Inherently Interpretable Sparse Word Embeddings through Sparse Coding

Adly Templeton

A thesis submitted in partial fulfillment of the requirements for the Degree of Bachelor of Arts with Honors in Computer Science

WILLIAMS COLLEGE

Williamstown, Massachusetts

May 21, 2020

Abstract

Word embeddings are a powerful natural language processing technique, but they are extremely difficult to interpret. In order to create more interpretable word embeddings, we transform pretrained dense word embeddings into sparse embeddings. We construct these embeddings through sparse coding. By inherently interpretable, we mean a system where each dimension is mechanically associated with some *hint* that can capture the meaning of that dimension. We show that models trained using our sparse embeddings can achieve good performance and are interpretable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Word embeddings, a recent technique from *Natural Language Processing* (NLP), represent each word in a natural language as a vector in a continuous high dimensional space. Many different pretrained embeddings are readily available [2, 31, 39], and are used in a range of applications [24]. This vector representation attempts to encode the meaning of the word; not only are similar words close together, but linear relationships between words are thought to have conceptual meaning. For example, the vector difference between 'man' and 'woman' is similar to the vector difference between 'king' and 'queen' [20, 31]. This observation suggests that each dimension of these vector spaces might have its own 'meaning'. However, in practice, these dimensions have not been successfully interpreted.

To enable interpretable models, we create vectors where each dimension is *inherently interpretable*. By inherently interpretable, we mean a system where each dimension is mechanically associated with some *hint* that can capture the meaning of that dimension. By comparison, other systems of interpretable word embeddings aim to create dimensions that humans may be able to manually interpret. To do this, we represent word embeddings as the sparse linear combination of a basis set of other word embeddings. That allows us to represent each natural language word as the linear combination of a small number of other

natural language words. This representation is itself a more 'interpretable' word embedding. This technique produces representations of words that have interpretable dimensions. We show that these representations are more interpretable and also maintain most of the representational power of the original embeddings. We show how the creation of inherently interpretable vectors can help us understand the behavior and structure of the original word embeddings.

Recent work has created more interpretable vectors through a variety of methods (see Chapter 2.5). However, relatively few approaches create *inherently interpretable* dimensions. Therefore, we believe that our work, which creates inherently interpretable embeddings through a simple thought novel method, can be the basis of future NLP tools.

The remainder of this work will be structured as follows. In Chapter 2, we consider related research and position our contributions in the field. Chapter 3 describes the techniques we use to create sparse interpretable word embeddings. Chapter 4 presents results on the overall performance and interpretability of our method and discusses its use in downstream tools. Chapter 5 analyzes the embeddings produced by our method in greater detail. In Chapter 6 we take stock of our progress and discuss directions for future research.

Chapter 2

Previous Work

Modern artificial intelligence techniques depend on tools that have been developed over the past several decades, and our work is no exception. New techniques in statistical machine learning have been able to harness large amounts of data to solve NLP tasks. In this chapter, we discuss some of these techniques and highlight many important contributions to this active field.

2.1 Neural Networks

Neural Networks are a broad and powerful family of techniques that have powered many recent advances in *Machine Learning* (ML). The motivation behind neural networks, and much of machine learning in general, is supervised function approximation: Given some inputs x_i and associated labels y_i, we model $y_i = f(x_i)$, where f is some unknowable real-world function. Here, we view the data x_i and y_i as realizations of the theoretical inputs and outputs, x and y. For example, in image recognition, f may be the function that determines whether a given image (x is the pixels of the image), contains a cat (y is the true/false value). Under this framework, we want to approximate f with some computable function $\hat{f}$.

To do this, we want to examine a family of easily-computable functions and find an element of the family that closely matches f over the known x_i and y_i (the *training dataset*). Neural Networks are a popular family of functions for this purpose [13].

These functions are ultimately built from *linear transformations*. If x and y are vectors, then a linear transformation is a transformation that can be represented as a matrix, A, such that $xA = y$. Typically, an additional constant term is added to x to allow the linear transformation to have a nonzero intercept. To express a more complicated function, one might try composing multiple linear combinations, such as $xAB = y$. However, basic linear algebra shows that this composition is, itself, linear. To form more complex models, we use an *activation function*, a nonlinear function inserted between linear transformations. For instance, one common choice is the *Rectified Linear Unit* (ReLU), defined as $\phi(x) = max(0, x)$. Then, we can express functions such as $\phi(xA)B = y$ [13].

The coefficients in these matrices are known as *weights*, and appropriate weights are found through *backpropagation*, an iterative adjustment process. The *loss* is a function that measures how much the labels predicted by the network differ from the true labels. To train the network, the derivative of the loss function is computed with respect to each weight. Then, each weight is moved in the opposite direction of the gradient. To add randomness to this process, to avoid local minima, and for computational efficiency, we typically perform individual updates based on a small number of data points, instead of the entire dataset (*stochastic gradient descent*) [13].

2.2 Introduction to Word Embeddings

Word embeddings represent natural language words as points in high dimensional space. In practice, these embeddings have been effective representations in NLP. They are frequently used as the first stage in applications such as syntax analysis, sentiment analysis, and machine translation [24]. However, the reasons for their success are complex and not well understood.

We review various word embedding techniques.

2.2.1 Early Techniques for Word Embeddings

The first technique for embedding words, Latent Semantic Analysis, (LSA), was originally published as a method for indexing information for retrieval in textual data [7]. The core idea was the representation of objects as vectors in a vector space. Later, the power of LSA when applied to natural language words, was demonstrated in a Cognitive Psychology paper by the same authors[20]. Here, the goal was to model the behavior of children learning new words. In particular, the authors sought to address '*Plato's problem*', also known as 'the poverty of the stimuli', the claim that children are not exposed to enough text to infer the meaning of all the words that they know.

The authors of LSA, Landauer and Dumais, argue that children do not learn the majority of their new vocabulary either by direct instruction (being taught the dictionary definition of a word) or by direct inference (logical deduction of the meaning of a word from context, which is nearly impossible when starting from a very small vocabulary). Instead, children take in many tiny pieces of information about the contexts in which words are used. Children then, Landauer and Dumais argue, subconsciously reprocess this information to discern the meaning of language. That is, vocabulary is learned from a large number of tiny context clues that are meaningless individually, but powerful in aggregate.

Landauer and Dumais model this process as the *Singular Value Decomposition* (SVD) of a word-document matrix, where entry i, j in the matrix is the frequency of word i in document j. They show that this technique can learn the meaning of words from a dataset that is of size comparable to the data to which children are exposed.

Landauer and Dumais place particular emphasis on the relationship between dimensionality reduction, which is implemented here with matrix factorization, and the inductive reasoning described above. Typically, dimensionality reduction techniques are used to condense the

information in the original matrix into a more tractable form or to filter out noise in the data. However, in the case of LSA, dimensionality reduction seems to be the basis for the primary inductive benefits of the technique. The performance of LSA, evaluated on a synonym test, peaks sharply around 300 dimensions (Figure 2.1). In particular, LSA performs very poorly when no dimensionality reduction is used.

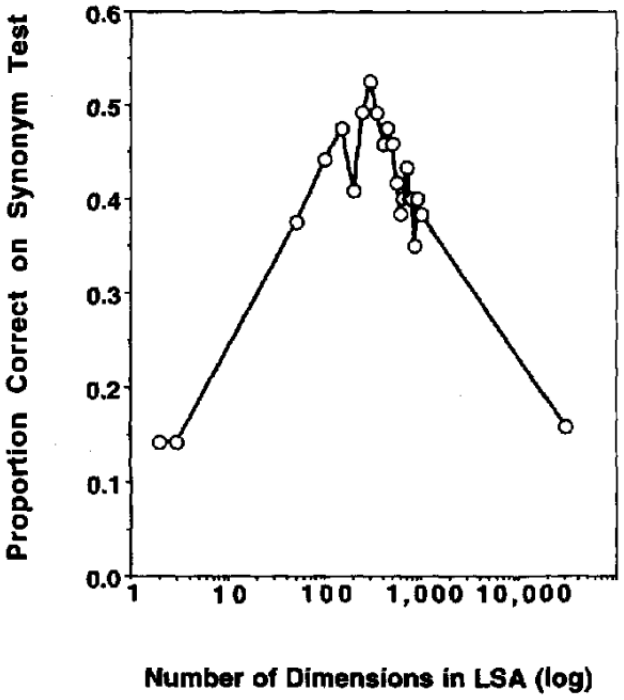

Figure 2.1: The performance of LSA, measured by the TOEFL synonym test, at different dimensionalities. Reproduced from Landauer and Dumais [20].

Landauer and Dumais argue that limited dimensionality leads to induction by imposing constraints. Without dimensionality reduction, LSA is essentially equivalent to a vector of the correlation between words. Instead, Landauer and Dumais argue that the constraints force the model to consider indirect relationships between words. Words can fail to co-occur for many reasons. For instance, they could be synonyms with such similar meaning that both are rarely used in the same context, or are words that have somewhat related meanings

and fit into the same slot in a sentence, or are simply unrelated words. To distinguish between these nuanced relationships, dimensionality reduction considers indirect relationships between words. Landauer and Dumais hypothesize that this is the purpose of the dimensionality constraint [20]. Competing explanations for the effects of dimensionality reduction are a recurring theme of the literature on word embeddings as well as an important question in our work.

Although LSA was a theoretical breakthrough in information retrieval and natural language understanding it was only marginally useful in practical complicated natural language understanding tasks in practice. One of the major limits was computational: performing SVD is computationally expensive and so LSA could not be used with large vocabulary. A new and effective technique by Mikolov et al.[31], *word2vec*, made word embeddings practical and became the basis for many modern NLP systems [24]. Word2vec computes word embeddings using a neural network. This neural network is trained to approximate the function that predicts words from their contexts. The use of a neural network for word embeddings was not revolutionary by itself—Landauer and Dumais imagined a neural network that would approximate LSA in their original paper—but the major innovation of word2vec was *negative sampling*, in which the network only adjusts weights based on a fraction of the incorrect predictions. This innovation greatly reduced the computational cost of updates and distributed computation more evenly across low-frequency words [31]. Later work would also show that negative sampling may have some benefits beyond computational efficiency [22]. See Chapter 2.4 for more details.

The increased efficiency allowed researchers to reformulate the training objective. Instead of learning word-document relationships, Mikolov et al. associated each word with a small window of a few surrounding words. The most popular model, *Skip-Gram*, trains a neural network to take a word as input and predict the surrounding window of words. Then, the intermediate layer learned by this network is used as the word embeddings [31].

