Understanding the Downstream Instability of Word Embeddings

Anonymous Authors[1]

Abstract

Many industrial machine learning (ML) systems require frequent retraining to keep up-to-date with constantly changing data. This retraining exacerbates a large challenge facing ML systems today: model training is unstable, i.e., small changes in training data can cause significant changes in the model's predictions. In this paper, we work on developing a deeper understanding of this instability, with a focus on how a core building block of modern natural language processing (NLP) pipelines—pre-trained word embeddings—affects the instability of downstream NLP models. We first empirically reveal a tradeoff between stability and memory: increasing the embedding memory $2\times$ can reduce the disagreement in predictions due to small changes in training data by 5% to 39% (relative). To theoretically explain this tradeoff, we introduce a new measure of embedding instability—the eigenspace instability measure. We relate the eigenspace instability measure to downstream instability by proving a bound on the disagreement in downstream predictions introduced by the change in word embeddings. Practically, we show that the eigenspace instability measure can be a cost-effective way to choose embedding parameters to minimize instability without training downstream models, achieving up to $3.71\times$ lower error rates than existing embedding distance measures. Finally, we demonstrate that the observed stability-memory tradeoffs extend to other types of embeddings as well, including knowledge graph and contextual word embeddings.

1 Introduction

Data is more dynamic than ever before: every input, interaction, and response is captured and archived in hopes of extracting insights with machine learning (ML) models. To stay up-to-date, models must be frequently retrained, with the freshness of models becoming a requirement for user satisfaction in numerous products, from ads (He et al., 2014) to recommendation systems (Covington et al., 2016). However, frequent retraining can lead to large and unwanted fluctuations in model predictions due to the *instability* of many machine learning training algorithms: minimal changes in training data can produce significantly different predictions (Fard et al., 2016). From discussions with engineers in an e-commerce firm, an online social media company, and a Fortune 500 software company, we found that instability from retraining is one of their largest, and also most under-addressed, pain points. As a result of instability, ML engineers struggle to identify genuine concept shifts, spend more time tracking down regressions, and require more resources retraining downstream model dependencies. Diagnosing and reducing instability in a cost-effective way is a major challenge for today's machine learning pipelines.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Systems and Machine Learning (SysML) Conference. Do not distribute.

In this work, we take a first step toward addressing the problem of ML model instability by examining in detail a core building block of most modern natural language processing (NLP) applications: word embeddings (Mikolov et al., 2013a;b; Pennington et al., 2014; Bojanowski et al., 2017). Several recent works have shown that word embeddings are unstable, with the nearest neighbors to words varying significantly across embeddings trained under different settings (Hellrich & Hahn, 2016; Antoniak & Mimno, 2018; Wendlandt et al., 2018; Pierrejean & Tanguy, 2018; Chugh et al., 2018; Hellrich et al., 2019). These results force researchers using embeddings for analysis to reassess the reliability of their conclusions. Moreover, these results raise questions about *how the embedding instability impacts downstream NLP tasks*—an area which remains largely unexplored and which we focus on in this work. We define the downstream instability between a pair of word embeddings as the percentage of predictions which change between the models trained on the two embeddings for a given task. By this notion of instability, we find that 15% of predictions on a sentiment analysis task can disagree due to training the embeddings on an accumulated dataset with just 1% more data. In embedding servers, where an embedding is reused among multiple downstream tasks (Hermann & Balso, 2017; Gordon, 2018; Shiebler et al., 2018; Sell & Pienaar, 2019), the impact of this instability can be quickly amplified. Understanding this downstream instability is challenging, however, because it requires *both theoretical* *and empirical* insights on how the embedding instability propagates to the downstream tasks.

The goal of this paper is to develop a deeper understanding of the downstream instability of word embeddings. This understanding could both drive the design choices for embedding systems (i.e. choosing hyperparameters) and lead to efficient techniques to distinguish among unstable and stable embeddings without training downstream models. To achieve this, we perform a study on the downstream instability of word embeddings across multiple embedding algorithms and downstream tasks. Our study exposes a novel trade-off between stability and another critical property of embeddings—*memory*. We find that increasing the memory can lead to more stable embeddings, with a 2× increase in memory reducing the percentage prediction disagreement on downstream tasks by 5% to 39% (relative). Determining how the memory affects the instability is not straightforward: factors like the *dimension*, a hyperparameter controlling the expressiveness of the embedding, and the *precision*, the number of bits used per entry in the embedding after compression, can independently affect the instability and interact in unexpected ways. To better understand the stability-memory tradeoff empirically, we study the effects of dimension and precision both in isolation and together. This important stability-memory tradeoff leads us to ask two key questions: (1) theoretically, how can we explain this tradeoff, and (2) practically, how can we select the dimension-precision[1] parameters to minimize the downstream instability?

To theoretically explain the stability-memory trade-off, we introduce a new measure for embedding instability—the eigenspace instability measure—which we theoretically relate to downstream instability in the case of linear regression models. The eigenspace instability measure builds on the eigenspace overlap score (May et al., 2019), and measures the degree of similarity between the eigenvectors of the Gram matrices of a pair of embeddings, weighted by their eigenvalues. We show that the expected downstream disagreement between the linear regression models trained on two embedding matrices can be expressed in terms of the eigenspace instability measure. Furthermore, these theoretical insights have a practical application: we propose using the eigenspace instability measure to efficiently select dimension-precision parameters with low downstream instability, without having to train downstream models.

We empirically validate that the eigenspace instability measure correlates strongly with the downstream instability and that the measure is effective as a selection criterion for the dimension-precision parameters. First, we show that the theoretically grounded eigenspace instability measure more strongly correlates with downstream instability than the majority of the other embedding distance measures (i.e. semantic displacement (Hamilton et al., 2016), the PIP loss (Yin & Shen, 2018), and the eigenspace overlap score (May et al., 2019)) and attains Spearman correlations from 0.04 better to 0.09 worse than the other top-performing measure, the k-NN measure (e.g., Hellrich & Hahn (2016); Antoniak & Mimno (2018); Wendlandt et al. (2018)), which lacks theoretical guarantees. Next, we show that when using an embedding distance measure to choose the more stable dimension-precision parameters out of a pair of choices, the eigenspace instability measure achieves up to 3.71× lower error rates than the weaker baselines and from $0.93\times$ to $1.55\times$ the error rate of the k-NN measure. On the more challenging task of selecting the combination of dimension and precision under a memory budget, we show that eigenspace instability measure attains a difference in prediction disagreement to the oracle up to 3.06% (absolute) better than the weaker baselines and within 0.46% (absolute) of the k-NN measure.

To summarize, we make the following contributions:

- We study the downstream instability of word embeddings, revealing a novel stability-memory tradeoff. In particular, we study the impact of two key parameters, dimension and precision, and propose a simple rule of thumb relating the embedding memory and downstream instability (Section 3).
- To theoretically explain this tradeoff, we introduce a new measure for embedding instability, the eigenspace instability measure, that we prove theoretically determines the expected downstream disagreement on a linear regression task (Section 4).
- To empirically validate our theory, we perform an evaluation of embedding distance measures to predict downstream instability. Practically, we show that the eigenspace instability measure can achieve up to 3.71× lower error rates in selecting more stable dimension-precision parameters than existing embedding distance measures (Section 5).
- Finally, we show that the stability-memory tradeoffs extend to knowledge graph embeddings (Bordes et al., 2013) and contextual word embeddings, such as BERT embeddings (Devlin et al., 2019). For instance, we find that increasing the memory of knowledge graph embeddings 2× decreases the instability on a link prediction task by 7% to 19% (relative) (Section 6).

2 Preliminaries

We begin by formally defining the notion of instability we use in this work. We then review the word embedding algorithms and compression technique used in our study, and discuss existing measures to compare two embeddings.

[1] For brevity, we refer to a pair of dimension and precision parameters as the "dimension-precision" parameters.

2.1 Instability Definition

We define the downstream instability as follows:

Definition 1. *Let $X \in \mathbb{R}^{n\times d}$ and $\tilde{X} \in \mathbb{R}^{n\times k}$ be two embedding matrices, and let f_X and $f_{\tilde{X}}$ represent models trained using X and $\tilde{X}$, respectively, for a downstream task T. Then the instability between X and $\tilde{X}$ with respect to task T is defined as*

$$\mathcal{DI}_T(X, \tilde{X}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f_X(z_i), f_{\tilde{X}}(z_i)),$$

where $\{z_i\}_{i=1}^n$ is a heldout set for task T, and $\mathcal{L}$ is a fixed loss function.

When the zero-one loss is used for $\mathcal{L}$, this measure captures the percentage of predictions which disagree on downstream models trained on each embedding.

2.2 Word Embedding Algorithms

Word embedding algorithms learn distributed representations of words by taking as input a textual corpus C and returning the word embedding $X \in \mathbb{R}^{n\times d}$, with d the dimension of the embeddings and n the vocabulary size. We evaluate matrix completion (MC) (Jin et al., 2016) and continuous bag-of-words (CBOW) (Mikolov et al., 2013a;b) embedding algorithms. MC explicitly factors the co-occurrence matrix $A \in \mathbb{R}^{n\times n}$ generated from C, whereas CBOW operates on the sequential corpus C directly. We elaborate below.

Matrix completion (MC) Matrix completion uses the word embeddings to approximate the observed word co-occurrence A and can be formally written as:

$$V = \arg\min_X \sum_{(i,j)\in\Theta} (X_i X_j^T - A_{ij})^2$$

where Θ are the observed (non-zero) entries in A. Following standard technique, A is the positive pointwise mutual information (PPMI) matrix, rather than the true co-occurrence matrix (Bullinaria & Levy, 2007).

We solve the matrix completion problem using an online algorithm similar to that proposed in Jin et al. (2016). We iteratively train X via stochastic gradient descent (SGD) after computing the loss on sampled entries of the observed co-occurrence matrix A.

Continuous bag-of-words (CBOW) The CBOW algorithm predicts a word given its local context words (Mikolov et al., 2013a;b). The embedding matrix X is trained via SGD, where the loss maximizes the probability that an observed word and context pair co-occurs in the corpus and minimizes the probability that a negative sample co-occurs. We use the `word2vec` implementation of CBOW.[2]

[2]https://github.com/tmikolov/word2vec

2.3 Compression Technique

We use a standard technique—uniform quantization—to compress word embeddings. Recent work (May et al., 2019) demonstrates that uniform quantization performs on par in terms of downstream quality with more complex compression techniques, such as k-means compression (Andrews, 2016) and deep compositional code learning (Shu & Nakayama, 2018). We leverage their implementation[3] to apply uniform quantization to word embeddings to study the impact of the precision on instability. Under uniform quantization, each entry in the word embedding matrix is rounded to a discrete value in a set of 2^b equally spaced values within an interval, such that each entry can be represented with just b bits. For more details on the way we use uniform quantization for our experiments, see Appendix B.2.

2.4 Embedding Distance Measures

We consider four embedding distance measures from the literature to quantify the differences between embeddings. For each measure, we assume we have a pair of embeddings $X \in \mathbb{R}^{n\times d}$ and $\tilde{X} \in \mathbb{R}^{n\times d}$ trained on corpora C and $\tilde{C}$, respectively, where n is the size of the vocabulary and d is the dimension of the embedding. Due to computational efficiency and our observation that downstream tasks use a majority of high frequency words, we only consider the top 10k most frequent words to compute each measure (including the eigenspace instability measure).

k-Nearest Neighbors (k-NN) Measure Variants of the k-NN measure were used in recent works on word embedding stability to characterize the intrinsic stability of embeddings (e.g., Hellrich & Hahn (2016); Antoniak & Mimno (2018); Wendlandt et al. (2018)). The k-NN measure is defined as $\frac{1}{Q}\sum_{q=0}^{Q} \frac{|\mathrm{N}_k(X;q)\cap \mathrm{N}_k(\tilde{X};q)|}{k}$, where Q is the number of randomly sampled query words (we use Q=1000), and the N_k function takes an embedding and the index of a query word, and returns the indices of the k most similar words to the query word by the cosine distance.

Semantic Displacement Researchers have used semantic displacement to compute the distance that words have shifted over time (Hamilton et al., 2016). Semantic displacement can be defined as $\frac{1}{n}\sum_{i=0}^{n}$ cos-dist$(X_i, R\tilde{X}_i)$, where $R = \arg\min_\Omega ||X - \tilde{X}\Omega||_F$, subject to $\Omega^T\Omega = I$ (i.e., the orthogonal Procrustes solution (Schönemann, 1966)).

Pairwise Inner Product Loss The Pairwise Inner Product (PIP) loss was proposed for dimensionality selection to optimize for the intrinsic quality of an embedding (Yin & Shen, 2018). The PIP loss is defined as $\|XX^T - \tilde{X}\tilde{X}^T\|_F$.

[3]https://github.com/HazyResearch/smallfry

Eigenspace Overlap Score The eigenspace overlap score was recently proposed as a measure of compression quality (May et al., 2019). The eigenspace overlap is defined as $\frac{1}{d}\|U^T\tilde{U}\|_F^2$, where $X = USV^T$ and $\tilde{X} = \tilde{U}\tilde{S}\tilde{V}^T$ are the singular value decompositions (SVDs) of X and $\tilde{X}$.

3 A Stability-Memory Tradeoff

We now present the empirical study that exposes the tradeoff we observe between downstream stability and embedding memory, and demonstrate that as the memory increases, the instability decreases. We consider the dimension and precision of the embedding as two important axes controlling the memory of the embedding. We first study the impact of the embedding's dimension and precision on downstream instability in isolation in Sections 3.1 and 3.2, respectively, followed by a discussion of their joint effect in Section 3.3.

Corpora We use two full Wikipedia dumps[4]: Wiki'17 and Wiki'18, which we collected approximately a year apart, to train embeddings. The corpora are pre-processed by a Facebook script[5], which we modify to keep the letter cases. We use these two corpora as examples of the temporal changes which can occur to the text corpora used to train word embeddings. Each corpora has about 4.5 billion tokens, and when training the embeddings, we only learn the embeddings for the top 400k most frequent words.

