2018), pp. 2092–2102.

[127] WIEDEMANN, G., REMUS, S., CHAWLA, A., AND BIEMANN, C. Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings. In *Konferenz zur Verarbeitung natrlicher Sprache / Conference on Natural Language Processing* (2019), pp. 161–170.

[128] WILSON, S., AND MIHALCEA, R. Measuring semantic relations between human activities. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing* (2017), Asian Federation of Natural Language Processing, pp. 664–673.

[129] YAGHOOBZADEH, Y., AND SCHÜTZE, H. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), pp. 236–246.

[130] YANG, Z., DAI, Z., YANG, Y., CARBONELL, J., SALAKHUTDINOV, R. R., AND LE, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*. 2019, pp. 5753–5763.

[131] YI, J., AND TAO, J. Self-attention based model for punctuation prediction using word and speech embeddings. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (2019), pp. 7270–7274.

[132] ZELLERS, R., BISK, Y., SCHWARTZ, R., AND CHOI, Y. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), Association for Computational Linguistics, pp. 93–104.

[133] ZHANG, X., ZHAO, J., AND LECUN, Y. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems* (2015), pp. 649–657.

[134] ZHANG, Z., HAN, X., LIU, Z., JIANG, X., SUN, M., AND LIU, Q. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th*

Annual Meeting of the Association for Computational Linguistics (2019), Association for Computational Linguistics, pp. 1441–1451.

[135] ZHAO, M., AND SCHÜTZE, H. A multilingual BPE embedding space for universal sentiment lexicon induction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics* (2019), pp. 3506–3517.

[136] ZHAO, S., GUPTA, R., SONG, Y., AND ZHOU, D. Extreme language model compression with optimal subwords and shared projections. *arXiv preprint arXiv:1909.11687* (2019).