With word2vec, vector embedding models finally displayed the linear vector properties de-

scribed in Chapter 1. Of particular practical use is word2vec's ability to serve as the input to a neural network for more complex NLP systems. Neural Networks cannot easily learn from untransformed information about the actual words in a sentence. The large number of inputs leads to *overfitting*, or memorization of the training set without generalization. But neural networks can very easily learn from the output of word2vec or similar systems that are constrained to comparatively fewer dimensions. Since simple neural networks can be thought of as generalized linear regression, this implies that linear combinations of dimensions of word embeddings, and, by extension, the dimensions themselves, likely have some level of meaning. However, it was still not possible to mechanically interpret meaning.

2.3 Sparse Neural Networks

Sparse neural networks are any neural networks where a large number of the weights are zero. Much recent work has described different approaches to sparse neural networks. These approaches have used sparsity to accomplish different goals.

2.3.1 Sparsity as Regularization

Neural networks are extremely powerful models, and sufficiently large neural networks can represent functions that can fit any training data [6]. However, this power can lead to overfitting. *Regularization* is a family of techniques that attempt to decrease this generalization error. Most commonly, these techniques encode, implicitly or explicitly, a preference towards simpler models [13]. One common class of regularization techniques are methods that add a penalty term to the objective function of the neural network. For instance, recent neural networks frequently use a penalty term proportional to the Euclidean (L_2) norm of the weights [13].

This penalty term changes the objective function of the neural network. Instead of finding

the closest possible approximation to the empirical training-set target function, we now want to find a function that still gives high-quality approximations but that also has small weights. The hope is that even if this function does not fit the training-set target function as well, it will generalize better to unseen data by closely fitting the unknowable real-world target function. Intuitively, smaller weights generalize better because smaller weights correspond to functions with smaller derivatives. For example, penalizing the weights will prevent a function from bending significantly to correctly classify a single data point in the training set.

From this perspective, the desire for sparsity can be viewed as a penalty term. In particular, we want to penalize the *number* of non-zero entries in the weight matrix. This goal corresponds to a L_0 penalty term, which imposes a penalty based on the number of non-zero weights. In practice, this penalty term is non-differentiable and hard to optimize. Instead, a L_1 penalty term proportional to the magnitude of the weights pushes weights to zero and is effective at inducing sparsity.

A more complex form of regularization, *dropout*, is also intimately tied to sparsity. Dropout is a popular technique for regularizing neural networks that, for each training example, randomly selects a percentage of the training weights and sets them to zero for the duration of that example [42]. For each training example, this means that only a sparse subnetwork is trained. The effect is a regularization that effectively trains all sparse subnetworks of the original network simultaneously.

A technique called *variational dropout*, based on a Bayesian theoretical framework for neural networks called *variational inference*, allows the neural network to learn per-parameter dropout rates during training [17]. Molchanov et al. extend this result by allowing the network to learn arbitrarily high dropout rates for individual parameters. If a parameter is selected for dropout all of the time, then it is effectively removed from the network. Therefore, the network is pushed to learn sparsity by pruning unneeded weights during training, regularizing the network [32]. In the Bayesian framework, the dropout parameters mea-

sure the uncertainty associated with a particular parameter and quantifying this uncertainty naturally leads to regularization.

Sparsity as regularization offers a compelling motivation for sparse neural networks. Modern neural networks, including those of word2vec, have a large number of parameters. It is likely that many of these are not necessary and may lead to overfitting. Still, sparse neural networks may yield other compelling benefits besides regularization. We will explore these benefits in the next section.

2.3.2 Sparsity, Dimensionality, and Compression

Another family of work considers sparsity to be a computational tool for efficiently compressing neural networks. Their goal is to find a small subnetwork that maintains almost the same performance, reducing the space and computation requirements. Unlike with regularization, their goal is not to improve, but to maintain the accuracy of the network as much as possible. The benefit is a much smaller network.

Han et al. [14] present a foundational pruning-based approach to compressing neural networks using sparsity. They train a dense neural network using some form of regularization (L_1 or L_2 norm), prune small learned weights, and then retrain the new, smaller subnetwork. For high pruning levels, multiple pruning and retraining iterations works better, and the L_2 norm is found to work better with iterative retraining. They are sometimes able to achieve significant compression with little drop in performance. For example, they can prune up to 90% of the parameters without a significant decrease in performance on the ImageNet dataset.

Though there have been a wide variety of similar approaches to sparsify neural networks after training, Gale et al. [12] found that, in practice, these techniques are not robust and, when applied to a wide range of datasets, do not work better than simply pruning weights based on magnitude in many scenarios.

Louizos et al. [14] provide a theoretical framework for compression-motivated sparsity based on variational inference. They model both sparsity-introducing priors and learned precisions (quantization) for each weight. Instead of a traditional sparsity-inducing prior, with a discrete probability at zero and a wide normal distribution, they use a more computationally tractable continuous relaxation. The goal here is to force the network to make the tradeoff between performance and storage space in an embedded system.

Though our primary interest is not compression, the degree of compression that these methods are able to achieve suggests that we can achieve good performance with few nonzero parameters.

2.3.3 Sparsity and the Lottery Ticket Hypothesis

Overparameterization may be explained by another perspective on sparsity, the *lottery ticket hypothesis*. Frankle and Carbin [10] hypothesize that a randomly-initialized network contains a sparse 'lottery winner' subnetwork that can match the accuracy of the original network. The weights in a neural network are highly symmetric, and this symmetry is only broken by the random initialization of the weights. Once identified, these subnetworks can be easily retrained, often faster and better than the dense network, but only when starting with the same initializations. Frankle et al. [11] later extend the lottery ticket hypothesis for deep networks. While pruning down to the lottery ticket winner and resetting the weights does not work as well for deeper networks, they show that pruning and 'rewinding' all weights to an early stage of training still works.

Zhou et al. [46] perform a series of experiments to explain the successful performance of winning tickets. They find that the signs of these weights, positive or negative, are crucial, and show that reinitializing to random magnitudes while keeping the signs the same achieves almost the same performance. In addition, if they 'freeze' weights that increase in magnitude, and set to zero all weights that decrease in magnitude, they can exceed the performance of

the original lottery ticket method. They also demonstrate that a lottery winner subnetwork achieves better-than-chance performance even before training.

Morcos et al. [33] shed further light on the behavior of the lottery ticket hypothesis. They show that winning tickets that work on one task can be generalized to similar tasks or similar datasets. This implies that the winning tickets provide legitimate inductive biases.

The lottery ticket hypothesis has been controversial. Other work, such as that of Liu et al. [26], characterizes network pruning as architecture search and argue that the network topology, not initializations, are the defining characteristic of network performance. However, Frankle et al. [11] suggested that this observation is only true for moderate levels of sparsity.

Whatever the explanation, it appears that sparse network architecture, and perhaps sparsity, however it is identified, is a sufficiently powerful optimization technique for modern applications.

2.3.4 Sparse Coding

Sparse coding is a family of non-neural-network techniques that represent vectors in a sparse way. Sparse coding finds a representation for a set of dense vectors such that each vector is approximated by a linear combination of a small minority of a *basis set* of vectors.

Sparse coding is a well-studied optimization problem. Lee et al. [21] provide a technique to solve standard sparse coding problems. They formulate an optimization problem in terms of the basis and coefficient matrices, achieving sparsity by a prior belief that the optimal coefficients have a high probability of being zero. They alternate, optimizing one matrix at a time. Optimizing the coefficient matrix is equivalent to a L_1-regularized least-squares optimization problem, and they present a new computational technique that can solve these problems more efficiently in the sparse context. They do this by guessing signs of each coefficient, which reduces the absolute value in the regularization penalty to a differentiable

function, optimizing the resulting problem, and then refining these guesses. Other optimization techniques are provided by Mairal et al. [28], who propose an algorithm that resembles *stochastic gradient descent* for online sparse coding. Hoyer [16] derives similar optimization techniques for non-negative sparse coding.

Coates and Ng [5] explore the success of sparse coding by examining the effect of the basis set. They find that the success of sparse coding largely does not depend on the basis set and that even a random basis set produces good results. Of particular interest to our work, they find that randomly selecting vectors from the set that is being encoded is a very good baseline.

Sparse coding provides yet another perspective on sparsity: a motivation based on alternative representations of information. This perspective of meaningful representation motivates our work.

2.4 Theory of Word Embeddings

The first embeddings of words, LSA, were found through matrix factorization. Although this matrix factorization can be easily approximated with a neural network, word2vec diverges enough from LSA that it is not immediately apparent that word2vec is equivalent to a form of matrix factorization. Levy and Goldberg [22] provided a theoretical formulation of word2vec (specifically, the skip-gram model) as a neural network that implicitly factors the *pointwise mutual information* matrix between words and their contexts. The pointwise mutual information between a word w and a corresponding context c is defined as:

$$\text{pmi}(w, c) = \log \frac{p(w, c)}{p(w)p(c)} \tag{2.1}$$

This measures the bits of information shared between two individual events. They also show that the negative sampling used in word2vec shifts this matrix by subtracting $\log k$, where k is the negative sampling constant, and clamping the values at zero. In effect, this regularizes

that matrix by removing the noise of smaller values. Word2vec optimizes for a solution to this factorization problem, where each word is weighted by the corresponding number of observations [22].

They also show that either using this matrix by itself as high dimensional sparse vectors or exact matrix factorization achieves similar results to word2vec itself on word analogy tasks. This result is especially perplexing given that Landauer and Dumais showed that the opposite was true for LSA [20]. This contradiction suggests that something in the word2vec model makes the actual dimensionality reduction redundant or unneeded. A plausible explanation is that dimensionality reduction acts primarily as a form of regularization, so it can be supplanted by more training data or the explicit theoretical regularization they used. Similarly, Li et al. [25] formulate skip-gram negative sampling as matrix factorization in a way that is theoretically similar to that of Levy and Goldberg but cast as the factorization of a different matrix relative to a different loss function.

Other theoretical approaches provide further explanation of the dimensionality mystery. Melamud and Goldberger [30] find yet another interpretation of skip-gram negative sampling, casting it in information-theoretic terms. They propose a dependency measure between two random variables, Jensen-Shannon mutual information (JSMI), which measures the difference in distributions between the joint distribution of two variables and the product of their marginal distributions. They show that the JSMI is an upper bound on the information in the low dimensional embeddings used in word embeddings. This allows them to develop a theoretical justification of information loss by representation in a lower dimensional space [30]. This approach leads to a view of reducing dimensionality as compression of information, suggesting that higher dimensional embeddings are at least as good as lower dimensional embeddings.