Downstream NLP Tasks After training the word embeddings, we compress the embeddings with uniform quantization and train models for downstream NLP tasks on top of the embeddings, fixing the embeddings during training. We train word embeddings with three seeds, and use the same corresponding seeds for the downstream models. Results are reported as averages over the three seeds, with error bars indicating the standard deviation. We also align all pairs of Wiki'17 and Wiki'18 embeddings (same dimension and seed) with orthogonal Procrustes (Schönemann, 1966) prior to compressing and training downstream models, as preliminary experiments found this helped to decrease instability. For each downstream task, we perform a hyperparameter search for the learning rate using 400-dimensional Wiki'17 embeddings, and use the same learning rate across all dimensions to minimize the impact of learning rate on our analysis. Here, we discuss the two standard downstream NLP tasks we consider throughout our paper. Please see Appendix B.3 for more experimental setup details.

Sentiment Analysis. We evaluate on a binary sentiment analysis task where given a sentence, the model determines if the sentence is positive or negative (Kim, 2014). We train a linear bag-of-words model for this task and evaluate on four benchmark datasets: MR (Pang & Lee, 2005), MPQA (Wiebe et al., 2005), Subj (Pang & Lee, 2004), and SST-2 (Socher et al., 2013b). We will be showing results on SST-2; for more results, see Appendix C.1.

[4]https://dumps.wikimedia.org

[5]https://github.com/facebookresearch/fastText/blob/master/get-wikimedia.sh

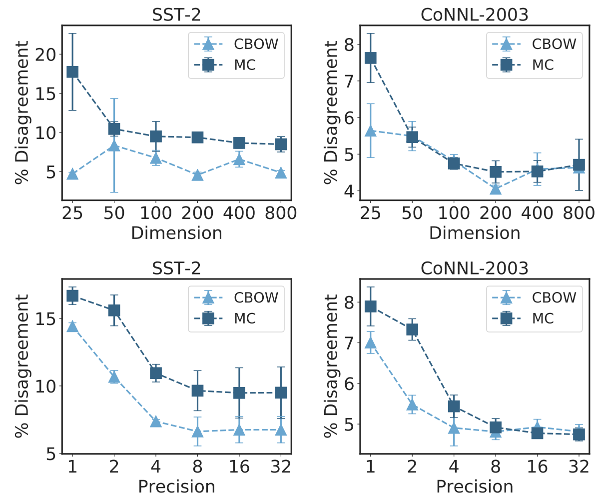


Figure 1. Downstream instability of sentiment analysis (SST-2) and NER (CoNLL-2003) tasks under different dimensions (top) and precisions (bottom) for CBOW and MC embeddings.

Named Entity Recognition (NER). The named entity recognition task is a multi-class classification task to predict whether each token in the dataset is an entity, and if so, what type. We use a BiLSTM model (Akbik et al., 2018) for this task and evaluate on the benchmark CoNLL-2003 dataset (Tjong Kim Sang & De Meulder, 2003). Each token is assigned an entity label of PER, ORG, LOC, and MISC, or an 'O' label, indicating outside of any entities (i.e., no entity). We measure instability only over the tokens for which the true value is an entity. We use the BiLSTM without the conditional random field (CRF) decoding layer for computational efficiency; in Appendix D.1 we show that the trends also hold on a subset of the results with a BiLSTM-CRF.

3.1 Effect of Dimension

We evaluate the impact of the dimension of the embedding on its downstream stability, and show that generally as the dimension increases, the instability decreases.

Tradeoffs To perform our tradeoff study, we train Wiki'17 and Wiki'18 embeddings with dimensions in {25, 50, 100, 200, 400, 800}, and train downstream models on top of the embeddings. We compute the prediction disagreement between models trained on Wiki'17 and Wiki'18 embeddings of the same dimension. In Figure 1 (top), we see that as the dimension increases, the downstream instability often decreases across embedding algorithms and downstream tasks, plateauing at larger dimensions. In Section 3.3, we see that these trends are even more pronounced in lower memory regimes when we also consider different precisions.

3.2 Effect of Precision

We evaluate the effect of the precision, the number of bits used to store each entry of the embedding matrix, on the downstream stability, and show that as the precision increases, the instability decreases.

Tradeoffs We compress 100-dimensional Wiki'17 and Wiki'18 embeddings with uniform quantization to precisions $b \in \{1, 2, 4, 8, 16, 32\}$,[6] and train downstream models on top of the compressed embeddings. We compute the prediction disagreement between models trained on Wiki'17 and Wiki'18 embeddings of the same precision. In Figure 1 (bottom), we show that as the precision increases, the instability generally decreases on sentiment analysis and NER tasks for both MC and CBOW embedding algorithms. Moreover, we see that for precisions greater than 4 bits, the impact of compression on instability is minimal.

3.3 Joint Effect of Dimension and Precision

We study the effect of dimension and precision together, and show that overall, as the memory increases, the downstream instability decreases. We also propose a simple rule of thumb relating the memory and instability, and evaluate the relative impact of dimension and precision on the instability. Finally, we discuss two key questions based on our empirical observations, which motivate the rest of the work.

Tradeoffs We uniformly quantize the Wiki'17 and Wiki'18 embeddings of dimensions {25, 50, 100, 200, 400, 800} to precisions {1, 2, 4, 8, 16, 32} to generate many dimension-precision pairs spanning over a wide range of memory budgets. Across the memory budgets, embedding algorithms, and tasks, we see that as we increase the memory, the downstream instability decreases (Figure 2). To propose a simple rule of thumb for the stability-memory tradeoff, we fit a single linear-log model to the dimension-precision pairs for all memory budgets less than 10^3 bits/word (after which the instability plateaus) across five downstream tasks (i.e., the four sentiment analysis tasks and one NER task) and two embedding algorithms. We find the following average stability-memory relationship for the downstream instability $\mathcal{DI}_T$ for a task T with respect to the memory, or bits/word, M : $\mathcal{DI}_T \approx C_T - 1.4 * log_2(M)$, where C_T is a task-specific constant. For instance, if we increase the memory 2×, then the instability decreases on average by 1.4% (absolute). Across the tasks, embedding algorithms, and memory budgets we consider, this 1.4%

[6] $b = 32$ signifies full-precision embeddings.

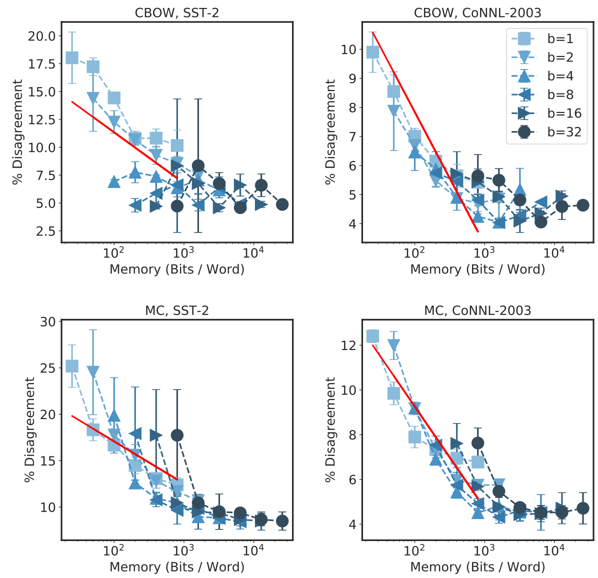


Figure 2. Downstream instability of sentiment analysis (SST-2) and NER (CoNLL-2003) tasks for various memory budgets with different dimension-precision combinations. The red line indicates the average linear-log model relating instability and memory.

(absolute) difference corresponds to an approximately 5% to 39% *relative* reduction in downstream instability, depending on the original instability value (3.6% to 25.9%). To understand the relative impact on instability of increasing the dimension vs. the precision, we fit independent linear-log models to each parameter. We find that precision has a larger impact on instability than dimension, with a 2× increase in precision decreasing instability by 1.5% (absolute) vs. a 2× increase in dimension decreasing instability by 1.2% (absolute). Please see Appendix B.4 for more details on how we fit these trends and Appendix D for results demonstrating the robustness of the stability-memory tradeoff (e.g., to more complex downstream models, other sources of downstream randomness).

This stability-memory tradeoff raises two key questions: (1) how can we theoretically explain this tradeoff between the embedding memory and the downstream stability, and (2) how can we jointly select the embedding's dimension-precision parameters to minimize the downstream instability? Practically, choosing these parameters is important, because as we can see in Figure 2, downstream instability can vary as much as 8% across the different combinations of dimension and precision within a given memory budget. The goal of the remainder of the paper will be to shed light on these questions.

4 Analyzing Embedding Instability

To address both questions raised above, we present a new measure of embedding instability, the eigenspace instability measure, which we show is both theoretically and empirically related to the downstream instability of the embeddings. The goal of this measure is to efficiently estimate, given two embeddings, how different the predictions of models trained with these embeddings will be. We first define the eigenspace instability measure and present its theoretical connection with downstream instability in Section 4.1; we then propose using this measure to efficiently select parameters to minimize downstream instability in Section 4.2.

4.1 Eigenspace Instability Measure

We now define the eigenspace instability measure between two embeddings, and show that this measure is directly related to the expected disagreement between linear regression models trained using these embeddings.

Definition 2. *Let $X = USV^T \in \mathbb{R}^{n\times d}$ and $\tilde{X} = \tilde{U}\tilde{S}\tilde{V}^T \in \mathbb{R}^{n\times k}$ be the singular value decompositions (SVDs) of two embedding matrices X and $\tilde{X}$, and let $\Sigma \in \mathbb{R}^{n\times n}$ be a positive semidefinite matrix. Then the eigenspace instability measure between X and $\tilde{X}$, with respect to Σ, is defined as*

$$\mathcal{EI}_\Sigma(X,\tilde{X}) := \frac{1}{\mathrm{tr}(\Sigma)}\,\mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^T UU^T\right)\Sigma\right).$$

Intuitively, this measure captures how different the subspaces spanned by the left singular vectors of X and $\tilde{X}$ are to one another; the measure will be equal to zero when the left singular vectors of X and $\tilde{X}$ span identical subspaces of $\mathbb{R}^n$, and will be equal to one when these singular vectors span orthogonal subspaces of $\mathbb{R}^n$ whose union covers the whole space. We note that the left singular vectors are particularly important in the case of linear regression models, because the predictions of the learned model on the training examples depend only on the label vector and the left singular vectors of the data matrix.[7]

We now present our result showing that the expected mean squared difference between the linear regression models trained on X vs. $\tilde{X}$ is equal to the the eigenspace instability measure, where Σ corresponds to the covariance matrix of the regression label vector. For the proof, see Appendix A.

Proposition 1. *Let $X \in \mathbb{R}^{n\times d}$, $\tilde{X} \in \mathbb{R}^{n\times k}$ be two full-rank embedding matrices, where x_i and $\tilde{x}_i$ correspond to the i^{th} rows of X and $\tilde{X}$ respectively. Let $y \in \mathbb{R}^n$ be a random regression label vector with zero mean and covariance $\Sigma \in \mathbb{R}^{n\times n}$. Then the (normalized) expected mean squared difference between the linear models f_y and $\tilde{f}_y$*[8] *trained on label vector y using embeddings X and $\tilde{X}$ satisfies*

$$\frac{\mathbb{E}_y\left[\sum_{i=1}^n (f_y(x_i) - \tilde{f}_y(\tilde{x}_i))^2\right]}{\mathbb{E}_y\left[\|y\|^2\right]} = \mathcal{EI}_\Sigma(X,\tilde{X}). \quad (1)$$

The above result exactly characterizes the expected downstream instability of linear regression models trained on X and $\tilde{X}$, in terms of the eigenspace instability measure, given the covariance matrix Σ of the label vector; but how should we select Σ? One desirable property for Σ could be that it produce label vectors with higher variance in directions believed to be important, for example because they correspond to eigenvectors with large eigenvalues of an embedding's Gram matrix. In Section 5, where we evaluate the instability of pairs of embeddings of various dimensions and precisions, we consider $\Sigma = (EE^T)^\alpha + (\tilde{E}\tilde{E}^T)^\alpha$; in those experiments, E and $\tilde{E}$ are the highest-dimensional ($d = 800$), full-precision embeddings for Wiki'17 and Wiki'18, respectively, and α is a scalar controlling the relative importance of the directions of high eigenvalue. This choice of Σ results in label vectors with large variance in the directions of high eigenvalues of these embedding matrices. In Section 5.1 we show that when α is chosen appropriately, there is strong empirical correlation between the eigenspace instability measure (with this Σ) and downstream instability.

4.2 Jointly Selecting Dimension and Precision

We now demonstrate a practical utility of the eigenspace instability measure: we propose using the measure to efficiently select embedding dimension-precision parameters to minimize downstream instability without training the downstream models. In particular, we propose an algorithm that takes two or more pairs of embeddings with different dimension-precision parameters as input, and outputs the pair with the lowest eigenspace instability measure between embeddings. In Section 5.2, we evaluate the performance of this proposed selection algorithm in two settings: first, a simple setting where the goal is to select the pair with the lowest downstream instability out of two randomly selected pairs, and second, a more challenging setting where the goal is to select the pair with the lowest downstream instability out of two or more pairs with the same memory budget. In both settings, we demonstrate that the eigenspace instability measure outperforms the majority of embedding distance measures and is competitive with the other top-performing embedding distance measure, the k-NN measure.

5 EXPERIMENTS

We now empirically validate the eigenspace instability measure's relation with downstream instability and demonstrate that the eigenspace instability measure is an effective selection criterion for dimension-precision parameters. In Sec-

[7] The linear model trained on data matrix $X = USV^T \in \mathbb{R}^{n\times d}$ with label vector $y \in \mathbb{R}^n$ makes predictions $Xw = X(X^TX)^{-1}X^Ty = UU^Ty \in \mathbb{R}^n$ on the n training points.

[8] $f_y(x) = w^Tx$, for $w = (X^TX)^{-1}X^Ty$, and $\tilde{f}_y(\tilde{x}) = \tilde{w}^T\tilde{x}$, for $\tilde{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^Ty$.

tion 5.1, we show that the theoretically grounded eigenspace instability measure strongly correlates with downstream instability, attaining Spearman correlations greater than the weaker baselines (semantic displacement, PIP loss, and eigenspace overlap score) and between 0.04 better and 0.09 worse than the strongest baseline (the k-NN measure). In Section 5.2, when selecting dimension-precision parameters without training the downstream models, we show that the eigenspace instability measure attains up to $3.71\times$ lower error rates than weaker baselines and from $0.93\times$ to $1.55\times$ the error rate of the k-NN measure.

Experimental Setup To evaluate how predictive the various embedding distance measures are of downstream instability, we take the embedding pairs and corresponding downstream models we trained in Section 3 and measure the embedding distance measures between these pairs of embeddings. Specifically, we compute the k-NN measure, semantic displacement, PIP loss, eigenspace overlap score, and eigenspace instability measure between the embedding pairs (Section 2.4). Recall that the k-NN measure and the eigenspace instability measure each have an important hyperparameter: the k in the k-NN measure, which determines how many neighbors we compare, and the α in the eigenspace instability measure, which controls how important the eigenvectors of high eigenvalue are. For both hyperparameters, we choose the values with the highest average correlation across four sentiment analysis tasks (SST-2, MR, Subj, and MPQA) and one NER task (CoNLL-2003) and two embedding algorithms (CBOW and MC) when using validation datasets for the downstream tasks (k = 5 and α = 3). See Appendix C.3 for more details on setting these values. The eigenspace instability measure also requires additional embeddings E and $\tilde{E}$: we use 800-dimensional, full-precision Wiki'17 and Wiki'18 embeddings as these are the highest dimensional, full-precision embeddings in our study.

5.1 Predictive Performance of the Eigenspace Instability Measure

We evaluate how predictive the eigenspace instability measure is of downstream instability, showing that the theoretically grounded eigenspace instability measure correlates strongly with downstream instability and is competitive with other embedding distance measures. To do this, we measure the Spearman correlations between the downstream prediction disagreement and the embedding distance measure for each of the five tasks and two embedding algorithms. The Spearman correlation quantifies how similar the ranking of the pairs of embeddings based on the embedding distance measure is to the ranking of the pairs of embeddings based on their downstream prediction disagreement, with a maximum value of 1.0. In Table 1, we see that the eigenspace instability measure and the k-NN measure are the top-performing embedding distance measures by Spearman correlation, with the eigenspace instability measure attaining Spearman correlations between 0.04 better and 0.09 worse than the k-NN measure on all tasks. Moreover, the strong correlation of at least 0.68 for the eigenspace instability measure across embedding algorithms and downstream tasks validates our theoretical claim that this measure relates to downstream disagreement. In Appendix C.4, we include additional plots showing the downstream prediction disagreement versus the embedding distance measures.

5.2 Embedding Distance Measures for Dimension-Precision Selection

We demonstrate that the eigenspace instability measure is an effective selection criterion for dimension-precision parameters, outperforming the majority of existing embedding distance measures and competitive with the k-NN measure, for which there are no theoretical guarantees. Specifically, we evaluate the embedding distance measures as selection criteria in two settings of increasing difficulty: in the first setting the goal is, given two pairs of embeddings (each corresponding to an arbitrary dimension-precision combination), to select the pair with the lowest downstream instability. In the second, more challenging setting, the goal is to select, among all dimension-precision combinations corresponding to the same total memory, the one with the lowest downstream instability. This setting is challenging, as for many memory budgets, there are more than two choices of embedding pairs, and some choices may have very similar expected downstream instability. We now discuss each of these settings, and the corresponding results, in more detail.

For the first, simpler setting, we first form all groupings of two embedding pairs with different dimension-precision combinations. For instance, a grouping may have one embedding pair with dimension 800, precision 32, and another embedding pair with dimension 200, precision 2, where a pair consists of a Wiki'17 and a Wiki'18 embedding from the same algorithm. For each embedding distance measure, we report the fraction of groupings where the embedding distance measure correctly chooses the embedding pair with lower downstream instability on a given task. We repeat over three seeds, comparing embedding pairs of the same seed, and report the average. In Table 2, we show that the eigenspace instability measure and k-NN measure are the most accurate embedding distance measures, with up to $3.71\times$ and $3.73\times$ lower selection error rates than the other embedding distance measures, respectively. Moreover, across downstream tasks, the eigenspace instability measure attains $0.93\times$ to $1.55\times$ the error rate of the k-NN measure.

For the second, more challenging setting, we enumerate all embedding pairs with different dimension-precision combinations *which correspond to the same total memory*. For

Table 1. Spearman correlation scores between embedding distance measures and downstream prediction disagreement across varying dimension-precision pairs for the embedding. Downstream models are trained for sentiment (SST-2, MR, Subj, MPQA) and NER (CoNNL-2003) tasks. Strongest correlation values are bolded.

Downstream Task	SST-2		MR		Subj		MPQA		CoNNL-2003	
Embedding Algorithm	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC
Eigenspace Instability	0.68	0.84	**0.86**	0.72	0.72	**0.78**	0.75	0.85	**0.80**	0.83
k-NN	**0.74**	**0.89**	0.85	**0.74**	**0.73**	0.76	**0.77**	**0.94**	0.76	**0.92**
Semantic Displacement	0.70	0.28	0.63	0.59	0.45	0.46	0.68	0.29	0.53	0.32
PIP Loss	-0.40	0.39	-0.11	0.66	-0.14	0.56	-0.38	0.42	0.01	0.44
1-Eigenspace Overlap	0.63	0.26	0.66	0.56	0.50	0.45	0.68	0.27	0.58	0.31

Table 2. Selection error when using embedding distance measures to predict the most stable embedding dimension-precision parameters on downstream tasks. Downstream models are trained for sentiment (SST-2, MR, Subj, MPQA) and NER (CoNNL-2003) tasks. Lowest errors are bolded.

Downstream Task	SST-2		MR		Subj		MPQA		CoNNL-2003	
Embedding Algorithm	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC
Eigenspace Instability	0.23	0.17	**0.14**	0.24	0.24	**0.20**	0.22	0.17	**0.20**	0.17
k-NN	**0.21**	**0.13**	0.15	**0.22**	**0.23**	0.21	**0.21**	**0.11**	0.21	**0.11**
Semantic Displacement	0.24	0.42	0.26	0.29	0.34	0.34	0.24	0.40	0.29	0.41
PIP Loss	0.64	0.35	0.52	0.25	0.57	0.28	0.64	0.33	0.50	0.32
1-Eigenspace Overlap	0.28	0.43	0.27	0.29	0.32	0.34	0.25	0.41	0.29	0.41

each embedding distance measure, we report the average absolute percentage difference between the downstream instability of the pair selected by the measure to the oracle pair, across different memory budgets. We also introduce two naive baselines that do not require an embedding distance measure: *high precision*, which selects the pair with the highest precision possible at each memory budget, and *low precision*, which selects the pair with the lowest precision possible at each memory budget. As before, we repeat over three seeds, comparing embedding pairs of the same seed, and report the average. We see that the eigenspace instability measure and k-NN measure again outperform the other baselines on the majority of downstream tasks, with the eigenspace instability measure attaining an distance up to 3.06% (absolute) closer to the oracle than the other baselines, and average distance to the oracle 0.02% (absolute) better to 0.46% (absolute) worse than the k-NN measure across downstream tasks (Table 3). For both settings, we include additional results measuring the worst-case performance of the embedding distance measure in Appendix C.5, where we find that the eigenspace instability measure and k-NN measure continue to be the top-performing measures.

6 EXTENSIONS

We demonstrate that the stability-memory tradeoffs we observe with pre-trained word embeddings can extend to knowledge graph embeddings and contextual word embeddings: as the memory of the embedding increases, the instability decreases. We first show how these trends hold on knowledge graph embeddings in Section 6.1 and then on contextual word embeddings in Section 6.2.

6.1 Knowledge Graph Embeddings

Knowledge graph embeddings (KGEs) are a popular type of embedding that is used for multi-relational data, such as social networks, knowledge bases, and recommender systems. Here, we show that as the dimension and precision of the KGE increases, the stability on two standard KGE tasks improves, aligning with the trends we observed on pre-trained word embedding algorithms. Unlike word embedding algorithms, the input to KGE algorithms is a directed graph, where the relations are the edges in the graph and the entities are the nodes in the graph. The graph can be represented as a set of triplets (h, r, t), where the entity head h is related by the relation r to the entity tail t. The output is two set of embeddings: (1) entity embeddings $(\mathbf{e}_h)$ and (2) relation embeddings $(\mathbf{r}_r)$. We study the stability of these embeddings on two standard benchmark tasks: link prediction and triplet classification. We summarize the datasets and training and evaluation protocols, and then discuss the results.

Table 3. Average difference (absolute percentage) to the oracle downstream instability when using embedding distance measures as the selection criteria for dimension and precision parameters under different memory budgets. Top-performing values are bolded.

Downstream Task	SST-2		MR		Subj		MPQA		CoNNL-2003	
Embedding Algorithm	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC
Eigenspace Instability	0.65	1.42	**0.69**	1.03	0.39	0.63	0.34	0.44	0.28	0.43
k-NN	0.57	**1.07**	0.71	**0.57**	0.38	**0.57**	**0.32**	**0.33**	0.32	**0.23**
Semantic Displacement	**0.37**	3.73	0.72	2.02	0.48	0.94	0.56	1.33	0.27	1.17
PIP Loss	3.63	3.32	3.33	1.96	1.16	0.74	3.24	1.05	0.83	0.99
1-Eigenspace Overlap	0.88	3.60	0.84	2.02	**0.34**	0.93	0.51	1.25	**0.20**	1.15
High Precision	0.85	3.94	1.55	2.07	0.61	1.01	0.40	1.49	0.60	1.28
Low Precision	3.63	1.23	3.33	4.09	1.16	1.39	3.24	0.66	0.83	0.74

Datasets We use two datasets to train KGE embeddings: FB15K-95 and FB15K. FB15K was introduced in Bordes et al. (2013) and is composed of a subset of triplets from the Freebase knowledge base. We construct FB15K-95 by randomly sampling 95% of the the triplets from the training dataset of FB15K. The validation and test datasets remain the same for both datasets. We use these datasets to study the stability of KGEs under small changes in training data.

Training Protocol We consider a standard KGE algorithm—TransE (Bordes et al., 2013). The TransE objective function minimizes the distances $d(\mathbf{e}_h + \mathbf{r}_r, \mathbf{e}_t)$ for observed triplets and maximizes the distances for negatively sampled triplets, where either h or t has been corrupted. We use the L_1 distance for the distance function d, and learn the embeddings iteratively via stochastic gradient descent.

To measure the impact of the dimension and precision on the stability of TransE embeddings, we train TransE embeddings of dimensions $\{10, 20, 50, 100, 200, 400\}$ and then uniformly quantize the entity and relation embeddings for each TransE embedding to bits $\{1, 2, 4, 8, 16, 32\}$ per entry in embedding.[9] We perform a hyperparameter sweep on the learning using dimension 50, and select the best learning rate on the validation set for link prediction. We use this learning rate for all dimensions to minimize the impact of learning rate on our analysis. We take other training hyperparameters from the TransE paper (Bordes et al., 2013) for the FB15K dataset, and use three seeds to train each dimension using the OpenKE repository (Han et al., 2018).

Evaluation Protocol For each dimension-precision, we evaluate all pairs of embeddings trained on FB15K-95 and FB15K on the link prediction and triplet classification tasks. For each test triplet, the link prediction task evaluates the mean predicted rank of an observed triplet among all corrupted triplets. We measure instability on this task with unstable-rank@10: the fraction of changes in rank greater than 10 between two embeddings across all test triplets.

[9] The same dimension is used for both the entity and the relation embeddings.

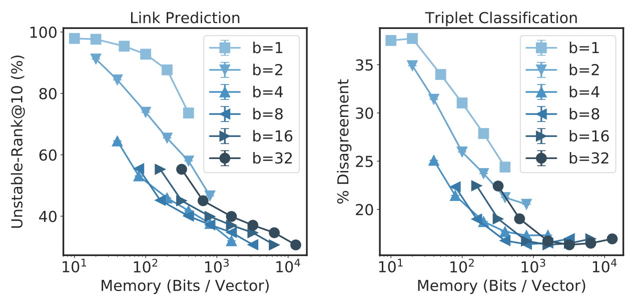


Figure 3. Stability of link prediction (left) and triplet classification (right) when evaluating embeddings trained on 95% of FB15K training triplets and all of FB15K.

The triplet classification task was introduced in Socher et al. (2013a) and is a binary classification task to determine whether or not a triplet occurs in the knowledge graph. For each relation, a threshold T_R is determined based on the validation set, such that if $d(\mathbf{e}_h + \mathbf{r}_r, \mathbf{e}_t) \leq T_R$ then the triplet is predicted as positive. For each dimension-precision pair, we set the thresholds on FB15K-95 embedding and use the same thresholds for the FB15K embedding. We include results with threshold set independently for each embedding in Appendix C.6. As for classification with downstream NLP tasks, we define stability on the triplet classification task as the percentage prediction disagreement.

Results We find that the stability-memory tradeoffs continue to hold for TransE embeddings on the link prediction and triplet classification tasks: overall as the memory increases, the instability decreases, and specifically, as the dimension and precision increases, the instability decreases. In Figure 3 (left), we show for link prediction that as the memory per vector increases, the unstable-rank@10 measure decreases. Each line represents a different precision, where each point on the line represents a different dimension. Thus, we can also see that as the dimension increases, the unstable-rank@10 decreases, and as precision increases, this measure also decreases. When fitting a linear-log model to the dimension-precision combinations for all memory budgets, we find that increasing the memory $2\times$ decreases the instability by 7% to 19% (relative). In Figure 3 (right), we similarly show for triplet classification that as the memory per vector increases, the prediction disagreement between the embeddings trained on the two datasets decreases. Finally, as we saw with word embeddings, we observe that the effect of the dimension or precision on stability is more significant at low memory regimes.

6.2 Contextual Word Embeddings

Unlike pre-trained word embeddings, contextual word embeddings (Peters et al., 2018; Vaswani et al., 2017) extract word representations dynamically with awareness of the input context. We find that the stability-memory trade-off observed on pre-trained embeddings can still hold for contextual word embeddings, though with noisier trends: higher dimensionality and higher precision can demonstrate better downstream stability. We pre-train shallow, 3-layer versions of BERT (Devlin et al., 2019) on sub-sampled Wiki'17 and Wiki'18 dumps ($\sim$200 million tokens) as feature extractors with different transformer layer output dimensionalities, ranging from a quarter as large to $4\times$ as large as the hidden size in $\text{BERT}_{\text{BASE}}$ (i.e., 768).[10] To evaluate the effect of precision, we use uniform quantization to compress the output of the last transformer layer in the BERT models. Finally, we measure the prediction disagreement between linear classifiers trained on top of the Wiki'17 and Wiki'18 BERT models, with the BERT model parameters fixed.