Yin and Shen [44] present a theoretical justification for regularization through dimensionality reduction. They explore, and theoretically justify, the heavy dependence on the word embedding's dimension. First, they define the *pairwise inner product loss*, which measures

the dissimilarity between any word embeddings. Then, treating the factorization of a random matrix as a classical statistical estimator, they can analytically find the bias–variance tradeoff. Finally, they compute the optimal dimensionality and show that the theoretical optimum matches empirical observations almost exactly.

However, recent large-scale models, such as those developed by Lan et al. [19], which learn embeddings without matrix factorization, also benefit from reducing dimensionality. The dimensionality they used is still around the same optimal dimensionality that has appeared throughout embedding models.

2.5 Previous Interpretable Word Embeddings

A number of attempts have been made to make word embeddings more interpretable [4]. In this context, interpretability is often measured by the *word intrusion task*. In this task, a group of a few words is selected such that one word is on the opposite end of a particular dimension from the others. Human participants are then asked to select the unusual word from the group. The hope is that, if humans can consistently identify the outlying words in a dimension, then that dimension can be said to have a human-interpretable meaning.

Some techniques for interpretable word embeddings seek spatial rotations that make embeddings more interpretable. The training objective of word embeddings is rotation-invariant, so the final orientation of embeddings is arbitrary. Therefore, important concepts are likely represented not by one dimension but by the linear combination of many dimensions, inhibiting interpretation of basis dimensions.

Park et al. [37] improve interpretability through rotation using techniques typically associated with factor analysis. Factor analysis is a commonly used statistical technique that also produces matrices that are rotation-invariant and hard to interpret. These techniques find rotations where values are either large or close to zero. Effectively, although the total

dimensionality is unchanged, these techniques create almost-sparse rotations, and hint at some of the benefits of sparsity. Other work [8, 41] rotates dense vectors using different methods.

Koç et al. [18] tie concepts to dimensions in a more direct way. They select a concept for each dense dimension, using Roget's thesaurus, and identify words that are associated with these concepts, either positively or negatively. They then add an additional term to the objective function that pulls the set of concept words towards the appropriate end of the associated dimension. This approach can successfully rotate the vector space, increasing performance on the word intrusion task, and cause related but non-forced words to have appropriate values in that dimension. However, this approach does not allow for novel concepts to be identified. The power of word embeddings is their ability to expand beyond human-enumerated concepts and to discover new conceptual dimensions. Approaches such as this one may limit either the emergence or discovery of subtle meaning.

Other work has focused on interpretability through sparsity. Subramian et al. [43] created more interpretable embeddings by passing pretrained dense embeddings through a sparse autoencoder. This autoencoder minimized both the *average sparsity loss*, which measures per-dimension activations across the dataset and the *partial sparsity loss*, which penalizes values far fromzeroand ond. They significantly increase interpretability, as measured by the word intrusion task, without a significant drop in accuracy, measured on multiple downstream classification tasks.

Panigrahi et al. [36] proposed Word2Sense, a generative approach that models each dimension as a *sense* and word embeddings as a sparse probability distribution over the senses. These senses are computed using *latent Dirichlet allocation* (LDA), a generative statistical model that models each document as a collection of a small number of senses, each of which generates some of the words in that document. These vectors are particularly focused on representing words with multiple senses.

Sparse coding is a well-studied optimization problem [5, 16, 21]. Previous work [5] has

also shown that basis vectors can be selected from the set that is being encoded without significantly decreasing efficiency.

Faruqui et al. [9] used non-negative sparse coding to recode dense word embeddings into more interpretable sparse vectors while learning a basis. They show that these embeddings are more interpretable, as measured by the word intrusion test, and that these sparse embeddings are better inputs for a range of tasks.

Zhang et al. [45] also used non-negative sparse coding to learn a set of *word factors* to recode word2vec embeddings. The basis vectors created in this way are highly redundant, so they use spectral clustering to remove near-duplicate factors. Then, they are able to manually infer reasonable post hoc interpretations for most of the factors. This produces robust and reasonable interpretations, with both syntactic and semantic factors. In particular, they show that the learned factors can be added to word embeddings to reach other word embeddings in a way that makes intuitive sense. For instance, they show that the nearest neighbor of the vector for 'Italy' plus 4 times the vector for 'national' is the vector for 'Italian'.

Concurrently with our work, Matthew et al. create an inherently interpretable subspace from pairs of antonyms. They then project dense word embeddings into that subspace, producing lower dimensional dense vectors [29].

2.6 Summary

Past work has demonstrated that words can be effectively represented as vectors. This 'embedding' has also proven to be a fundamentally useful foundation for building more advanced tools that can understand or interpret natural language. But these embeddings are difficult to interpret. A number of methods aim to make these embeddings more interpretable. However, most of these methods still require humans to manually examine each dimension in order to assign a conceptual interpretation to each dimension.

At the same time, there has been a recent growth in research into sparse neural networks. This recent research has shown that sparse neural networks can achieve similar performance to dense neural networks. In the next chapter, we will apply the techniques of sparse neural networks to create word embeddings that can be interpreted as the sparse linear combination of a basis vocabulary.

Chapter 3

Our Model for Interpretable Sparse Word Embeddings

The goal of this work is to create a method that encodes words as embeddings that are interpretable while retaining representational power. In particular, we want a system of embeddings that inherently interpretable. Ideally, each dimension would have a label that completely capture the meaning of their associated dimensions. In reality, for some dimensions, the label is more like a *hint*, which provides a starting point for human interpretation. In this chapter, we will describe a method for embedding words in a sparse vector space that achieves these goals.

3.1 Sparse Coding Glossary

Here, we define a number of terms that we will use throughout the remainder of this work. In particular, we use some terms slightly differently from their traditional, formal, mathematical meaning.

- *word embedding*: The vector that corresponds to a natural language word.
- *vector space*: A matrix that represents a mapping between a large set of natural language words and their word embeddings.
- *dense vector space*: A vector space where words are represented by vectors with many non-zero components. We will use the dense vector spaces created by previous work (in particular, the FastText pretrained dense vectors [2]) as the input into our model.
- *sparse vector space*: A vector space where words are represented using only a few non-zero components.
- *basis vector*: One of the vectors that makes up the rows of the basis matrix used in sparse coding. Out of necessity, these vectors are *overcomplete* (there are more dimensions than vectors) and so they do not form a basis according to the traditional definition.
- *coefficient*: The element of a vector at a specified dimension.
- *norms:* The L_1 norm of a vector, denoted $||x||_1$ is the sum of the magnitudes of its elements. Formally,

  $$||x||_1 = \sum_i |x_i|$$

  The L_2 norm is the square root of the sum of the squared elements, denoted $||x||$. Formally,

  $$||x|| = \sqrt{\sum_i x_i^2}$$

  We define both norms on matrices as the sum of the norms of the vectors along the rows of the matrix. The L_2 norm is also known as the Euclidean norm.
- *cosine similarity*: A measure of the distance between two vectors. The cosine similarity between two unit vectors v_1 and v_2 is defined as:

  $$cos(v_1, v_2) = \frac{\sum_i v_{1_i} v_{2_i}}{||v_1||\ ||v_2||}$$

The cosine similarity is equal to the cosine of the angle between the two vectors.

We naturally expand this notion to the *cosine dissimilarity*, defined as $1 - \cos(v_1, v_2)$.

3.2 Sparse Coding

Our work uses *sparse coding* (see Chapter 2.3.4) to transform a set of word embeddings from a dense and uninterpretable space into a sparse and interpretable space. Let v_D represent a dense word embedding, and let $\mathcal{B}$ represent a matrix with basis vectors along the columns. We achieve sparse coding using regularized regression, inducing sparsity using the L_1 norm. Formally, this corresponds to finding the sparse vector v_S that minimizes the following objective function

$$\arg\min_{v_S} \; ||v_D - v_S\mathcal{B}||_2^2 + \alpha||v_s||_1 \tag{3.1}$$

α is a hyperparameter that controls the level of sparsity. The first term in Equation 3.1 ensures the sparse vector corresponds to a vector in the dense space that is similar to the original vector. The second term is a sparsity-inducing penalty.

Previous work using sparse coding to create interpretable word embeddings has considered the basis $\mathcal{B}$ to be part of the optimization problem [9, 45]. Our key innovation is that we draw the columns of the basis from the original set of dense word embeddings. This strategy provides a natural label for each dimension in the sparse space.

3.3 Syntactic Basis

We can roughly divide the 'meaning' carried by a word embedding into *syntactic* and *semantic* properties. Here we use 'syntactic properties' to mean properties that describe how that word fits into the grammar of the language, such as its part-of-speech, tense, or number.

We use 'semantic properties' to mean all other aspects of the meaning of a word. For instance, we expect the embedding for the word 'swimming' to include a syntactic component representing that this word is a present-tense participle and a semantic component that represents the meaning 'to swim'. Of course, this deconstruction is imperfect. Nevertheless, this approach provides a useful insight towards decomposing the meaning of a word embedding.

Intuition suggests that word-labeled dimensions poorly capture the syntactic properties of word embeddings. Preliminary experiments showed that, without special consideration, syntactic properties would be captured in an unintuitive way. To illustrate this, we display representations from preliminary experiments. The following settings are used to produce these sparse representations: We use FastText [2] pretrained dense vectors. The basis set is the most frequent 20,000 words in the FastText corpus, excluding 'Swimming', 'swimming', and 'swim'. The word frequency limit is set lower than in other places in this work in order to limit the number of terms in the vectors, for readability. α is set to 0.75.

$$\begin{aligned} \text{swim} = {} & 0.73 * \text{swimmers} + 0.20 * \text{dive} + 0.19 * \text{walk} \\ & + 0.15 * \text{sink} + 0.13 * \text{swimmer} + 0.04 * \text{float} \end{aligned} \tag{3.2}$$

$$\begin{aligned} \text{swimming} = {} & 0.45 * \text{swimmer} + 0.41 * \text{swimmers} + 0.28 * \text{pool} \\ & + 0.13 * \text{diving} + 0.03 * \text{rowing} \end{aligned} \tag{3.3}$$

The syntactic components cannot be easily isolated to one subset of the nonzero dimensions. Ideally, the appropriate Instead, each basis vector captures part of the syntactic component and part of the semantic component. This duality creates difficulty when interpreting our representations. In addition, it greatly increases the number of basis vectors required to form interpretable representations.

To address this, we construct a small number of *syntactic basis vectors* and add them to the basis set. For instance, we construct a 'POS-NOUN' vector by taking the mean of all word

embeddings corresponding to nouns.

Our approach makes use of four types of syntactic basis vectors:

1. We use the first principal component of the embeddings of the 30,000 most frequent words. Previous work on word embedding has referred to this as the *common discourse vector*, or c_0, and has shown that this vector encodes words that appear commonly in all contexts, such as 'the'.

2. We take the mean of all vectors of capitalized words and use this as a syntactic basis vector to represent capitalization.