Across four sentiment analysis tasks, we can observe reduced instability with higher dimensional BERT embeddings (Figure 12a in Appendix C.7); however, the reduction in instability from increasing the dimension is noisier than with pre-trained word embeddings. We hypothesize this is due to the instability of the training of the BERT embedding itself, which is a much more complex model than pre-trained word embeddings. We also observe that increasing the precision can decrease the downstream instability, such that using 1 or 2 bits for precision often demonstrates observable degradation in stability, but precisions higher than 4-bit have negligible influence on stability (Figure 12b in Appendix C.7). For more details on the training and evaluation, see Appendix C.7.

7 Related Work

There have been many recent works studying word embedding instability (Hellrich & Hahn, 2016; Antoniak & Mimno, 2018; Wendlandt et al., 2018; Pierrejean & Tanguy, 2018; Chugh et al., 2018; Hellrich et al., 2019); these works have focused on the *intrinsic* instability of word embeddings, meaning the stability measured between the embedding matrices without training a downstream model. In the work of Wendlandt et al. (2018) they do consider a downstream task (part-of-speech tagging), but focus on how the intrinsic instability impacts the *error* of words on this task. In contrast, we focus on the downstream instability (i.e., prediction disagreement), evaluating how different parameters of embeddings impact downstream instability with large-scale Wikipedia embeddings over multiple downstream NLP tasks. Furthermore, we provide theoretical analysis which is specific to the downstream instability setting to help explain our empirical observations.

More broadly, researchers have also studied the general problem of ML model instability in the context of online training and incremental learning. Fard et al. (2016) study the problem of reducing the prediction churn between consecutively trained classifiers by introducing a Monte Carlo stabilization operator as a form of regularization. Cotter et al. (2016) further define stability as a design goal for classifiers in real-world applications, along with goals such as precision, recall, and fairness, and propose an algorithm to optimize for these multiple design goals. Other researchers have also studied the problem of catastrophic forgetting when models are incrementally trained (Yang et al., 2019), which shares a similar goal of wanting to learn new information, while minimizing changes with respect to previous models. As these works focus on changes to the downstream model training to reduce instability, we believe these works are complementary to our work, which focuses on better understanding the instability introduced by word embeddings.

8 Conclusion

We performed the first in-depth study of the downstream instability of word embeddings. In our study, we exposed a novel stability-memory tradeoff, showing that increasing the embedding dimension or precision decreases downstream instability. To better understand these empirical results, we introduced a new measure for embedding instability—the eigenspace instability measure—which we theoretically relate to downstream prediction disagreement. We showed that this theoretically grounded embedding measure correlates strongly with downstream instability, and can be used to select dimension-precision parameters, performing competitively with other embedding measures on minimizing downstream instability without training the downstream tasks. Finally, we demonstrated that the stability-memory tradeoff extends to other types of embeddings, including contextual word embeddings and knowledge graph embeddings. We hope our study motivates future work on the instability of ML models in even more complex pipelines.

[10] The recent 12-layer $\text{BERT}_{\text{BASE}}$ model is pre-trained with 3 billion tokens from BooksCorpus (Zhu et al., 2015) and Wikipedia, and requires 16 TPU chips to train for 4 days.

REFERENCES

Akbik, A., Blythe, D., and Vollgraf, R. Contextual string embeddings for sequence labeling. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 1638–1649, 2018.

Andrews, M. Compressing word embeddings. In *International Conference on Neural Information Processing (ICONIP)*, pp. 413–422, 2016.

Antoniak, M. and Mimno, D. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics (TACL)*, 6:107–119, 2018.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146, 2017.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2787–2795, 2013.

Bullinaria, J. A. and Levy, J. P. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526, 2007.

Chugh, M., Whigham, P. A., and Dick, G. Stability of word embeddings using word2vec. In *AI 2018: Advances in Artificial Intelligence*, pp. 812–818, 2018.

Cotter, A., Friedlander, M. P., Goh, G., and Gupta, M. R. Satisfying real-world goals with dataset constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2415–2423, 2016.

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, pp. 191–198, 2016.

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 (NAACL-HLT)*, pp. 4171–4186, 2019.

Fard, M. M., Cormier, Q., Canini, K. R., and Gupta, M. R. Launch and iterate: Reducing prediction churn. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3179–3187, 2016.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pp. 1–6, 2018.

Gordon, J. Introducing tensorflow hub: A library for reusable machine learning modules in tensorflow, 2018. URL https://medium.com/tensorflow/introducing-tensorflow-hub-a-library-for-reusable-machine-learning-modules-in-tensorflow-cdee41fa18f9.

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 (ACL)*, pp. 1489–1501, 2016.

Han, X., Cao, S., Xin, L., Lin, Y., Liu, Z., Sun, M., and Li, J. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.

He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., and Candela, J. Q. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (ADKDD)*, pp. 5:1–5:9, 2014.

Hellrich, J. and Hahn, U. Bad company–neighborhoods in neural embedding spaces considered harmful. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 2785–2796, 2016.

Hellrich, J., Kampe, B., and Hahn, U. The influence of down-sampling strategies on SVD word embedding stability. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pp. 18–26, 2019.

Hermann, J. and Balso, M. D. Meet michelangelo: Uber's machine learning platform, 2017. URL https://eng.uber.com/michelangelo/.

Jin, C., Kakade, S. M., and Netrapalli, P. Provable efficient online matrix completion via non-convex stochastic gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4527–4535, 2016.

Kim, Y. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

May, A., Zhang, J., Dao, T., and Ré, C. On the downstream performance of compressed word embeddings. *arXiv preprint arXiv:1909.01264*, 2019.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3111–3119, 2013b.

Pang, B. and Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *"Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)"*, pp. 271–278, 2004.

Pang, B. and Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 115–124, 2005.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

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 (EMNLP-HLT)*, pp. 2227–2237, 2018.

Pierrejean, B. and Tanguy, L. Predicting word embeddings variability. In *The Seventh Joint Conference on Lexical and Computational Semantics (*SEM)*, pp. 154–159, 2018.

Schönemann, P. H. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

Sell, T. and Pienaar, W. Introducing Feast: an open source feature store for machine learning, 2019. URL https://cloud.google.com/blog/products/ai-machine-learning/introducing-feast-an-open-source-feature-store-for-machine-learning.

Shiebler, D., Green, C., Belli, L., and Tayal, A. Embeddings@Twitter, 2018. URL https://blog.twitter.com/engineering/en_us/topics/insights/2018/embeddingsattwitter.html.

Shu, R. and Nakayama, H. Compressing word embeddings via deep compositional code learning. In *International Conference on Learning Representation (ICLR)*, 2018.

Socher, R., Chen, D., Manning, C. D., and Ng, A. Y. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 926–934, 2013a.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013b.

Tjong Kim Sang, E. F. and De Meulder, F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

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 (NeurIPS)*, pp. 5998–6008, 2017.

Wendlandt, L., Kummerfeld, J., 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 (NAACL-HLT)*, pp. 2092–2102, 2018.

Wiebe, J., Wilson, T. S., and Cardie, C. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210, 2005.

Yang, Y., Zhou, D.-W., Zhan, D.-C., Xiong, H., and Jiang, Y. Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 74–82, 2019.

Yin, Z. and Shen, Y. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 887–898, 2018.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A EIGENSPACE INSTABILITY: THEORY

We present the proof of Proposition 1, which shows that the expected prediction disagreement between the linear regression models trained on embedding matrices X and $\tilde{X}$ is equal to the eigenspace instability measure between X and $\tilde{X}$.

Proposition 1. *Let $X \in \mathbb{R}^{n\times d}$, $\tilde{X} \in \mathbb{R}^{n\times k}$ be two full-rank embedding matrices, where x_i and $\tilde{x}_i$ correspond to the i^{th} rows of X and $\tilde{X}$ respectively. Let $y \in \mathbb{R}^n$ be a random regression label vector with zero mean and covariance $\Sigma \in \mathbb{R}^{n\times n}$. Then the (normalized) expected disagreement between the linear models f_y and $\tilde{f}_y$[11] trained on label vector y using embedding matrices X and $\tilde{X}$ respectively satisfies*

$$\frac{\mathbb{E}_y\left[\sum_{i=1}^n (f_y(x_i) - \tilde{f}_y(\tilde{x}_i))^2\right]}{\mathbb{E}_y\left[\|y\|^2\right]} = \mathcal{EI}_\Sigma(X, \tilde{X}). \tag{2}$$

Proof. Let $X = USV^T \in \mathbb{R}^{n\times d}$ and $\tilde{X} = \tilde{U}\tilde{S}\tilde{V}^T \in \mathbb{R}^{n\times d}$ be the SVDs of X and $\tilde{X}$ respectively, and let x_i and $\tilde{x}_i$ in $\mathbb{R}^d$ be the i^{th} rows of X and $\tilde{X}$. Recall that parameter vector $w \in \mathbb{R}^d$ which minimizes $\|Xw - y\|_2^2$ is given by $w^* = (X^TX)^{-1}X^Ty$ (where here we use the assumption that X is full-rank to know that X^TX is invertible). Thus, the linear regression model $f_y(x) = x^Tw^*$ trained on data matrix X with label vector $y \in \mathbb{R}^n$ makes predictions $Xw^* = X(X^TX)^{-1}X^Ty = USV^T(VS^{-2}V^T)VSU^Ty = UU^Ty \in \mathbb{R}^n$ on the n training points. So if we train linear model with data matrices X and $\tilde{X}$, using the same label vector y, these model will make predictions UU^Ty and $\tilde{U}\tilde{U}^Ty$ on the n training points, respectively. Thus, the expected disagreement between the predictions made using X vs. $\tilde{X}$, over the randomness in y, can be expressed as follows:

$$\begin{aligned}
\mathbb{E}_y\left[\sum_{i=1}^n (f_y(x_i) - \tilde{f}_y(\tilde{x}_i))^2\right] &= \mathbb{E}_y\left[\|UU^Ty - \tilde{U}\tilde{U}^Ty\|^2\right] \\
&= \mathbb{E}_y\left[\left(UU^Ty - \tilde{U}\tilde{U}^Ty\right)^T\left(UU^Ty - \tilde{U}\tilde{U}^Ty\right)\right] \\
&= \mathbb{E}_y\left[y^TUU^TUU^Ty + y^T\tilde{U}\tilde{U}^T\tilde{U}\tilde{U}^Ty - 2y^T\tilde{U}\tilde{U}^TUU^Ty\right] \\
&= \mathbb{E}_y\left[y^T\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^TUU^T\right)y\right] \\
&= \mathbb{E}_y\left[\mathrm{tr}\left(y^T\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^TUU^T\right)y\right)\right] \\
&= \mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^TUU^T\right)\mathbb{E}_y\left[yy^T\right]\right) \\
&= \mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^TUU^T\right)\Sigma\right), \quad \text{where } \Sigma = \mathbb{E}_y\left[yy^T\right]
\end{aligned}$$

Furthermore, we can easily compute the expected norm of the label vector y.

$$\begin{aligned}
\mathbb{E}_y\left[\|y\|^2\right] &= \mathbb{E}_y\left[\mathrm{tr}(y^Ty)\right] \\
&= \mathbb{E}_y\left[\mathrm{tr}(yy^T)\right] \\
&= \mathrm{tr}(\mathbb{E}_y\left[yy^T\right]) \\
&= \mathrm{tr}(\Sigma).
\end{aligned}$$

Thus, we have successfully shown that

$$\begin{aligned}
\frac{\mathbb{E}_y\left[\sum_{i=1}^n (f_y(x_i) - \tilde{f}_y(\tilde{x}_i))^2\right]}{\mathbb{E}_y\left[\|y\|^2\right]} &= \frac{\mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^TUU^T\right)\Sigma\right)}{\mathrm{tr}(\Sigma)} \\
&=: \mathcal{EI}_\Sigma(X, \tilde{X}),
\end{aligned}$$

as desired. $\square$

[11] $f_y(x) = w^Tx$, for $w = (X^TX)^{-1}X^Ty$, and $\tilde{f}_y(\tilde{x}) = \tilde{w}^T\tilde{x}$, for $\tilde{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^Ty$.

A.1 Efficiently Computing the Eigenspace Instability Measure

We now discuss an efficient way of computing the eigenspace instability measure, assuming $\Sigma = (EE^T)^\alpha + (\tilde{E}\tilde{E}^T)^\alpha = VR^{2\alpha}V^T + \tilde{V}\tilde{R}^{2\alpha}\tilde{V}^T$ as discussed in Section 4.1. Here, E and $\tilde{E}$ correspond to fixed embedding matrices,[12] where $E = VRW^T$ and $\tilde{E} = \tilde{V}\tilde{R}\tilde{W}^T$ are the SVDs of E and $\tilde{E}$ respectively.