3. For a variety of parts-of-speech, we use the mean vectors for words with that part-of-speech (POS). Specifically, we encode a vector for each of the following: nouns, verbs, adjectives, adverbs, and numbers.

4. We create mean vector differences for the following syntactic concepts: the relationship between singular and plural nouns, the relationship between present-tense verbs and their present participle form, and the relationship between present-tense verbs and their past-tense forms. For each of these relationships, we manually collect approximately 50 example word pairs that fit that relationship. We manually filter for word pairs where either the syntactic relationship does not change the form of the word (i.e., 'deer') or for word pairs where the syntactic change is likely to produce a more complicated change in meaning (i.e., 'math' and 'maths'). We average the differences between pairs of each relationship type and use it as the vector for that relationship.

These basis vectors help us to disentangle syntax from semantics. The choice and construction of these syntactic basis vectors is highly arbitrary, and different syntactic basis vectors could easily be used in different applications or in follow-up work.

Next, we make the syntactic basis vectors orthogonal using the Gram-Schmidt process. Primarily, we do this for mathematical reasons: the projection of vectors with respect to

the *complete syntactic basis*, is only unique if the basis is orthogonal. However, we also hope that this process can better align each syntactic basis vector with the concept that we created it to represent. Ideally, the concepts represented by the syntactic basis vectors are distinct and independent. Orthogonality is a way of enforcing this constraint. The Gram-Schmidt process iteratively creates an orthogonal basis by, for each new vector, subtracting the projection along all previous vectors. Note that the order of the basis matters in this orthonormalization process. We use the order that the basis vectors were introduced in.

To examine and evaluate the manner in which orthogonalization affects the syntactic basis, we visualize the cosine similarity between syntactic basis vectors before orthogonalization in Figure 3.1. Vectors created through word-pair differences (method 4) have low correlation with each other, while the part-of-speech vectors have high correlation with each other. In particular, verbs, adjectives and adverbs are positively correlated with each other and negatively correlated with nouns. High correlation between part-of-speech vectors is not a problem—the preliminary method of construction does not imply or require that the vectors are independent. But the concepts represented by these vectors are independent, and disentangling these concepts likely improves the fidelity between the vector and the concept.

A good deal of this correlation, especially the positive correlation between vectors that intuitively should not be associated, stems from correlation with the common discourse vector (demonstrated in Figure 3.2). This observation reveals the benefits of the common discourse vector.

Finally, we subtract the projection along the syntactic basis vectors from all other ('semantic') basis vectors we use and renormalize them. This procedure separates the syntactic meaning from our semantic basis vectors, ensuring that semantic bases are not also coding for syntactic concepts. We find that this procedure does not remove more than 50% of the length of any individual vector, and 50% of vectors have less than 20% of their length removed.

When encoding a dense vector, instead of finding the syntactic coefficients using sparse

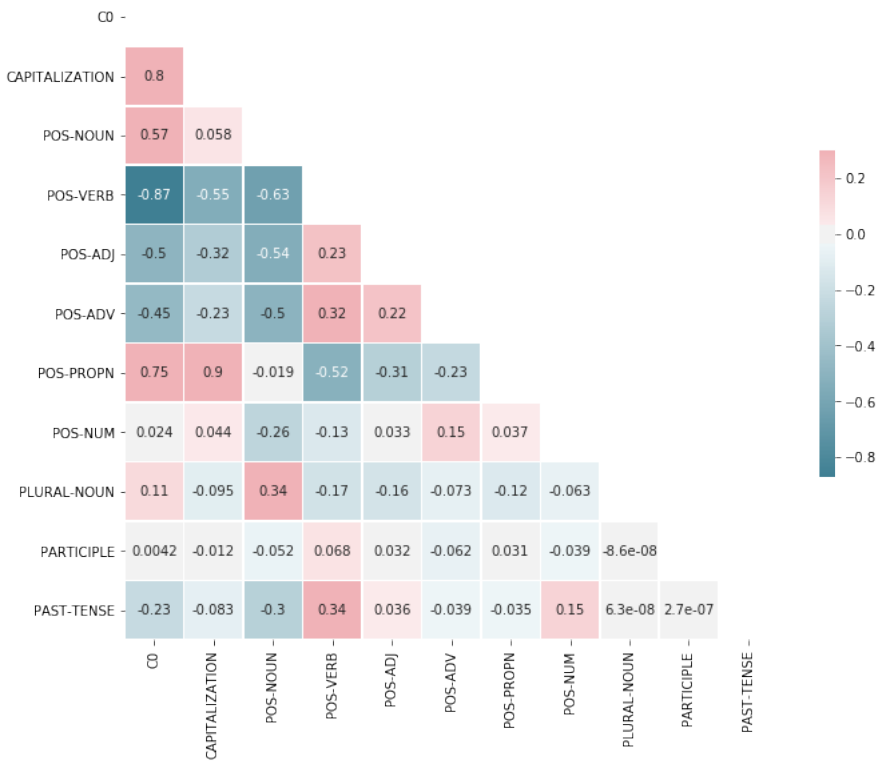

Figure 3.1: The pairwise cosine similarity between syntactic basis vectors (before orthogonalization). For example, capitalization correlated heavily with the proper noun part-of-speech. Note that the sign of the c_0 basis vector, which is the result of principal component analysis, is arbitrary.

.

coding, we set each syntactic coefficient to the projection along the corresponding syntactic basis vector, which is equal to the dot product similarity between the original vector and the syntactic basis vector. Because the syntactic basis is orthogonal, we can do this for every syntactic basis vector simultaneously. This residual is then transformed using Equation 3.1.

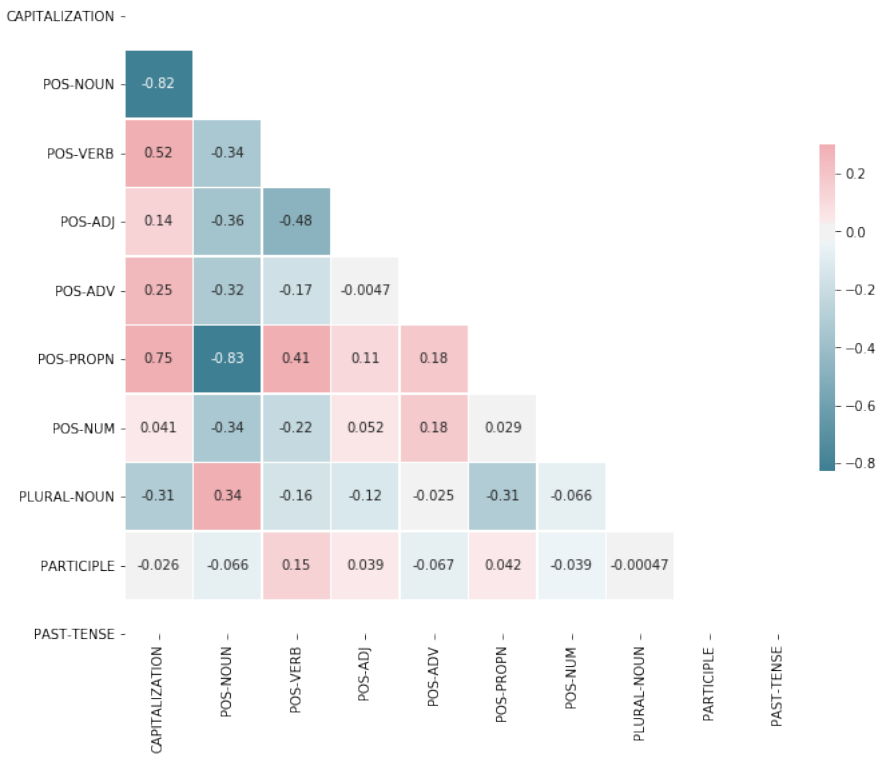

Figure 3.2: The pairwise cosine similarity between syntactic basis vectors after orthogonalizing only with respect to the common discourse vector c_0. Recall that c_0 is found by taking the first principal component of the entire word embedding space.

3.4 Basis Selection

We cannot practically use all words as our basis set, so we have to be selective. Note that our basis is *overcomplete*, with more elements than there are dimensions in the original space, and is not a basis by the standard linear algebra definition. The goal in basis selection is not to find a set of basis vectors that can represent every point in the space. Indeed, almost every set of random vectors with the same size can do this. Instead, we need an overcomplete basis that can approximate most vectors by a sparse linear combination.

First, we start with the 30,000 most frequent words, taken from the source of our dense word

embeddings (in this work, pretrained FastText dense vectors [2]). We filter out any words that are capitalized or that are not in a standard English vocabulary (using the vocabulary of the spaCy *en_core_web_sm* model). Next, we filter out any words that are not nouns, verbs, or adjectives. This process removes many basis vectors that may be hard to interpret. This gives us approximately 11,000 remaining potential basis words. From these, we will select 3,000 words to use in the final basis. We choose this basis size for comparison with previous work.

Vectors in this basis must fulfill two simultaneous criteria. They must be close to other vectors in the original space while being dissimilar from each other. Given the original set of vectors V_D, we select a set of basis vectors, B, from the set of vectors that pass the above filter, F, using the following algorithm, *guided basis selection*:

Guided Basis Selection Algorithm

B $\leftarrow$ Syntactic Basis

while $|B| < 3,000$ **do**

x $\leftarrow \arg\max_{x \in \mathcal{F} \setminus B} \sum_{v \in V_D} (x \cdot v) \max_{b \in B}(1 - b \cdot v)$

$B \leftarrow B \bigcup x$

end while

Intuitively, this equation takes the potential basis vector with the highest mean cosine similarity to all other vectors. To encourage diversity, this mean is weighted by the lowest cosine dissimilarity that each vector has with any already-selected basis vector.

3.5 Summary

We have described a procedure for transforming a dense vector space into an inherently interpretable sparse vector representation. The core idea of this transformation is to select some number of syntactic and semantic *basis vectors* and use *sparse coding* to recode all vectors as a sparse linear combination of these basis vectors. Each basis vector we use

has an inherent interpretation, either because it is a word embedding itself or because it represents a specific syntactic concept. We now analyze the efficacy of these transformed embedding.

Chapter 4

Results

We will evaluate our embeddings in multiple ways. We care about two contradictory properties of our transformed vector space. First, we want our vector space to be useful in downstream machine learning applications. We expect that, in most applications, increased interpretability comes with some performance cost. Therefore, we care about the performance loss when moving from dense vectors to our sparse vectors.