Recall the definition of the eigenspace instability measure:

$$\mathcal{EI}_\Sigma(X, \tilde{X}) := \frac{1}{\mathrm{tr}(\Sigma)} \mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^T UU^T\right)\Sigma\right).$$

We now show that both traces in this expression can be computed efficiently.

$$\begin{aligned}
\mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^T UU^T\right)\Sigma\right) &= \mathrm{tr}\left(\left(UU^T + \tilde{U}\tilde{U}^T - 2\tilde{U}\tilde{U}^T UU^T\right)\left(VR^{2\alpha}V^T + \tilde{V}\tilde{R}^{2\alpha}\tilde{V}^T\right)\right) \\
&= \mathrm{tr}\left(R^\alpha V^T UU^T VR^\alpha\right) + \mathrm{tr}\left(R^\alpha V^T \tilde{U}\tilde{U}^T VR^\alpha\right) - 2\,\mathrm{tr}\left(R^\alpha V^T \tilde{U}\tilde{U}^T UU^T VR^\alpha\right) + \\
&\quad \mathrm{tr}\left(\tilde{R}^\alpha \tilde{V}^T UU^T \tilde{V}\tilde{R}^\alpha\right) + \mathrm{tr}\left(\tilde{R}^\alpha \tilde{V}^T \tilde{U}\tilde{U}^T \tilde{V}\tilde{R}^\alpha\right) - 2\,\mathrm{tr}\left(\tilde{R}^\alpha \tilde{V}^T \tilde{U}\tilde{U}^T UU^T \tilde{V}\tilde{R}^\alpha\right) \\
&= \|U^T VR^\alpha\|_F^2 + \|\tilde{U}^T VR^\alpha\|_F^2 - 2\,\mathrm{tr}\left(R^\alpha (V^T\tilde{U})(\tilde{U}^T U)(U^T V)R^\alpha\right) + \\
&\quad \|U^T \tilde{V}\tilde{R}^\alpha\|_F^2 + \|\tilde{U}^T \tilde{V}\tilde{R}^\alpha\|_F^2 - 2\,\mathrm{tr}\left(\tilde{R}^\alpha (\tilde{V}^T\tilde{U})(\tilde{U}^T U)(U^T \tilde{V})\tilde{R}^\alpha\right). \qquad (3)
\end{aligned}$$

$$\begin{aligned}
\mathrm{tr}\left(\Sigma\right) &= \mathrm{tr}\left(VR^{2\alpha}V^T + \tilde{V}\tilde{R}^{2\alpha}\tilde{V}^T\right) \\
&= \mathrm{tr}\left(V^T VR^{2\alpha}\right) + \mathrm{tr}\left(\tilde{V}^T\tilde{V}\tilde{R}^{2\alpha}\right) \\
&= \mathrm{tr}\left(R^{2\alpha}\right) + \mathrm{tr}\left(\tilde{R}^{2\alpha}\right). \qquad (4)
\end{aligned}$$

We now note that the traces in Equation (4), and all the matrix multiplications in Equation (3) can be computed efficiently and with low-memory (no need to ever store an n by n Gram matrix, for example), assuming the embedding matrices are "tall and thin" (large vocabulary, relatively low-dimensional). More specifically, the eigenspace instability measure can be computed in time $O(nd^2)$ and memory $O(d^2)$, where we take $X, \tilde{X}, E, \tilde{E}$ to all be in $\mathbb{R}^{n\times d}$ (or in $\mathbb{R}^{n\times d'}$ for $d' \leq d$). Thus, even for large vocabulary n, the eigenspace instability measure can be computed relatively efficiently (assuming the dimension d isn't too large).

B EXPERIMENTAL SETUP DETAILS

We discuss the experimental protocols used for each of our experiments. In Appendix B.1, we discuss the training procedures for the word embeddings, and in Appendix B.2, we discuss how we compress and post-process the embeddings. In Appendix B.3, we describe the models, datasets, and training procedures used for the downstream tasks in our study, and in Appendix B.4, we discuss how we analyze the instability trends we observe on these tasks. Finally, in Appendix B.5 and Appendix B.6 we describe setup details for the extension experiments on knowledge graph and contextual word embeddings, respectively.

B.1 Word Embedding Training

We use Google's C implementation of word2vec CBOW[13] and our own C++ implementation of MC to train word embeddings. For CBOW, we use the default learning rate, and for MC, since we are using our own implementation, we use a learning rate which we found to achieve low loss on Wiki'17. We include the full details on the hyperparameters used for both embedding algorithms in Table 4.

B.2 Word Embedding Compression and Post-Processing

We now discuss some important implementation details for uniform quantization related to stability. To minimize confounding factors with stability, we use deterministic rounding for each word. The bounds of the interval for uniform quantization

[12] In our experiments, E and $\tilde{E}$ are the highest-dimensional ($d = 800$), full-precision embeddings for Wiki'17 and Wiki'18, respectively.
[13] https://github.com/tmikolov/word2vec

Table 4. Hyperparameters for embedding algorithms.

Algorithm	Hyperparameter	Value
Shared	Training epochs	50
	Window size	15
	Minimum count	5
	Threads	56
CBOW	Learning rate	0.05
	Negative samples	5
MC	Learning rate	0.2
	LR decay epochs	20
	Batch size	128
	Stopping tolerance	0.0001

are determined by computing an optimal clipping threshold which is based on the distribution of the real numbers to be quantized. As we assume that embeddings X and $\tilde{X}$ have similar distributions in terms of their vector values, we use the same clipping threshold across embeddings X and $\tilde{X}$ to avoid unnecessary sources of instability, and we compute the clipping threshold using embedding X. Finally, we apply orthogonal Procrustes to align embedding $\tilde{X}$ to embedding X before compressing the embeddings and training downstream models. Preliminary results indicated that this alignment decreased instability, particularly at high compression rates, and we use this technique throughout our experiments.

B.3 Downstream Tasks

We discuss the models, datasets, and training procedure we use for the sentiment analysis and NER tasks.

B.3.1 Sentiment Analysis

We use a simple, bag-of-words model for sentiment analysis. The goal of the task is to classify a sentence as positive or negative. For each sentence, the bag-of-words model averages the word embeddings of the words in the sentence and then passes the sentence embedding through a linear classifier. This simple model allows us to study the impact of the embedding on the downstream task in a controlled setting, where the downstream model itself is expected to be fairly stable.

We use four datasets for the sentiment analysis task: SST-2, MR, Subj, and MPQA. These are the four largest binary classification datasets used in Kim (2014).[14] We use their given train/validation/test splits for SST-2. For MR, Subj, and MPQA, which do not have these splits, we take 10% of the data for the validation set, 10% for the test set, and use the remaining 80% for the training set.

We tune the learning rate for each dataset and embedding algorithm. We use the 400-dimensional Wiki'17 embeddings to tune the learning rate in the grid of {1e-6, 1e-5, 0.0001, 0.001, 0.01, 0.1, 1}. We choose the learning rate which achieves the highest validation accuracy on average across three seeds for each dataset and report the selected values in Table 5. To avoid choosing unstable learning rates, we also throw out learning rate values where the validation errors increase by 15% or greater between any consecutive epochs, though this only affects the MC MPQA learning rate. We include the hyperparameters shared among all datasets in Table 6.

Table 5. Selected learning rates for the sentiment analysis datasets per embedding algorithm.

Algorithm	SST-2	MR	Subj	MPQA
CBOW	0.0001	0.001	0.0001	0.001
MC	0.001	0.1	0.1	0.001

[14] https://github.com/harvardnlp/sent-conv-torch/tree/master/data

Table 6. Training hyperparameters shared across embedding algorithms for the sentiment analysis task.

Hyperparameter	Value
Optimizer	Adam
Batch size	32
Training epochs	100

B.3.2 *Named Entity Recognition*

We use the single-layer, BiLSTM model from Akbik et al. (2018) for named entity recognition.[15] We turn off the conditional random field (CRF) for computational efficiency and include a smaller subset of results with the CRF turned on in Appendix D.1.

We use the standard English CoNLL-2003 dataset with the default setup for train/development/test splits (Tjong Kim Sang & De Meulder, 2003). Following Gardner et al. (2018), we ignore article divisions (denoted with "-DOCSTART-") and do not consider them as sentences.[16]

We tune the learning rate per embedding algorithm, and otherwise follow the training hyperparameter settings of Akbik et al. (2018). Using the 400-dimensional Wiki'17 embeddings, we sweep the learning rate in the grid of $\{0.001, 0.01, 0.1, 1, 10\}$, and choose the one which achieves the highest validation micro F1-score on average across three seeds for each embedding algorithm. We train with vanilla SGD without momentum and use learning decay with early stopping if the learning rate becomes too small. We provide the selected learning rates in Table 7 and the hyperparameters shared across embeddings in Table 8.

Table 7. Selected learning rates for the NER task per embedding algorithm.

CBOW	MC
0.1	1.0

Table 8. Training hyperparameters shared across embedding algorithms for the NER task.

Hyperparameter	Value
Optimizer	SGD
Batch size	32
Max. training epochs	150
LSTM hidden size	256
LSTM num. layers	1
Patience	3
Anneal factor	0.5
Word dropout	0.05
Locked dropout	0.5

B.4 Fitting Linear-Log Models to Trends

We describe in detail how we fit linear-log model to the memory, dimension, and precision trends in Section 3.3. To propose the simple rule of thumb relating stability and memory, we consider 10 tasks to form a data matrix for the linear-log model: 5 downstream tasks (the four sentiment tasks in our study and the NER task) for two embedding algorithms (CBOW and MC embeddings). Let P denote the number of Wiki'17/Wiki'18 pairs of embedding matrices from our experiments which correspond to a combination of dimension d, precision b, and random seed s (we consider 3 random seeds) such that

[15] https://github.com/zalandoresearch/flair
[16] https://github.com/allenai/allennlp

the number of bits per row is less than our cutoff of 10^3 ($bd < 10^3$).[17] For each task t (out of $T = 10$ total tasks), we construct a data matrix $X^{(t)} \in \mathbb{R}^{P\times(T+1)}$, and a label vector $y^{(t)} \in \mathbb{R}^P$, as follows: Each row in $X^{(t)}$ corresponds to one of the above P pairs of Wiki'17/Wiki'18 embedding matrices. For each of these embedding matrix pairs, we compute the memory m' in bits occupied per row of the embedding matrices, as well as the downstream prediction disagreement percentage $y' \in [0, 100]$ between the models trained on those embeddings. We then set the corresponding row in $X^{(t)}$ to be $[log_2(m'), e_t] \in \mathbb{R}^{T+1}$, where $e_t \in \mathbb{R}^T$ is a binary vector with a one at index t and zeros everywhere else, and the corresponding entry of $y^{(t)}$ to the prediction disagreement y'; note that appending e_t to $\log_2(m')$ allows us to learn a different bias term (i.e., y-intercept) per task. We then vertically concatenate all the $X^{(t)}$ matrices and label vectors $y^{(t)}$, to form a single data matrix $X \in \mathbb{R}^{TP\times(T+1)}$ and label vector $y \in \mathbb{R}^{TP}$. To fit our log-linear model, we use X and y to solve the least squares problem using the closed form solution, $\hat{\beta} = (X^TX)^{-1}X^Ty$. Given $\hat{\beta} \in \mathbb{R}^{T+1}$, for each task t we can extract the fitted log-linear trend: $\mathcal{DI}_t \approx \hat{\beta}_t - \hat{\beta}_0 * log_2(m)$, where $\hat{\beta}_0 \approx 1.4$ is the first element of $\hat{\beta}$, and $\hat{\beta}_t$ is the $(t+1)^{th}$ element of $\hat{\beta}$. This implies that doubling the memory of the embeddings on average leads to a 1.4% reduction in downstream prediction disagreement.

To fit the individual dimension and precision log-linear trends, we follow a protocol very similar to the above. For the dimension (respectively, precision) trend, the primary difference with the above protocol is that instead of having an independent y-intercept term per task, we have an independent y-intercept term for each combination of task and precision (resp., dimension). Furthermore, in the rows of the data matrices, instead of $\log_2(\cdot)$ of the memory m, we consider $\log_2(\cdot)$ of the dimension d (resp., precision b).

We also use the linear-log model for stability-memory to compute the minimum and maximum relative percentage decreases in downstream instability when increasing the memory of word embeddings. In particular, our goal is to understand how much the 1.4% decrease in prediction disagreement is in relative terms. To do this, we consider the combination of downstream task and embedding algorithm which is most stable at high memory (task: Subj; embedding algorithm: CBOW), and the combination which is least stable at low memory (task: MR; embedding algorithm: MC). At these extreme points, the instability is approximately 2.2% and 26%, respectively. A 1.4% absolute decrease in instability from 3.6% to 2.2% corresponds to a relative decrease of approximately 39% $\left(\frac{1.4}{3.6} \approx 0.39\right)$. Similarly, a 1.4% absolute decrease in instability from 25.9% to 24.5% corresponds to a relative decrease of approximately 5% $\left(\frac{1.4}{25.9} \approx 0.05\right)$. Thus, we conclude that this 1.4% absolute decrease in instability corresponds to a relative decrease in instability between 5% and 39%, across the tasks and embedding algorithms we consider.

We repeat the procedures described in this section to fit a linear-log model to the stability-memory trend for knowledge graph embeddings in Section 6.1.

B.5 Knowledge Graph Embeddings

We use the OpenKE repository to generate knowledge graph embeddings (Han et al., 2018).[18] We follow the training hyperparameters described in Bordes et al. (2013) for TransE embeddings for the FB15K dataset where available, and use default parameters from the OpenKE repository, otherwise. We modify the repository to follow the early stopping procedure and normalization of entity embeddings to follow the protocol of Bordes et al. (2013). We additionally sweep the learning rate in {1e-5, 0.0001, 0.001, 0.01, 0.1} using dimension 50 on the FB15K-95 dataset, and choose the learning rate which attains the lowest mean rank (i.e., highest quality) on the validation set for the link prediction task. We include the full hyperparameters in Table 9. We also note that unlike with word embeddings, we do not align embeddings with orthogonal Procrustes before compressing the embeddings with uniform quantization. We found alignment to result in a quality drop on knowledge graph embeddings, likely due to the fact that there are two sets of embeddings jointly learned (relation and entity embeddings) which require special alignment techniques.

B.6 Contextual Word Embeddings

To study the downstream task stability of contextual word embeddings, we pretrain BERT Devlin et al. (2019) models and then use them as fixed feature extractors to train downstream task models. We use BERT model without fine-tuning their parameters for downstream tasks because our goal is to isolate and study the stability resulting from the difference in

[17] In our case $P = 63$, because we have 3 random seeds, and 21 pairs of dimension $d \in \{25, 50, 100, 200, 400, 800\}$ and precision $b \in \{1, 2, 4, 8, 16, 32\}$ such that $db < 10^3$.

[18] https://github.com/thunlp/OpenKE/tree/OpenKE-PyTorch

Table 9. Hyperparameters for training TransE knowledge graph embeddings. Bolded values indicate we performed a grid search. Other values are from Bordes et al. (2013) and Han et al. (2018).

Hyperparameter	Value
Optimizer	SGD
Max. training epochs	1000
Num. batches	100
Threads	8
Early stopping patience	10
Head/tail replacement strategy	Uniform
Entity negative rate	1
Relation negative rate	0
Margin γ	1
Distance d	L_1
Learning rate	**0.001**

pretraining corpora; this is in analogy to our study in Section 3 on the stability of conventional fixed pre-trained embeddings.

Pretraining In the pretraining phase, we use Wikipedia dumps (the major component of the corpus used by Devlin et al. (2019)) to train the BERT models. We use Wiki'2017 and Wiki'2018 dumps respectively for pretraining to study the stability introduced by these two corpora. We pretrain BERT models with 3 transformer layers on 10% subsampled articles from the Wikipedia dumps, which consists of approximately 200 million tokens. We use these shallower BERT model on the subsampled pretraining corpus to allow for computationally feasible training of BERT models with different transformer output dimensionality. As our corpus size are different from the one used by the original BERT model (Devlin et al., 2019), we first grid search the pretraining learning rate with the subsampled Wiki'17 corpus using the same transformer output dimensionality as the $\text{BERT}_{\text{BASE}}$. We then use the grid-searched optimal learning rate to pretrain the BERT model with different transformer dimensionality for both Wiki'17 and Wiki'18 corpus.[19]

Downstream Evaluation To evaluate the downstream stability of pre-trained BERT models, we take BERT model pairs with the same model configuration but trained on Wiki'17 and Wiki'18 respectively. We measure the percentage of disagreement in downstream task prediction of the BERT pairs as proxy for downstream stability. Specifically, we evaluate the stability on the sentiment analysis task using the SST, Subj, MR and MPQA datasets. In these tasks, we use linear bag-of-words models on top of the last transformer layer output; this output acts as the contextual word vector representation. To train the sentiment analysis task models, we first grid-search the learning rate using BERT with 768-dimensional transformer output for each dataset and choose the value with the highest validation accuracy.[20] We then use the grid-searched learning rate to train the sentiment analysis models using different pre-trained BERT models. To ensure statistically meaningful results, we use three random seeds to pretrain BERT models and train the downstream sentiment analysis models. We otherwise use the same hyperparameters reported in Table 6.

C Extended Empirical Results

We now present additional experimental results to further validate the claims in this paper and provide deeper analysis of our results. We organize this section as follows:

- In Appendix C.1, we present additional results showing that the stability-memory trends (and individual dimension and precision trends) hold on more sentiment analysis tasks.
- In Appendix C.2, we evaluate another important property–quality—exploring the tradeoffs of quality with memory and stability for the tasks in our study.

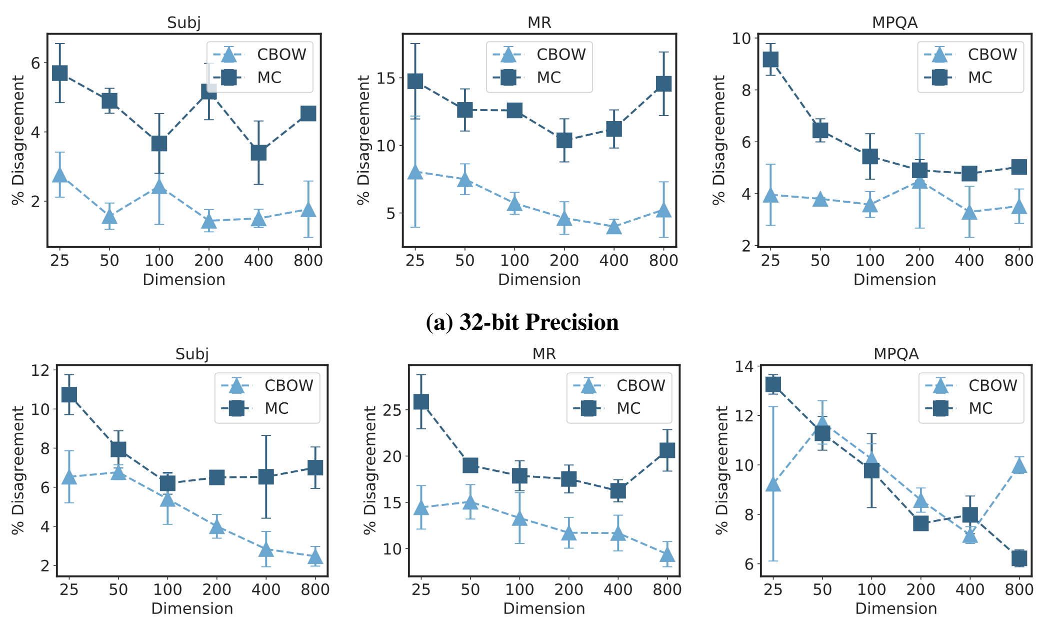
[19] We follow the experiment design from pre-trained word embeddings to use the same learning rate for pretraining BERT models with transformer configurations.

[20] We use the dimensionality used for original $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019).

- In Appendix C.3, we discuss how we choose the single hyperparameter required for both the k-NN measure and the eigenspace instability measure.
- In Appendix C.4, we use visualizations to further analyze the relationship between the downstream instability and the embedding distance measures and validate that eigenspace instability measure can help us explain the observed stability-memory trends.
- In Appendix C.5, we evaluate the worst-case performance of the embedding distance measures as selection criteria, showing that the eigenspace instability measure and k-NN measure remain the top-performing measures overall.
- In Appendix C.6, we experiment with a modified setup for the triplet classification task, showing that the trends continue to hold, but the instability plateaus faster under this modification.
- In Appendix C.7, we include the figures for the contextual word embedding results presented in Section 6.2.

C.1 Stability-Memory Tradeoff

We validate that the stability-memory tradeoff holds on three more sentiment tasks (Subj, MR, and MPQA) for dimension and precision, first in isolation and then together. As always, we train embeddings and downstream models over three seeds, and the error bars indicate the standard deviation over these seeds. In Figure 4, we can see more evidence that as the dimension increases, the downstream instability often decreases, with the trends more consistent for lower precision embeddings. In Figure 5, we further validate that as the precision increases, the downstream instability decreases. Finally, in Figure 6, we again see that when jointly varying dimension and precision, the instability decreases as the memory increases.



(a) 32-bit Precision

(b) 1-bit Precision

Figure 4. The effect of embedding dimension on the downstream instability of sentiment analysis tasks for CBOW and MC embedding algorithms. We show the results at two different precisions: (top) 32-bit precision (uncompressed), and (bottom) 1-bit precision (32× compressed).

C.2 Quality Tradeoffs

We also evaluate the quality-memory tradeoffs and quality-stability tradeoffs, finding that like stability, the quality also increases with the embedding memory. In Figures 7 (a) and 8 (a), we show the quality-memory tradeoff across sentiment analysis and NER tasks and embedding algorithms for different dimension-precision combinations. We see that the dimension tends to impact the quality significantly more than the precision (i.e., the change in dimension for a fixed precision affects the quality more than the change in precision for a fixed dimension affects the quality). Recall that in contrast, for

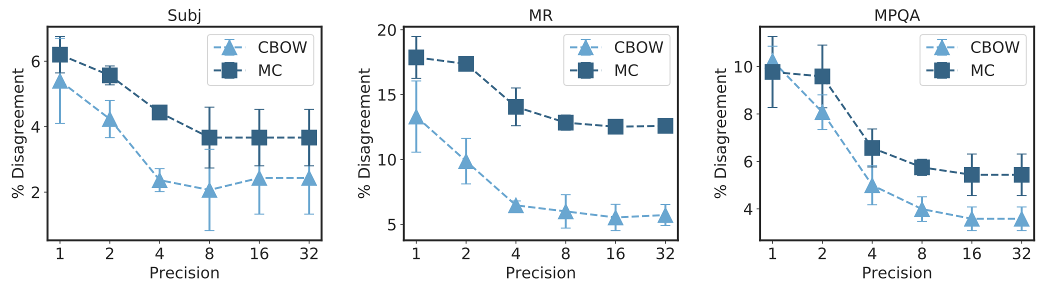


Figure 5. The effect of embedding precision on the downstream instability of sentiment analysis tasks for CBOW and MC embedding algorithms with 100-dimensional embeddings.

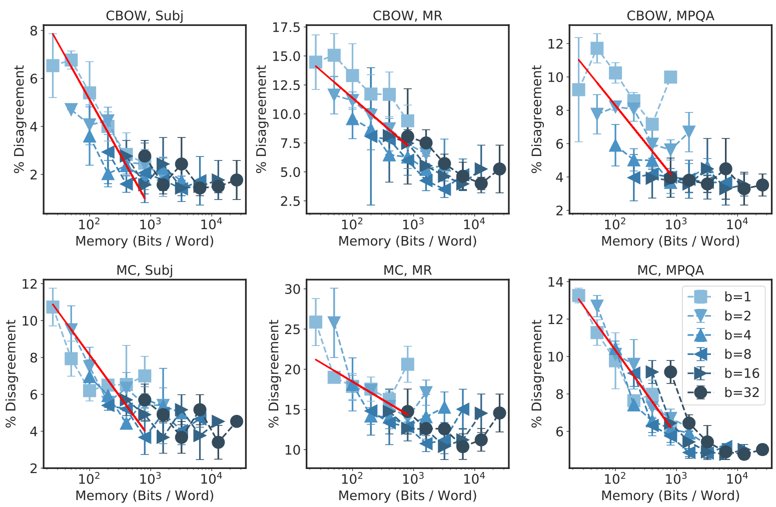
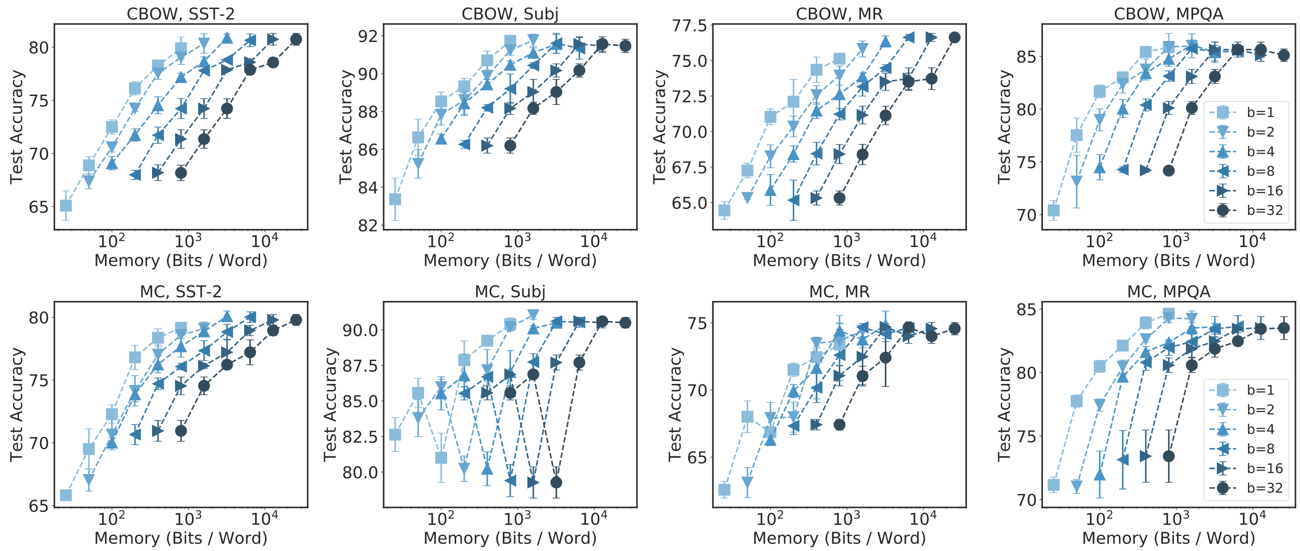


Figure 6. The effect of embedding dimension and precision on the downstream instability of sentiment analysis tasks for CBOW and MC embedding algorithms.

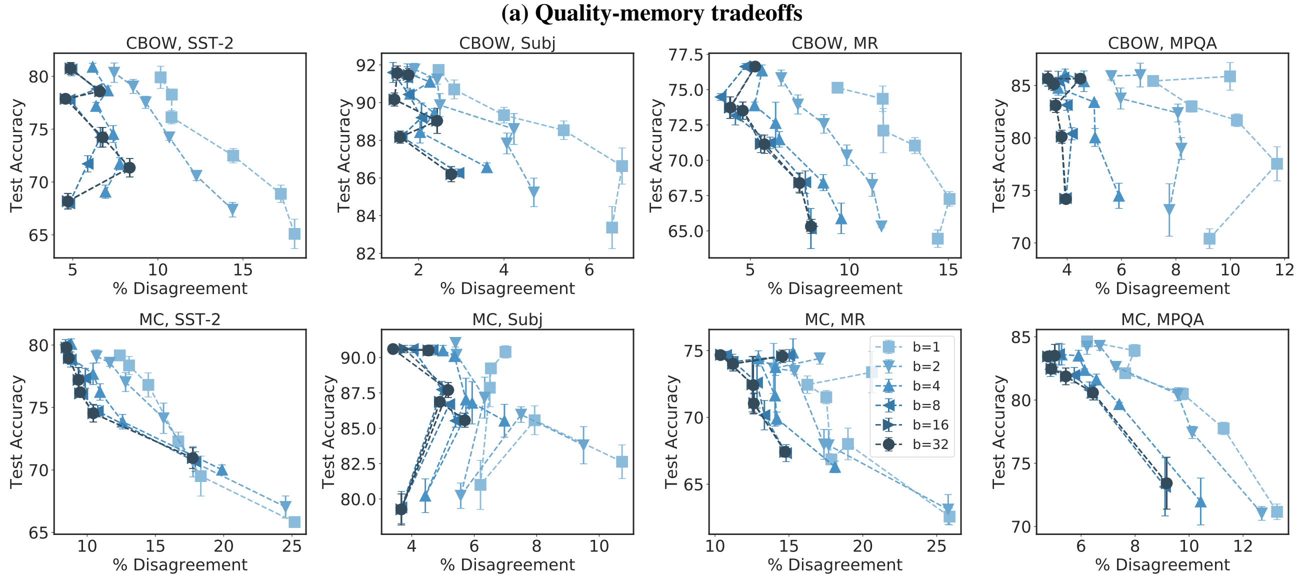
instability, we saw that the precision actually had a slightly greater effect than the dimension in Section 3.3. In Figures 7 (b) and 8 (b) we also show the quality-stability tradeoffs. For many of the sentiment analysis tasks, there is not significant evidence of a strong relationship between the two; however, for the NER task, we can clearly see that as the instability increases, the quality decreases. For several of the tasks (e.g., CBOW, MR; CBOW, MPQA), we can see that for different precisions (i.e., lines), the instability changes significantly, but the quality is relatively constant. This aligns with the previous observation that the precision tends to impact the instability more than it does the quality. In a similar way, for different dimensions (i.e., points), we see that the quality can change significantly while the instability may stay relatively constant, especially for higher precisions (e.g., CBOW, SST-2, CBOW, MPQA).