The other goal is that our sparse vectors should be interpretable. It is much harder to articulate exactly what interpretability is or how we can measure it. Metrics such as the Word Intrusion Task can act as a useful proxy for interpretability, and we use it as our primary quantitative measure of interpretability. But part of interpretability is, by definition, subjective. Therefore we will also attempt to demonstrate the interpretability of our vectors through qualitative means. In this chapter, we analyze the qualitative properties of interpretability by analyzing the behavior of classifiers trained on our sparse embeddings and showing that these classifiers are interpretable. Later, we will directly analyze sparse word embeddings and qualitatively consider their interpretability.

4.1 Preprocessing and Implementation Details

We use the FastText [2] pretrained 300 dimensional English vectors (without subword information) trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset as the dense vectors that we input into our models. Unless otherwise mentioned, we only consider the 30,000 most frequent words, for computational reasons. We normalize all vectors to have mean zero and unit length. After learning sparse vectors, we normalize each sparse vector so that it corresponds to a dense vector of unit length.

In practice, the sparse penalty term will only push coefficients very close to zero. We clamp any coefficient with a magnitude of less than .001 to zero. We found this threshold by taking the lowest cutoff that does not cause an uneven shape in the tradeoff curves in Chapter 4.4.

4.2 Implementation

We solve the regularized optimization problems using the FISTA algorithm [3], as implemented in the Python Lightning package [1], using default hyperparameters. FISTA is an optimization algorithm that can efficiently solve sparse coding problems. We use the spaCy library [15] to check for out of vocabulary words and perform part-of-speech tagging. We use the numpy [35], CuPy [34], and Scikit learn [38] libraries for various linear algebra implementations. We use the open-source Gensim library [40] to manipulate word embeddings.

4.3 Comparison with Previous Work

To compare our work against other sparse coding approaches, we will often reference the vectors created by Faruqui et al. [9]. That work generates more interpretable vectors using sparse coding but without inherently interpretable dimensions. In comparisons, we use

their publicly available implementation with the following settings: We use the same input vectors without preprocessing, a dimensionality of 3,000, L_2 regularization penalty $\tau = 10^{-5}$, as suggested in their paper, and various L_1 regularization penalties (λ).

4.4 Reconstruction Error and Sparsity

Recall that for any word represented by a dense vector v_D our method finds the sparse vector v_S that minimizes the following objective function:

$$\arg\min_{v_S} \ (||v_D - v_S\mathcal{B}||_2^2 + \alpha||v_s||_1) \tag{4.1}$$

where $\mathcal{B}$ is the basis matrix.

Note that, despite the fact that $\mathcal{B}$ spans the entire dense space, v_D is not equal to $v_S\mathcal{B}$. We refer to $\bar{v}_S\mathcal{B}$ as the *reconstructed vector*. This vector is the vector in the original dense space that is represented by our new sparse vector. We refer to the difference between the original vector and the reconstructed vector as the *reconstruction error*. This error is a likely source of any degradation in downstream tasks when switching to our transformed vectors. Taking the derivative of Equation 4.1, with respect to the coefficients of v_s, shows that this reconstruction error depends on α, the coefficient of sparsity. Therefore, we expect a tradeoff between sparsity and reconstruction error.

This tradeoff curve is displayed in Figure 4.1. Measuring this tradeoff gives a useful task-agnostic measure of performance. Comparing to the sparse coding of Faruqui et al., we can see that, despite the additional constraints of an inherently interpretable system, we suffer only a minor increase in reconstruction error compared to traditional sparse coding.

For the remainder of this work, unless otherwise mentioned, when we display vector representations we will display those made with $\alpha = 0.35$, for readability. These vectors have,

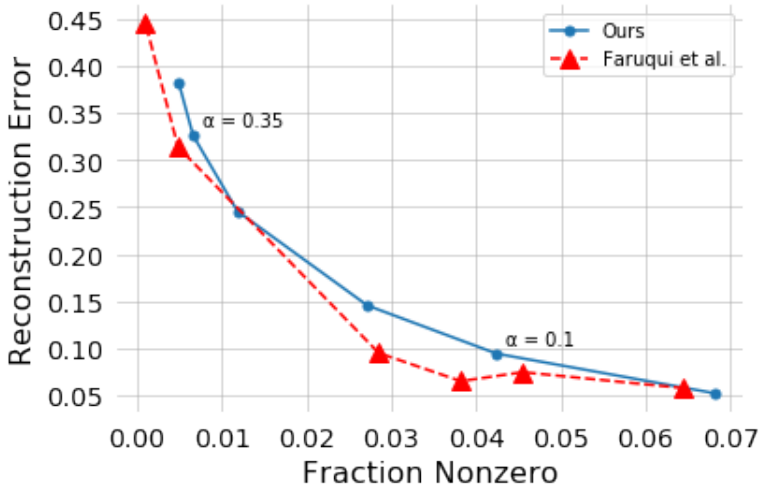

Figure 4.1: The tradeoff curve between sparsity and reconstruction error. Sparsity is measured as the mean fraction of coefficients that are nonzero in the sparse vectors. Reconstruction error is measured by cosine dissimilarity between dense and reconstructed vectors. The solid line shows the tradeoff curve achieved by our models. The dashed line shows the tradeoff curve achieved by Faruqui et al. using sparse coding without inherently interpretable dimensions (see Chapter 4.3).

on average, 20 nonzero entries. When comparing across different levels of sparsity, we will frequently compare vectors made with $\alpha = 0.1$ and vectors made with $\alpha = 0.35$. We will return to the concept of reconstruction error and analyze it in much greater depth in the next chapter.

4.5 Analogy Task

The Mikolov et al. [31] paper detailing word2vec measured the performance of their system by showing that it could successfully complete *word analogy tasks*. In the word2vec vector space, famously, the vector for 'king' plus the vector for 'woman' minus the vector for 'man' is close to the vector for 'queen'. Analogy tasks quantitatively test these properties.

The task consists of analogies of the form A *is to* A' *as* B *is to* B'. The vector space is evaluated on its ability to correctly determine the value of B'.

We use the analogy evaluation library provided by Gensim [40], also known as the *3CosAdd* method, which closely mirrors the original word2vec implementation. This method performs the vector operation $B + A - A'$, finds the closest vector by cosine similarity (Excluding A, B, or A' themselves), and returns the corresponding word as the guess for B'.

The performance of our vector space at this task is displayed in Table 4.1. A further breakdown can be found in Tables 4.2 and 4.3. We used the word2vec analogy test set [31]. We randomly divided the examples in the provided set into a development set and a test set. We looked at only the development set during exploratory development, and report results on the test set. In general, our model performs poorly on this task. This degradation comes from two sources. First, the drop from the original vectors to the reconstructed vectors due to reconstruction error. Second, an additional degradation is caused by the transformation from dense vectors to sparse vectors. The method used for the analogy task is based on cosine similarity, which is noisy with sparse vectors.

	Nonzero	Total	Syn.	Sem.
FastText (centered)	300	0.88	0.85	0.94
Faruqui $\lambda = 0.75$	136	0.65	0.60	0.73
Ours $\alpha = 0.1$	127	0.50	0.46	0.59
Ours Recons. $\alpha = 0.1$	300	0.83	0.80	0.88
Ours $\alpha = 0.35$	20	0.20	0.22	0.15
Ours Recons. $\alpha = 0.35$	300	0.33	0.38	0.25

Table 4.1: Accuracy on the word2vec analogy evaluation set for various vector spaces. The first column shows the average number of nonzero entries in each sparse vector. Accuracy is also broken down by semantic and syntactic categories. All results are on a 50% held-out test set. For comparison with the original dense vectors (FastText [2]), we subtract the mean of all vectors, to match preprocessing.

	Syn.	Adv.	Opp.	Comp.+Super.	Nationality	Tense	Plural
FastText (centered)	0.85	0.37	0.63	0.95	1.00	0.90	0.91
Faruqui $\lambda = 0.75$	0.60	0.13	0.39	0.67	0.93	0.56	0.67
Ours $\alpha = 0.1$	0.46	0.13	0.15	0.44	0.90	0.60	0.32
Ours Recons. $\alpha = 0.1$	0.80	0.38	0.50	0.91	0.97	0.84	0.84
Ours $\alpha = 0.35$	0.22	0.07	0.05	0.25	0.30	0.29	0.17
Ours Recons. $\alpha = 0.35$	0.38	0.16	0.08	0.40	0.42	0.43	0.41

Table 4.2: A detailed breakdown of the syntactic portion of Table 4.1 into subcategories. These subcategories are, in order, adjective-to-adverb, opposites, adjective comparatives and superlatives, nationality adjectives, past tense, and plurality of nouns and verbs. Some of these subcategories are the combination of subcategories from the original word2vec analogy set, for ease of display.

	Sem.	Capital	Currency	Cities in States	Family
FastText (centered)	0.94	0.99	0.23	0.89	0.95
Faruqui $\lambda = 0.75$	0.73	0.89	0.05	0.58	0.66
Ours $\alpha = 0.1$	0.59	0.72	0.09	0.50	0.20
Ours Recons. $\alpha = 0.1$	0.88	0.98	0.09	0.79	0.83
Ours $\alpha = 0.35$	0.25	0.36	0.00	0.10	0.35
Ours Recons. $\alpha = 0.35$	0.25	0.36	0.00	0.10	0.35

Table 4.3: A detailed breakdown of the semantic portion of Table 4.1 into subcategories. These subcategories are, in order, capitals of countries, currencies of countries, cities in states, and family terms. Some of these subcategories are the combination of subcategories from the original word2vec analogy set, for ease of display.

Our vectors show a massive degradation from the input vectors and a significant degradation from comparable sparse coding vectors. Looking at the breakdown of this gap, especially for syntactic tasks, shows considerable improvement among dimensions that we implicitly represent with syntactic basis vectors. We match or exceed the performance of the standard sparse coding vectors at adjective-adverb relationships or tense relationships, but fall very short in dimensions such as degree of adjectives or the number of verbs. We did not add these dimensions to avoid over-fitting to this task, but this result suggests that these constructed dimensions are helpful and that future work using our system can easily increase performance by adding relevant syntactic basis vectors.

4.6 Classification

Next, we demonstrate that our model can be used to build interpretable machine learning systems. To this end, we train classifiers using our word embeddings as input. We

demonstrate that these classifiers are not only effective but also interpretable.

We evaluate our vectors on two datasets, the IMDB sentiment analysis dataset [27] and the TREC question classification dataset [23]. For both of these datasets, we use the same model. Both datasets involve reading in English text and placing it into one of several categories. We use a bag-of-words approach that adds together the vector representations of each word in the sentence, ignoring capitalization. Then we train a logistic regression model on this fixed-length representation of each sentence. The logistic regression model is regularized using a L_2 penalty, which is found by cross-validation on the training set. We use the standard train splits and test splits for each dataset.