C.3 Selecting Hyperparameters for Embedding Distance Measures

The eigenspace instability measure and the k-NN measure each have a single hyperparameter to tune. For the eigenspace instability measure, α determines how important the directions of the eigenvalues of high variance are. For the k-NN measure, k determines how many neighbors are compared for each query word. To tune these hyperparameters, we compute the Spearman correlation between the embedding distance measure and the downstream prediction disagreement on the validation datasets for the five tasks and two embedding algorithms in our study. In Table 10a we report the average Spearman correlation for different values of α for the eigenspace instability measure where we see $\alpha = 3$ is the top-performing value. In Table 10b we report the average Spearman correlation for different values of k for the k-NN measure, where we see $k = 5$ is the top-performing value. Based on these results, we use $\alpha = 3$ and $k = 5$ for our experiments throughout the paper.



(a) Quality-memory tradeoffs



(b) Quality-stability tradeoffs

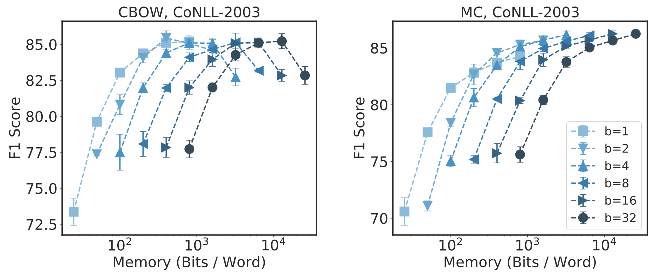
Figure 7. Quality tradeoffs for the sentiment analysis tasks with CBOW (top) and MC (bottom) embeddings for varying dimension-precision combinations.

C.4 Predictive Performance of the Eigenspace Instability Measure

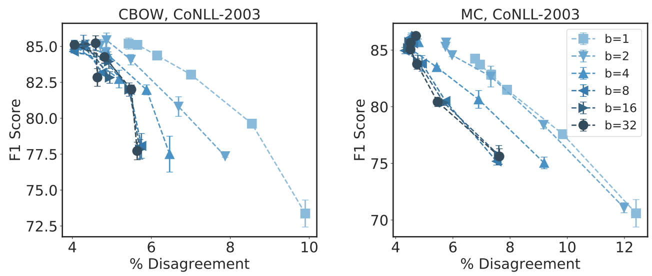
We now provide several additional results validating the strong relationship between the eigenspace instability measure and downstream instability. First, we show downstream instability v. embedding distance measure plots for each embedding distance measure, and then we show that the eigenspace instability measure demonstrates the same isolated trends with dimension and precision that we observed in Sections 3.1 and 3.2.

In addition to the Spearman correlation results we provide in Table 1, we visualize the downstream instability v. embedding distance measure results for the CoNLL-2003 NER task in Figure 9 with CBOW and MC embeddings, taking the average over three seeds. We see that k-NN measure and the eigenspace instability measure achieve strong correlations since the lines are generally monotonically increasing for both CBOW and MC embedding algorithms.

Empirically, we also validate that the eigenspace instability measure can explain the dimension and precision trends we observed in Sections 3.1 and 3.2. To explain the dimension trend, we compute the eigenspace instability measure between



(a) Quality-memory tradeoffs



(b) Quality-stability tradeoffs

Figure 8. Quality tradeoffs for the NER task with CBOW and MC embeddings for varying dimension-precision combinations.

Table 10. Average Spearman correlation ρ values for different values of α for the eigenspace instability measure (a) and k for the k-NN measure (b). Top value bolded.

(a) α for the eigenspace instability measure

α	ρ
0	-0.350
1	-0.067
2	0.498
3	**0.751**
4	0.748
5	0.741
6	0.738
7	0.739
8	0.739

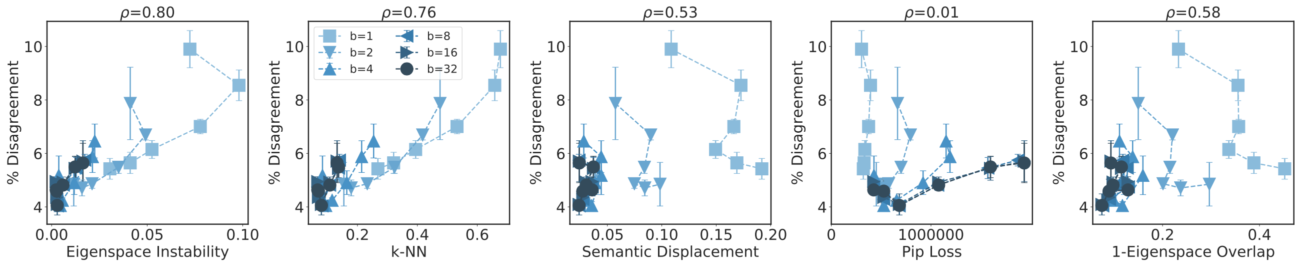
(b) k for the k-NN measure

k	ρ
1	0.766
2	0.777
5	**0.785**
10	0.782
50	0.774
100	0.763
500	0.703
1000	0.675

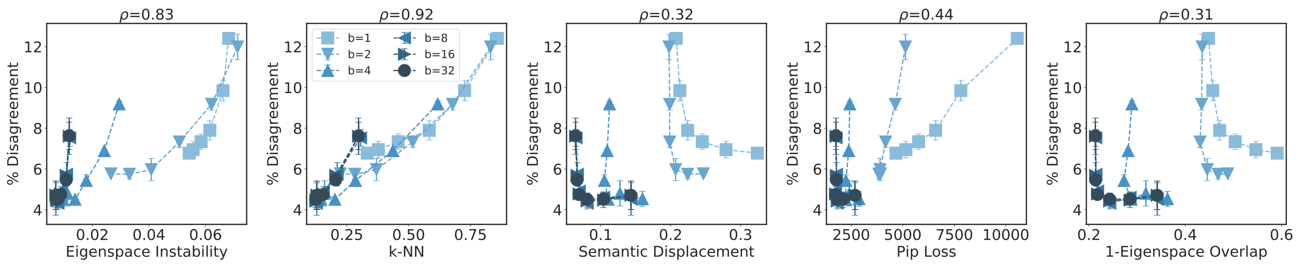
full-precision Wiki'17 and Wiki'18 embeddings of the same dimension for dimensions $\{25, 50, 100, 200, 400, 800\}$. To explain the precision trend, we compute the eigenspace instability measure between 100-dimensional Wiki'17 and Wiki'18 embeddings of the same precision for precisions $\{1, 2, 4, 8, 16, 32\}$. We use $\alpha = 3$ as described in Appendix C.3. In Figure 10, we see that as the dimension and precision increase, the eigenspace instability measure decreases.

C.5 Embedding Distance Measures for Dimension-Precision Selection

We evaluate the the worst-case performance of the embedding distance measures when used as a selection criterion for dimension-precision parameters. First, on the easier task of choosing the more stable dimension-precision pair out of two choices, we define the worst-case performance as the maximum increase in instability that may occur by using the embedding distance measure to choose the dimension-precision parameters (rather than the ground truth choice). On the more challenging task of choosing the most stable dimension-precision pair under a memory budget, we define the worst-case performance as the worst-case absolute percentage error to the oracle parameters under a given memory budget. We see in Tables 11 and 12 that the eigenspace instability measure and k-NN measure are the top-performing measures for



(a) CBOW



(b) Matrix completion (MC)

Figure 9. Downstream instability versus embedding distance measures for the NER task on the CoNLL-2003 dataset. ρ is the Spearman correlation between the embedding distance measure and the downstream instability.

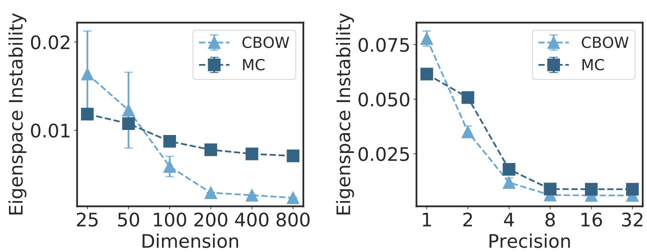


Figure 10. The eigenspace instability measure captures the stability-memory tradeoffs for dimension and precision for CBOW and MC embeddings.

the first task and in the top-three selection criterion for the majority of downstream models for the second task.

C.6 Knowledge Graph Embeddings

In Section 6.1, we showed that as the memory of the TransE embedding increases, the instability on link prediction and triplet classification task decreases; we now experiment with a modified setup for the triplet classification experiments. In Figure 11, we use thresholds tuned per dataset (in Figure 3 (right) we use the *same threshold* on both the FB15K-95 and FB15K dataset) and see that the stability-memory tradeoffs are less pronounced across the memory budgets. For low precisions, we continue to see that as the dimension increases, the instability decreases. For higher precisions, the instability decreases as we increase the dimension, but quickly plateaus for dimensions greater than 50.

C.7 Contextual Word Embeddings

We include the plots for the contextual word embedding experiments with BERT embeddings in Figures 12a and 12b for dimension and precision, respectively. As discussed in Section 6.2, although noisier than the trends with pre-trained word embeddings, we see that generally as the dimension and precision increase, the downstream instability decreases.

Table 11. Worst-case absolute percentage error when using each embedding distance measure to predict the most stable embedding parameters on downstream tasks over of all pairs of parameters. Downstream models are trained for sentiment (SST-2, MR, Subj, MPQA) and NER (CoNNL-2003) tasks. Lowest errors are bolded.

Downstream Task	SST-2		MR		Subj		MPQA		CoNNL-2003	
Embedding Algorithm	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC
Eigenspace Instability	**10.43**	13.18	**5.06**	**7.12**	3.50	**3.40**	**4.24**	5.00	3.30	4.11
k-NN	**10.43**	**11.75**	7.87	9.28	**2.80**	**3.40**	4.62	**3.68**	**2.17**	**3.16**
Semantic Displacement	11.70	16.80	11.06	14.71	5.40	7.10	5.66	7.63	5.13	7.03
PIP Loss	16.14	15.76	13.40	12.84	6.40	4.40	9.99	6.13	6.78	5.77
1-Eigenspace Overlap	12.69	16.80	10.59	14.71	5.50	7.10	5.94	7.63	5.86	7.03

Table 12. Worst-case absolute percentage error to the oracle downstream instability when using embedding distance measures as the selection criteria for dimension and precision parameters.

Downstream Task	SST-2		MR		Subj		MPQA		CoNNL-2003	
Embedding Algorithm	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC
Eigenspace Instability	3.08	11.37	3.94	6.37	**1.80**	**2.40**	**1.32**	2.54	**0.84**	1.73
k-NN	3.02	11.37	3.94	**2.62**	**1.80**	2.60	1.60	2.54	1.29	**1.01**
Semantic Displacement	**2.47**	13.95	**3.75**	9.56	**1.80**	3.10	2.36	3.96	1.73	3.92
PIP Loss	7.96	13.95	7.97	9.56	3.30	3.10	6.88	3.96	2.02	3.92
1-Eigenspace Overlap	10.43	13.95	3.84	9.56	**1.80**	3.10	2.26	3.96	1.29	3.92
High Precision	10.43	13.95	6.75	9.56	**1.80**	3.10	2.36	3.96	2.03	3.92
Low Precision	7.96	**5.33**	7.97	10.22	3.30	4.60	6.88	**2.36**	2.02	2.33

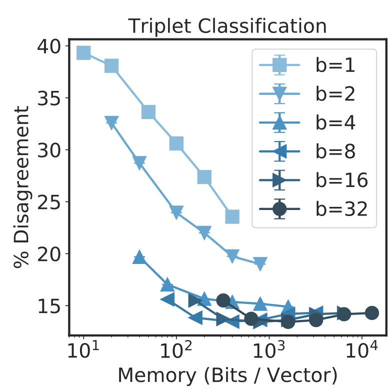


Figure 11. Stability of triplet classification when evaluating embeddings trained on 95% of FB15K training triplets and all of FB15K, and tuning the threshold for the task per dataset.

D ROBUSTNESS OF TRENDS

We explore the robustness of our study by providing preliminary investigation on the effect of more complex downstream models, other sources of randomness introduced by the downstream model (e.g., model initialization and sampling order), the downstream model learning rate, and fine-tuning embeddings on downstream instability.

D.1 Complex Downstream Models

In the main text, our primary downstream models are a simple linear bag-of-words model for sentiment analysis and a single layer BiLSTM for NER. We now demonstrate that complex downstream models such as CNNs or BiLSTM-CRFs can still

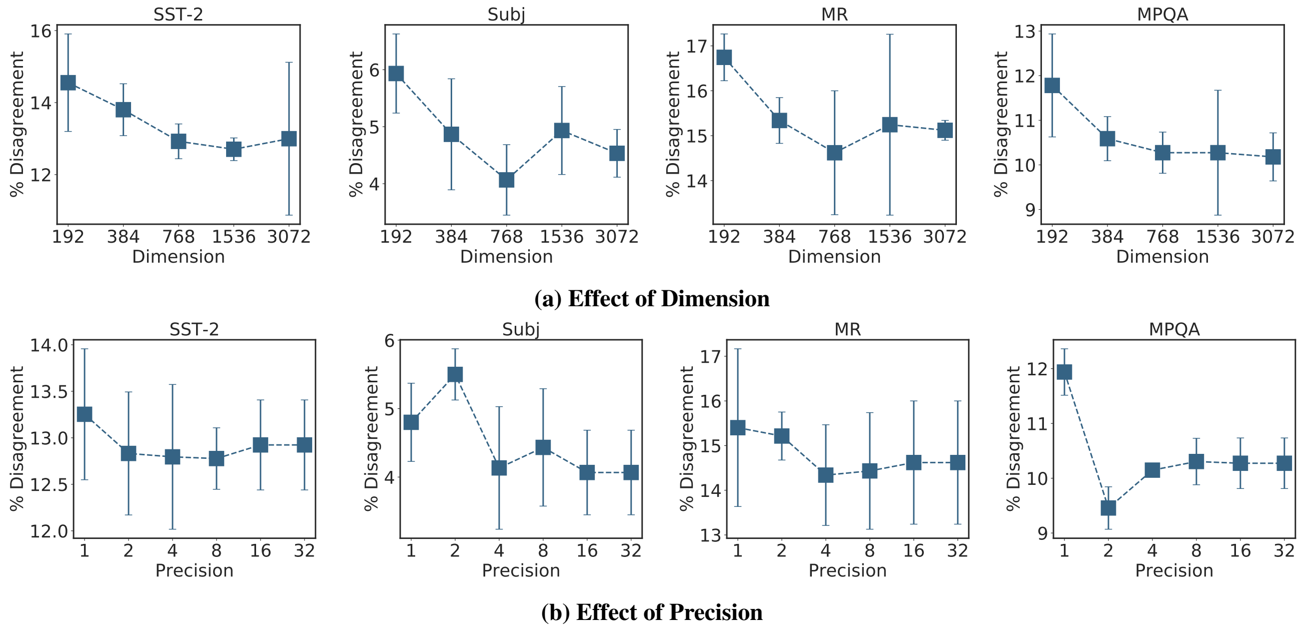


Figure 12. Downstream instability of the BERT embeddings on sentiment analysis tasks as a function of (a) transformer layer output dimensionality, and (b) uniform quantization precision.

demonstrate the stability-memory tradeoffs, such that as the memory increases, the instability decreases. In Figure 13, we show that when using a CNN for the SST-2 sentiment analysis task, embeddings with very low memory budgets result in high instability. The instability quickly plateaus for memory budgets greater than 10^2 for the CBOW embeddings, but continues to decrease until a memory budget of 10^3 for the MC embeddings. The CNN architecture has one convolutional layer, with kernels of widths 3, 4, and 5, and 100 output channels. The convolutional layer is followed by a ReLU layer, a max-pooling layer, and finally a linear classification layer. We sweep the learning rate in a grid of {1e-5, 0.0001, 0.001, 0.01, 0.1} and choose the best learning rate by validation accuracy. Shared hyperparameters are shown in Table 13 and selected learning rates are shown in Table 14.

Table 13. Training hyperparameters for the CNN architecture for sentiment analysis.

Hyperparameter	Value
Optimizer	Adam
Batch size	32
Training epochs	100
Dropout	0.5

Table 14. Selected learning rates for the CNN architecture for sentiment analysis task on the SST-2 dataset per embedding algorithm.

CBOW	MC
0.0001	0.001

We now demonstrate that the BiLSTM-CRF also demonstrates the stability-memory tradeoffs, where as the dimension and precision increase, the instability decreases (Figure 14). We use the same hyperparameters as in Table 8 and repeat our setup for the BiLSTM with the CRF turned on for the CoNLL-2003 NER task. Due to the CRF being computationally expensive, we train a representative subset of points (dimensions in {25, 100, 800} and precisions in {1, 4, 32}). For each embedding algorithm, we grid search the learning rate for the BiLSTM-CRF with 400-dimensional embeddings in {0.001, 0.01, 0.1, 1.0, 10.0} and find that a learning rate of 0.1 is best for both embedding algorithms.

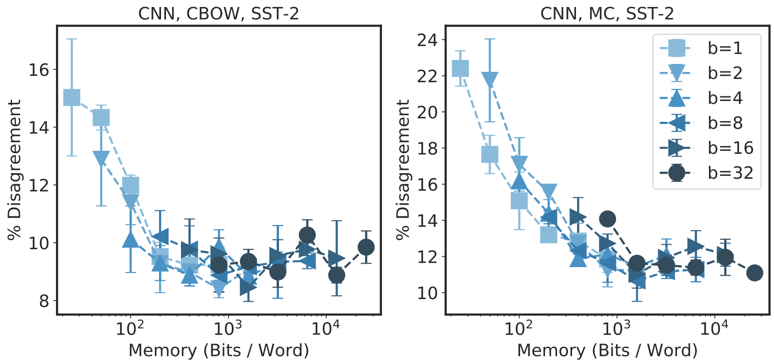


Figure 13. Downstream instability of the SST-2 sentiment analysis task with a CNN model for different CBOW and MC embedding dimension-precision combinations.

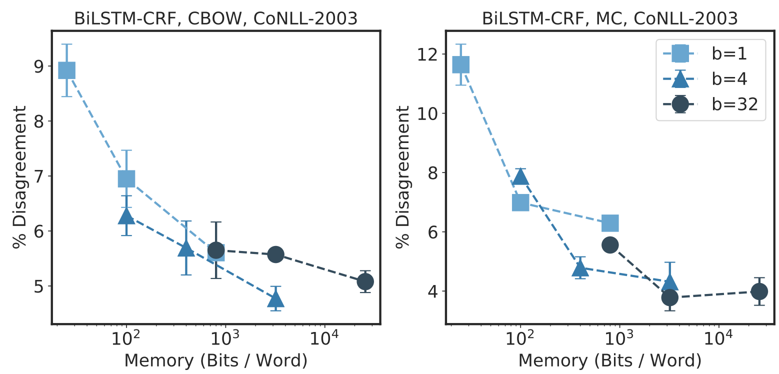


Figure 14. Downstream instability of the NER task with a BiLSTM-CRF model for different CBOW and MC embedding dimension-precision combinations.

D.2 Sources of Randomness Downstream

We study the impact of two sources of randomness in the downstream model training—the model initialization seed and the sampling seed—on the downstream instability. First, we fix the embeddings and vary the model initialization seed and sampling seed independently. We vary the sampling order by shuffling the order of the batches in the training dataset. We compare the instability from these sources of randomness in the downstream model training to the instability from the embeddings. For each source of randomness downstream, we fix the embedding (using a single seed of the Wiki'17, full-precision, 400-dimensional embedding), and measure the instability between models trained with different random seeds. We repeat over three pairs of models and report the average. We see in Table 15 that across four sentiment analysis tasks using the linear bag-of-words models, the sampling order seed introduces comparable instability to the change in embedding training data with full-precision, 400-dimensional embeddings, while the model initialization seed often contributes less instability. We note that using smaller memory budgets for the embeddings introduces much greater instability from the change in embedding training data, however, as shown in Figures 2 and 6.

In our experiments, we had also fixed the model initialization seeds and sampling order seeds to match that of the embedding, such that the seeds were the same between any two models we compared. We now remove this constraint, and vary the model initialization and sampling order seed of the model corresponding to the Wiki'18 embedding, such that no two models compared have the same seeds and otherwise repeat the experimental described in Section 3 and Appendix B.3. In Figure 15, we see that the stability-memory tradeoffs continue to hold and the trends are very similar to when we fixed the seeds (Figure 2). We note that many of the instability values themselves, particularly for CBOW, are slightly higher in Figure 15 than they are in Figure 2, likely due to the additional instability from the change in downstream model and sampling seeds.

Table 15. Using fixed 400-dimensional Wiki'17 embeddings and a linear bag-of-words model for sentiment analysis, we vary the model initialization seed and sampling order seed to measure their effect on downstream instability, compared to changing the embedding training data. Values represent the average percentage disagreement between models, and the largest instability is bolded for each embedding algorithm and task combination.

Downstream Task	SST-2		MR		Subj		MPQA	
Embedding Algorithm	CBOW	MC	CBOW	MC	CBOW	MC	CBOW	MC
Model Initialization Seed	3.48	7.08	2.44	9.28	1.10	4.53	1.45	4.30
Sampling Order Seed	**8.99**	5.96	**5.87**	10.09	0.57	**6.13**	**5.59**	1.92
Embedding Training Data	6.59	**8.66**	4.00	**11.22**	**1.50**	3.40	3.30	**4.78**

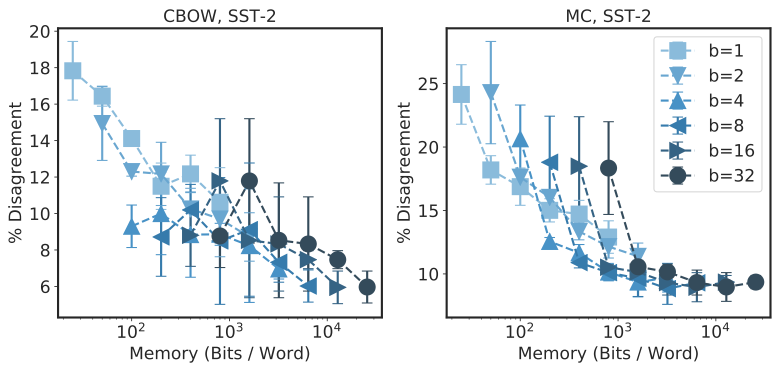


Figure 15. Downstream instability of the SST-2 sentiment analysis task for different CBOW and MC embedding dimension-precision combinations when we relax the constraint of having the model and sampling order seed be the same between models.

D.3 Effect of Downstream Learning Rate

We now study the impact of the downstream model learning rate on the instability, showing that the learning rate of the downstream model is another factor that impacts the downstream instability. In Figure 16, we show the instability of CBOW and MC embeddings on the SST-2 and MR sentiment analysis tasks when different learning rates are used for the downstream linear model. We mark the optimal learning rate by validation accuracy with a red star. We see that very small learning rates and very large learning rates tend to be the most unstable for both 100 and 400-dimensional embeddings. Moreover, the optimal learning rates do not significantly increase the instability compared to the other learning rates in our sweep. Since we see that the learning rate further contributes to the instability, we fix the learning rate in our main study to have a controlled setting to study the impact of dimension and precision on instability.

D.4 Effect of Fine-tuning Embeddings Downstream

We study the impact of fine-tuning the embeddings downstream and find that the stability-memory tradeoff becomes noisier, but continues to hold under fine-tuning, and fine-tuning can dramatically help to decrease the downstream instability. In Figure 17, we show that as the memory increases, the instability generally decreases for both CBOW and MC embeddings, even when we allow the embeddings to be updated (i.e., fine-tuned) when training the downstream models. We note that we do not compress the embeddings during training in these experiments, therefore the memory denotes the memory required to store the embedding *prior to training*. To perform the fine-tuning experiments, we follow the procedure described in Appendix B.3, and perform an additional learning rate sweep per embedding algorithm with fine-tuning in the grid {1e-5, 0.0001, 0.001, 0.01, 0.1, 1, 10}. We found the optimal learning rate for both algorithms on the SST-2 sentiment analysis task with fine-tuning to be 0.0001. We also see that overall the instability decreases with fine-tuning compared to fixing the embeddings (as we did in Figure 2). We note that the learning rate for the downstream model with MC and fine-tuning is smaller than with fixed embeddings, which may also contribute to the reduced instability; however, from Figure 16, the reduction in instability with fine-tuning still appears greater than that which can be achieved from a small change in learning rate alone.

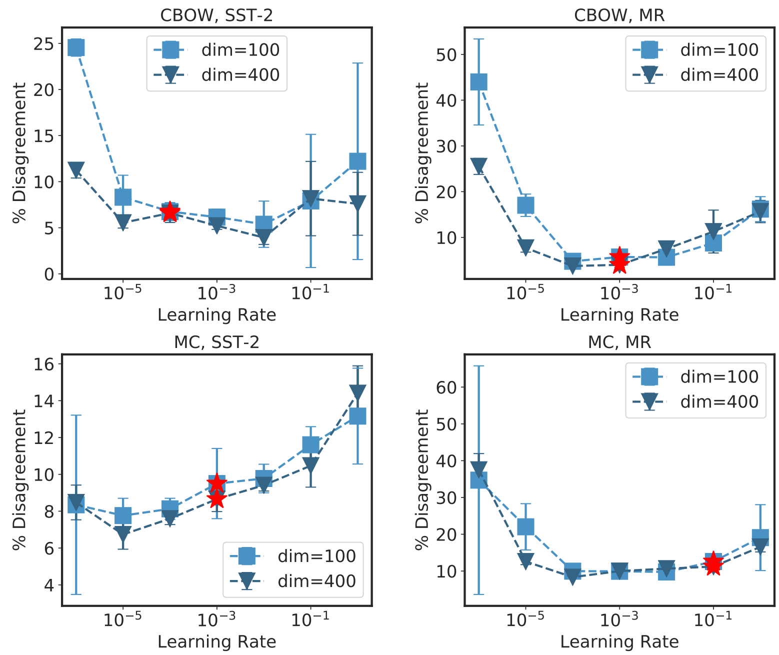


Figure 16. Downstream instability of CBOW and MC embeddings on the SST-2 and MR sentiment analysis tasks with various learning rates.

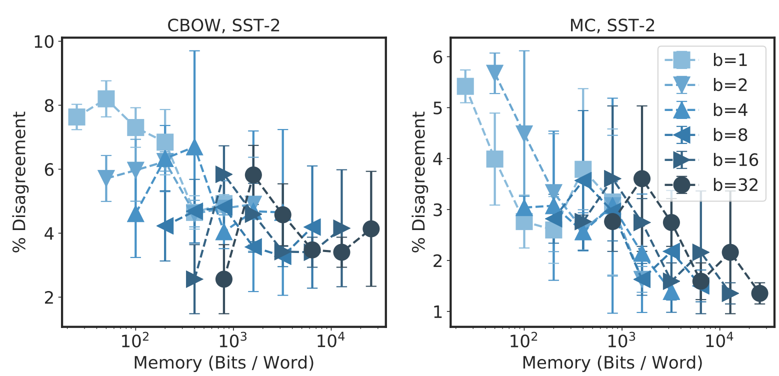


Figure 17. Downstream instability of CBOW and MC embeddings on the SST-2 sentiment analysis tasks when embeddings are fine-tuned. The memory indicates the embedding memory *prior to training the downstream model*, and embeddings are full-precision during downstream model training.