Though this preprocessing produces interpretable models, simple regression-based models are not able to perform complex tasks, especially tasks relying on word order, which is not preserved by the bag-of-words preprocessing. For these reasons, we carefully select the tasks on which we evaluate.

4.6.1 IMDB Sentiment Analysis Dataset

The IMDB movie review dataset consists of 50,000 passages taken from IMDB movie reviews, evenly split between positive and negative reviews. The task is to determine whether each passage expresses positive or negative sentiment about the movie it describes [27].

Table 4.4 presents sample positive and negative passages from the training set.

We train classifiers using various word embedding spaces as inputs. The results are presented in Table 4.5. Our vector spaces demonstrate improvement over the original dense vectors (FastText [2]), as well as the traditional sparse coding approach of Faruqui et al. This result holds despite a slight decrease in performance caused by the reconstruction error (as demonstrated by the low performance with reconstructed vectors).

Sentence	Label
An old family story told to two young girls by their grandfather is brought to life 16 years later as he foretold. People are getting murdered and blood is being spilled and rats are scampering all over and naked bodies are being enjoyed. Kitty (Barbara Bouchet) is the suspect, but we know she is not the killer. Is it Franziska (Marina Malfatti)? Is it Evelyn back from death for revenge? Is it a plot to steal an inheritance? The color is superb in this thriller from Emilio Miraglia, who only did one other Giallo, as far as I know. The only thing that spoiled the film was the appearance that several frames were cut out. Someone calls the police, and suddenly they are there trying to save Kitty.	Positive
From the beginning of the movie, it gives the feeling the director is trying to portray something, what I mean to say that instead of the story dictating the style in which the movie should be made, he has gone in the opposite way, he had a type of move that he wanted to make, and wrote a story to suite it. And he has failed in it very badly. I guess he was trying to make a stylish movie. Any way I think this movie is a total waste of time and effort. In the credit of the director, he knows the media that he is working with, what I am trying to say is I have seen worst movies than this. Here at least the director knows to maintain the continuity in the movie. And the actors also have given a decent performance.	Negative

Table 4.4: Examples of positive and negative passages from the IMDB movie review dataset

	Nonzero Entries	Accuracy
FastText (centered)	300	85.35
Faruqui $\lambda = .75$	136	85.54
Ours $\alpha = 0.1$	127	**87.51**
Ours Recons. $\alpha = 0.1$	300	85.08
Ours $\alpha = 0.35$	20	86.46
Ours Recons. $\alpha = 0.35$	300	83.00

Table 4.5: Accuracy on the IMDB sentiment analysis dataset, using a logistic regression classifier, which uses as input a bag-of-words sum of various word embeddings.

Unlike a classifier trained on the original vector space, we can interpret our classifier's coefficients. Here, we present the most significant coefficients ($\alpha = 0.1$):

$$\begin{aligned}\ln \frac{P(\text{positive})}{1 - P(\text{positive})} = &-157 \cdot \text{dreadful} - 153 \cdot \text{horrible} + 150 \cdot \text{fabulous} - 140 \cdot \text{dull} \\ &+ -132 \cdot \text{dreary} - 107 \cdot \text{worsen} - 105 \cdot \text{ridiculous} + ...\end{aligned} \tag{4.2}$$

Note that these are not coefficients on the frequencies of individual words. Instead, these are coefficients on vectors in the basis set. We can consider them to be coefficients on concepts, which are labeled by the displayed words. The coefficients make sense: positive concepts have positive coefficients, while negative concepts have negative coefficients. This pattern continues for much longer than displayed above, and we have omitted other terms for space reasons. The first term to not fall into this clear interpretation is the 24th-most significant:

$$\ln \frac{P(\text{positive})}{1 - P(\text{positive})} = ... + 74 \cdot \text{shall} + ...$$

At first, this term appears nonsensical. Looking more closely at this dimension can reveal

more about our system. The top five words in the dimension represented by 'shall' are the following: 'henceforth', 'herein', 'hereafter', 'thereof', 'hereby'. We can see here how both our vector space and our regression model pick up on tone. This dimension appears to correspond to a formal and somewhat archaic tone, which is likely not found in a negative internet comment. This example demonstrates the potential for our interpretable vector space to reveal non-trivial details about how a classifier makes decisions.

4.6.2 TREC Question Classification Dataset

Our next classification task is more complex. The TREC question classification dataset consists of 6,000 questions that are divided into 6 categories based on the expected answer: abbreviations, descriptions, entities, humans, locations, and numeric. Examples of questions from each category are presented in Table 4.6.

Sentence	Label
What is the full form of .com?	Abbreviation
How did serfdom develop in and then leave Russia?	Description
What films featured the character Popeye Doyle?	Entity
Who killed Gandhi?	Human
Which two states enclose Chesapeake Bay?	Location
When was Ozzy Osbourne born?	Numeric

Table 4.6: Examples of questions from each category in the TREC question classification dataset.

Accuracy for various vector spaces is presented in Table 4.7

. Again, our model does better than the unmodified input vectors we start with, despite some loss from the reconstruction error. Both results suggest that our vector spaces are efficient in regression-based settings, even though the performance at the word-analogy task

	Nonzero Entries	Accuracy
FastText (centered)	300	84.2
Faruqui $\lambda = .75$	136	84.4
Ours $\alpha = 0.1$	127	**86.2**
Ours Recons. $\alpha = x$	300	81.4
Ours $\alpha = 0.35$	20	84.0
Ours Recons. $\alpha = 0.35$	300	75.8

Table 4.7: Mean accuracy on the TREC question classification dataset, using a logistic regression classifier, which uses as input a bag-of-words sum of various word embeddings.

suffers a serious degradation. It is likely that different qualities are needed for these different tasks. The exact-match evaluation of the word analogy task severely punishes even slight noise in the vector space, and cosine similarities are noisy in sparse vectors.

Once again, we directly interpret the coefficients learned by logistic regression.

$$\begin{aligned}\ln \frac{P(\text{ABBR})}{1-P(\text{ABBR})} &= 43 \cdot \text{abbreviated} + 38 \cdot \text{abbreviation} + 35 \cdot \text{acronyms} + 34 \cdot \text{ports} + 29 \cdot \text{acronym} \\ &+ 26 \cdot \text{prefixes} + 24 \cdot \text{top} - 22 \cdot \text{biggest} + 20 \cdot \text{regional} + 19 \cdot \text{inquiry} + 19 \cdot \text{stood} \\ &+ \cdot 19 \cdot \text{software} + 19 \cdot \text{subsequent} + 19 \cdot \text{clothes} - 18 \cdot \text{<POS-NUM>} \\ &+ 17 \cdot \text{subcommittee} + 17 \cdot \text{mainframe} + 16 \cdot \text{particular} + 16 \cdot \text{wrong} + ...\end{aligned} \tag{4.3}$$

$$\begin{aligned}\ln \frac{P(\text{DESC})}{1-P(\text{DESC})} &= 45 \cdot \text{explained} + 36 \cdot \text{wonder} - 33 \cdot \text{most} - 32 \cdot \text{pretty} + 30 \cdot \text{derivation} \\ &- 29 \cdot \text{thick} - 27 \cdot \text{fearing} + 24 \cdot \text{effect} + 23 \cdot \text{rally} - 23 \cdot \text{innumerable} \\ &+ 22 \cdot \text{justification} + 20 \cdot \text{concerning} + 20 \cdot \text{imply} + 20 \cdot \text{coming} + ...\end{aligned} \tag{4.4}$$

$$\ln \frac{P(\text{ENTITY})}{1-P(\text{ENTITY})} = -35 \cdot \text{whom} + 32 \cdot \text{animals} - 27 \cdot \text{nation} + 27 \cdot \text{humans} + 25 \cdot \text{fearing} - 24 \cdot \text{explained} + 24 \cdot \text{birds} - 23 \cdot \text{imply} - 23 \cdot \text{wonder} - 23 \cdot \text{companies} + 22 \cdot \text{diseases} - 22 \cdot \text{town} - 22 \cdot \text{stood} + 21 \cdot \text{sequels} + 21 \cdot \text{suffered} + ... \tag{4.5}$$

$$\ln \frac{P(\text{HUM})}{1-P(\text{HUM})} = -77 \cdot \text{wonder} + 66 \cdot \text{organizations} + 54 \cdot \text{companies} + 51 \cdot \text{poet} + 49 \cdot \text{songwriter} + 48 \cdot \text{identities} + 42 \cdot \text{fan} - 42 \cdot \text{movie} - 39 \cdot \text{resulting} + 36 \cdot \text{university} - 36 \cdot \text{diseases} + 36 \cdot \text{successive} + 35 \cdot \text{consist} + 35 \cdot \text{cabinets} + ... \tag{4.6}$$

$$\ln \frac{P(\text{LOC})}{1-P(\text{LOC})} = 49 \cdot \text{nation} + 37 \cdot \text{locations} + 33 \cdot \text{instances} + 29 \cdot \text{city} + 28 \cdot \text{town} + 27 \cdot \text{metropolis} + 27 \cdot \text{headquarters} + 25 \cdot \text{webpage} - 24 \cdot \text{wannabe} + 24 \cdot \text{ethnicity} + 24 \cdot \text{islands} + ... \tag{4.7}$$

$$\ln \frac{P(\text{NUM})}{1-P(\text{NUM})} = -123 \cdot \text{explained} + 117 \cdot \text{innumerable} + 97 \cdot \text{yr} + 97 \cdot \text{most} + 90 \cdot \text{pretty} + 88 \cdot \text{thick} + 84 \cdot \text{young} - 83 \cdot \text{surname} + 72 \cdot \text{deal} - 70 \cdot \text{whom} + 68 \cdot \text{autumn} + 64 \cdot \text{expense} + 64 \cdot \text{percentage} - 62 \cdot \text{barring} + 62 \cdot \text{fact} + 62 \cdot \text{amount} + ... \tag{4.8}$$

Because of the increased complexity of the categories, interpreting these coefficients is both difficult and more rewarding than interpreting the coefficients for the IMDB dataset. We analyze and attempt to provide an explanation for some of the more unintuitive terms:

- 'rally' is associated with political and economic words, and is likely associated with the Entity category because there are many description questions that ask about these issues.

- ‘fearing’ is associated with many historical concepts, which is likely why it is associated with the Entity category.
- ‘cabinets’ is likely associated with the Human category because of its second, political, sense.
- ‘autumn’ is associated very heavily with many words representing years, which is likely why it is associated with the Numeric category.

Interpretation fails on some coefficients. For example, ‘biggest’ is associated with many abbreviations, which is presumably the reason why it has a high weight in the abbreviation class, but there is no clear intuitive motivation for this association. Similarly, we are unable to find good justifications for the high weight for dimensions such as ‘successive’, ‘most’, or ‘particular’, among others. Some of these anomalies may be explained by overfitting since the dataset contains only 5,500 training examples. Nevertheless, these irregularities likely demonstrate defects in our system. Even so, our system provides considerable insight into the behavior of a previously uninterpretable classifier trained on a non-trivial task.

Note that the presence of this analysis itself, as well as the analysis in Chapter 5, shows that we have not yet achieved perfectly inherently interpretable dimensions and that some of our dimensions still require manual interpretation. However, the majority of the dimensions displayed above do not need this manual interpretation. In addition, manual interpretation is easier with the provided labels.

4.6.3 Word Intrusion Task

To quantitatively measure interpretability, we use human experiments. In particular, we use the word intrusion task [4]. In this task, humans are presented with five words, four of which are associated highly with a particular dimension. Participants are asked to choose the word that does not belong. See Chapter 2.5 for more details about the framing of the task.

Figure 4.2: An example of the user interface given to annotators. The following instructions were given to the annotators: 'You will be presented with a group of 5 words. Four of these words are similar in some way and the other one is not. Pick out the word which is dissimilar. You may be provided with a hint about how the words are similar.'

We use the following procedure for generating questions. First, we filter candidate words, starting with the 20,000 most frequent words and filtering out words that are not lowercase, words that are not made up of only ASCII alphabetic characters, and words with only one letter. Then we randomly select a dimension. We pick the 4 highest words along that dimension, and one word randomly selected from the bottom 50% of words in that dimension, then randomize the order.

We use three types of vectors: the original FastText vectors, the baseline sparse coding vectors from Faruqui et al., our vectors with $\alpha = 0.1$.

We use our vectors both with and without providing the label of the dimension as a 'hint'. Each example is presented to three different Mechanical Turk annotators.

We excluded a small number of examples from one annotator because their accuracy and agreement with other annotators were significantly below that of the other annotators and close to the level expected from random chance.

An example of the user interface is shown in Figure 4.2. The results of the word intrusion task are presented in Table 4.8. In particular, when hints are provided, we see a statistically significant improvement in accuracy between our vectors and the sparse coding baseline ($p = .00055$). In addition, using hints produces a statistically significant improvement ($p = .040$),

	Accuracy	CI	Complete Accuracy	CI	Agreement	CI
FastText	0.31 (253/826)	[0.27,0.34]	0.12 (29/252)	[0.08,0.15]	0.43 (746/1,718)	[0.41,0.46]
Faruqui	0.77 (639,/829)	[0.74,0.80]	0.63 (157/249)	[0.57,0.69]	0.72 (1248/1,736)	[0.70,0.74]
Ours	0.80 (661/827)	[0.77,0.83]	0.67 (167/249)	[0.61,0.73]	0.76 (1,306/1,726)	[0.74,0.78]
Ours + Hints	0.84 (690/ 822)	[0.81,0.86]	0.70 (176/252)	[0.64,0.76]	0.78 (1,326/1,698)	[0.76,0.80]

Table 4.8: Results on the word intrusion task. Three metrics are shown for each vector space. *Accuracy* measures the fraction of questions in which the annotator correctly identified the intruder. *Complete accuracy* measures the fraction of questions in which all three annotators correctly identified the intruder. *Agreement* measures the chance that two annotators on the same question agree with each other. Each number is accompanied by a 95% normal confidence interval.

validating our motivation for inherently interpretable dimensions.

4.7 Summary

We have quantitatively evaluated the performance of our word embeddings at both interpretability and performance. We measured performance in a number of ways: We first measured the *reconstruction error*, or the noise caused by the sparsity inducing penalty, and find that our methods do not introduce substantially more noise than comparable sparse coding methods that learn an optimal basis matrix. Next, we measure the performance at word analogy tasks and find a significant loss in performance with our method. Finally, we use our embeddings as inputs to two classification tasks and find that the resulting models perform *better* than models trained on the original dense vectors while maintaining extremely interpretable coefficients. We then measure interpretability using the word intrusion task and find that our sparse vectors achieve better results than previous sparse coding approaches, when the labels of each basis dimension are provided.

Overall, our transformed word embeddings performs well. We find that our transformed word embeddings are more interpretable than that of previous work without a significant change in performance.

Chapter 5

Analysis

Overall quantitative measurements do not fully assess all the benefits or completely capture the behavior of our system. In particular, we want to understand the interpretability of our system. To address this shortcoming we will perform a variety of analyses in order to gain deeper insight into the behavior and performance of our system.

This chapter is broken down into seven sections. Each of the sections is a largely independent analysis of an aspect of our system using different techniques.

5.1 Ablation Analysis

The model we described in Chapter 3 included a number of design choices. While we made these choices based on exploratory experiments, we did not present any justification for them. In this section, we will explore the quantitative effects of a number of these design choices.

Throughout the rest of the section, we run modified versions of our system, changing one of the decisions we have described. For each change, we evaluate the modified system with three metrics: The mean reconstruction error, the performance on the analogy task, and the

	$\alpha = 0.1$			$\alpha = 0.35$		
	Reconstruction	Analogy	Classification	Reconstruction	Analogy	Classification
Base	0.91	0.50	0.88	0.67	0.20	0.86
Syntactic Removed	0.91	0.37 (-0.17)	0.87 (-0.01)	0.65 (- 0.02)	0.09 (- 0.11)	0.86
Sparse Syntactic	0.92 (+ 0.01)	0.51 (+ 0.01)	0.87 (- 0.01)	0.62 (- 0.05)	0.13 (- 0.07)	0.86
No Basis Filtering	0.91	0.50	0.87 (- 0.01)	0.67	0.20	0.86
Random Basis	0.91	0.50	0.87 (- 0.01)	0.67	0.18 (- 0.02)	0.87 (+ 0.01)
Max. Distinct Basis	0.91	0.54 (+ 0.04)	0.87	0.67	0.25 (+ 0.05)	0.86

Table 5.1: Changes in performance for various metrics under various modifications to our model. Higher is better for all metrics; the Reconstruction column measures reconstruction similarity (1 minus the reconstruction error). See the text for a full description of the modifications made.

performance on the IMDB sentiment analysis classification task. See the previous chapter for a full description of these measurements.

5.1.1 Syntactic Basis Removal

First, we investigate the overall effect of the syntactic basis vectors. Table 5.1 shows the results of removing the syntactic basis. We find that the syntactic basis does not affect performance in most tasks, and only improves on the analogy task, in which all versions of our system perform poorly.

5.1.2 Sparse Syntactic Basis

Recall that the syntactic coefficient for a basis vector is equal to the dot product of that basis vector and the word embedding. Here, we examine the consequences of determining the syntactic coefficients through sparse coding, instead of this strategy.

Table 5.1 shows the results of this change under 'sparse syntactic'. This change has little

effect at smaller levels of sparsity, but has a stronger negative effect at high levels of sparsity, likely because higher levels of sparsity will force syntactic coefficients to 0.

5.1.3 Basis Filtering

Recall that we adopt a number of heuristic filters to remove unintuitive potential basis vectors. Specifically, we remove all words that are not lowercase nouns, verbs, or adjectives that can be found in a standard dictionary. We expect that this choice causes a slight decrease in performance in exchange for a large increase in interpretability. Table 5.1 shows that the decrease in performance caused by this filter is minimal.

5.1.4 Basis Selection

Recall that we selected basis vectors, after the filter, through an algorithm, guided basis selection, that balances diversity of basis vectors with closeness to a large number of word embeddings. Here, we compare this algorithm against two alternative algorithms. The first alternative selects the basis vectors at random from the filtered set of basis vectors (*random basis selection*). The second alternative iteratively selects the potential basis vector that is the farthest away from the basis vectors we have selected so far (*maximally distinct basis selection*). Table 5.1 shows the quantitative results of these comparisons. The comparison to the random selection algorithm shows little change. However, a selection algorithm that maximizes diversity shows a modest performance increase at the analogy task. See Chapter 5.4 for background on why a more diverse basis set may be expected to perform better at analogy tasks.

This improvement is modest and only occurs in a task where all versions of our system perform badly and are unlikely to be used. Even so, we will examine the ablation experiment for the basis selection algorithm more deeply. We expect that the primary advantage of our

basis selection algorithm is that it produces more interpretable basis vectors. Our quantitative measures of interpretability require human annotators and therefore cannot easily be performed in the ablation analysis. Instead, we offer a rough qualitative measurement of interpretability through manually comparing the representations of a single word. We use the same word, 'swimming', that we use in other examples. We selected 'swimming' for this purpose before looking at its embedding in the modified system.

First, we present the representation of 'swimming' in our system with guided basis selection:

$$\begin{aligned}\text{swimming} = {} & 0.28 * \text{POS-VERB} + 0.28 * \text{water} + 0.26 * \text{walking} \\ & - 0.26 * \text{CAPITALIZATION} + 0.26 * \text{surfing} + 0.23 * \text{volleyball} \\ & - 0.23 * \text{PAST-TENSE} \ + 0.22 * \text{POS-ADJ} + 0.22 * \text{underwater} \\ & + 0.19 * \text{dolphins} + 0.17 * \text{PARTICIPLE} \ - 0.16 * \text{PLURAL-NOUN} \\ & - 0.16 * \text{POS-NOUN} + 0.15 * \text{diver} + 0.11 * \text{fish} \\ & + 0.10 * \text{synchronized} + 0.10 * \text{dancing} + 0.08 * \text{sailing} \\ & + 0.07 * \text{POS-NUM} + 0.07 * \text{POS-ADV} + 0.07 * \text{POS-PROPN} \\ & - 0.07 * \text{C0} + 0.04 * \text{waters} + 0.04 * \text{skiing} \\ & + 0.00 * \text{badminton} + 0.00 * \text{basketball} \end{aligned} \tag{5.1}$$

And the corresponding representation in the modified system with distinct basis selection:

$$\begin{aligned}\text{swimming} = {} & 0.54 * \text{pool} + 0.40 * \text{bathing} + 0.34 * \text{rowing} \\ & + 0.28 * \text{POS-VERB} - 0.26 * \text{CAPITALIZATION} - 0.23 * \text{PAST-TENSE} \\ & + 0.22 * \text{POS-ADJ} + 0.17 * \text{PARTICIPLE} \ + 0.16 * \text{surfing} \\ & - 0.16 * \text{PLURAL-NOUN} \ - 0.16 * \text{POS-NOUN} + 0.09 * \text{diver} \\ & + 0.07 * \text{POS-NUM} + 0.07 * \text{POS-ADV} + 0.07 * \text{POS-PROPN} \\ & - 0.07 * \text{C0} + 0.05 * \text{volleyball} + 0.02 * \text{hiking}\end{aligned}$$

$$+ 0.01 * \text{lakes} \tag{5.2}$$

The strong basis filters ensure that the basis vectors selected by the maximal distinct algorithm are still reasonable. We see that the representations produced by the maximally distinct vectors tend to use words that are farther in meaning from the original word. These basis vectors are not necessarily incorrect, and prioritizing distinct syntactic basis vectors may be desirable for some systems. However, we find it likely that the behavior of our original system is useful in a wider range of circumstances and we will, therefore, continue to discuss that system.

Overall, most of the design choices have little impact on performance, though we expect these choices have a much greater impact on interpretability

5.2 Manual Analysis of Individual Basis Dimensions

To what extent do basis dimensions, both syntactic and semantic, actually capture meaningful properties of words? And to what extent do these properties align with the word that labels that dimension? In this section, we attempt to answer these questions through a manual analysis of interesting basis dimensions.

5.2.1 Syntactic Basis Dimensions

We examine a small but crucial set of basis dimensions: the syntactic dimensions that we manually constructed.

First, we examine the aggregate distribution of syntactic basis coefficients. Figure 5.1 shows the distribution of syntactic basis coefficients for words that do and do not have the features encoded by each dimension. We can see that, in most cases, the syntactic basis dimension can correctly encode the feature in question. This encoding, however, is not clean or discrete.

Note that the distinctiveness between the two groups increases as we move through the basis dimensions. Recall that the order is important because the basis vectors are orthogonalized in order, suggesting that this process leads to more distinct basis vectors once a greater number of previous vectors are taken into consideration.

Table 5.2: Words sampled randomly based on their syntactic basis coefficients. We sort the 3,000 most frequent words by each syntactic coefficient, and then sample from intervals that are 1 percentile wide.

Basis	**0%**	**25%**	**50%**	**75%**	**99%**
C0	introduced	ban	but	So	Hill
	rejected	same	vote	regime	WikiProject
	compared	forward	See	clients	Party
	placed	runs	methods	temperature	Putin
	led	recent	role	parties	Labour
CAPITALIZATION	items	features	longer	Bank	My
	methods	victory	College	think	Of
	buildings	baby	Christmas	believe	Their
	jobs	experiment	Syria	at	She
	properties	program	today	increase	Three
POS-NOUN	48	noted	screen	policy	Division
	58	oil	farm	Thanks	Company
	41	public	Hi	won	icon
	43	Two	territory	produce	Award
	36	learned	score	statistics	Project

Basis	**0%**	**25%**	**50%**	**75%**	**99%**
POS-VERB	long-term	Mary	candidates	check	showed
	Many	hope	you	Microsoft	took
	basic	atmosphere	radio	surgery	represented
	August	solution	here	doesn	earned
	several	policies	press	film	sent
POS-ADJ	2009	strategies	solutions	possible	disruptive
	1994	Each	yourself	neighborhood	Indian
	1986	findings	mission	buildings	industrial
	2014	accept	Last	Library	conservative
	2006	notes	work	island	Catholic
POS-ADV	57	m	1st	sentences	Because
	34	arms	law	notability	Though
	56	academic	Don	customers	They
	47	determined	modified	against	Most
	32	River	location	felt	somewhat
POS-PROPN	For	flag	powers	matter	anyone
	Many	references	results	properly	Lewis
	An	elements	former	represented	Clinton
	Before	hits	potential	move	Jim
	Each	service	types	remain	Cameron
POS-NUM	you	final	visitors	volume	46
	WP	their	had	multiple	49
	Maybe	tone	across	city	48
	During	going	cases	Roman	51
	wikipedia	all	connection	contemporary	37

Basis	**0%**	**25%**	**50%**	**75%**	**99%**
PLURAL-NOUN	figure tool job suggestion agent	brand toward earth executive subject	left special ‘ summer speaking	00 lies contribs far shared	don employees firms editors families
PARTICIPLE	send sell appear decide allow	world indeed paragraph carried fish	anything St. city African pictures	ongoing ability accounts differences Alexander	developing improving helping paying giving
PAST-TENSE	determine make producing stop improving	traditional believe consumer protection re	background . 1990 Law 1989	limited 26 is probably Kim	managed founded conducted moved got

Table 5.2 characterizes each syntactic basis dimension by randomly sampling words based on their syntactic coefficients. At the 99th percentile, all dimensions exhibit a reasonable amount of consistency, and the words that we examine all fall into the categories represented by the syntactic dimension. The 75th percentile is much more inconsistent. In some cases, this is not a problem; for many of these categories, the 75th percentile should completely exclude words that are not in the category in question, because words related to the category comprise less than 25% of words. In other dimensions, such as capitalization, we can see positive and negative examples intermixed. In general, combined with Figure 5.1, we see that syntactic basis coefficients cannot cleanly separate syntactic features.

In most cases, the 99th percentile is coherent. In many cases, the words in the 0th percentile

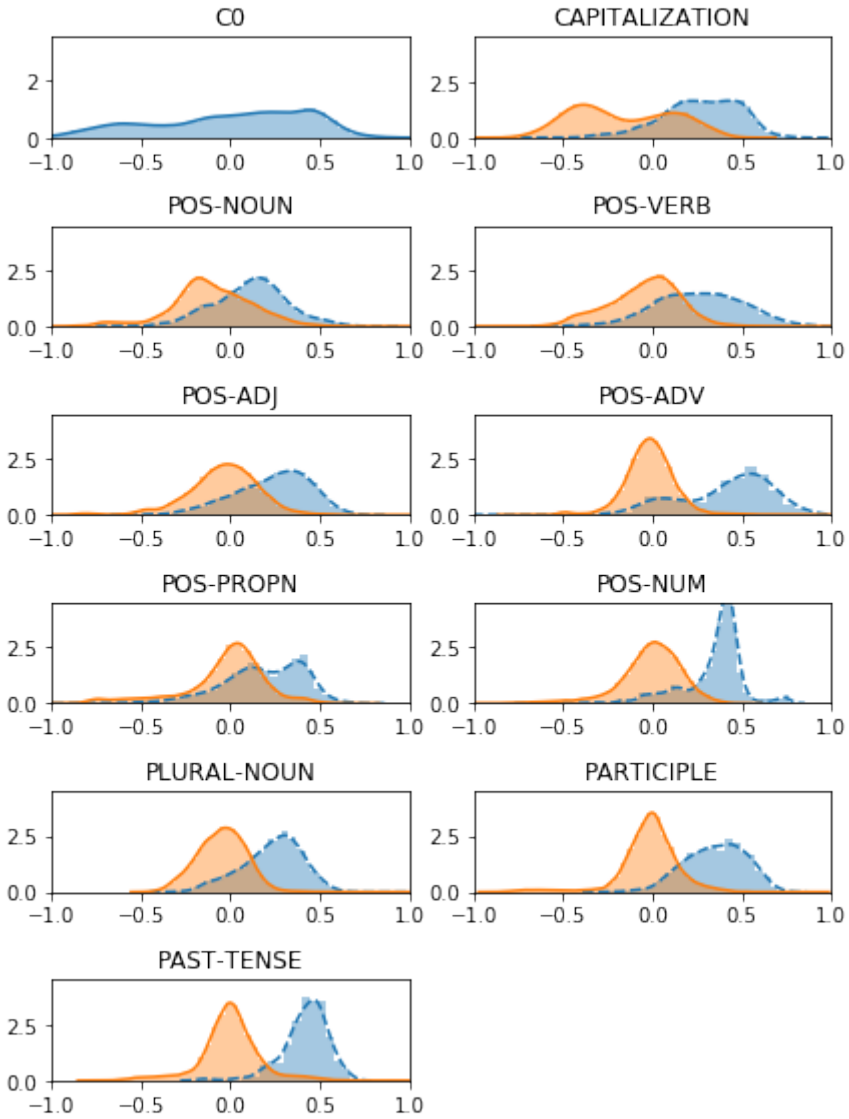

Figure 5.1: The distribution of syntactic basis coefficients across the entire vocabulary. The vocabulary is divided into two subpopulations, and colored accordingly, based on whether that word belongs to the category that the syntactic basis dimension attempts to model. Solid lines represent words that are in the positive subpopulation. These subpopulations are estimated using the spaCy library. Height of these plots is a kernel density estimate.

are conceptually related, though their relationship to the dimension is sometimes unintuitive. For instance, in multiple dimensions, the 0th percentile is filled with numbers. However, the dimensions created from word pairs all have coherent and sensible groups of words at the 0th dimension. For instance, present tense indicative verbs are in the 0th percentile of the participle dimension.

More concerning is a lack of clear distinction as to the meaning of a mid-range syntactic coefficient. There does not appear to be any clear distinction between the 25th and the 75th percentiles in any of the syntactic dimensions. Figure 5.1 shows that these intermediate ranges generally have low absolute coefficients, suggesting that they may not be that important. Recall that the syntactic coefficients are equal to the cosine similarity between the original dense vector and the syntactic basis vector. Even if these vectors are completely uncorrelated, there will be some non-zero cosine similarity due to noise, which causes non-zero syntactic coefficients.

5.2.2 Frequently Used Semantic Basis Vectors

Next, we will manually analyze some semantic basis dimensions. Because we cannot practically investigate every one of the 2,989 semantic basis dimensions, we sample basis dimensions that are unusual or interesting in various ways.

First, we investigate the most frequently used words for each value of α. On average, with $\alpha = 0.1$, each semantic basis vector is used to code for 1,167 different words. When $\alpha = 0.35$, each semantic basis vector is used in only 89 words on average. The distribution of the number of words for which each vector is used is approximately normal.

For $\alpha = 0.1$, the five most frequently used semantic basis vectors are ‘dos’, ‘ing’, ‘mater’, ‘shortcut’, and ‘la’. Of these, ‘dos’ and ‘la’ are used to code for foreign-language words, which show up in many contexts even in an English-language corpus. For instance, the words with the highest coefficient on ‘la’ are all European foreign language words for various determin-

ers. The basis ‘ing’ is associated with many short word-fragments, as well as numbers (see Chapter 5.5). The basis ‘mater’, as in the phrase ‘alma mater’, is used in the representation of many colleges. The basis ‘shortcut’ is used in the representation of many internet-related terms.

For $\alpha = 0.35$, the five most frequently used semantic basis vectors are ‘countries’, ‘namespace’, ‘page’, ‘imposes’, and ‘cruft’. ‘Countries’ is used to represent many countries, while ‘namespace’ is used for internet-related terms, and ‘page’ for terms related to Wikipedia metadata(which is likely to appear often in a dataset partially taken from Wikipedia). ‘Imposes’ differs from the basis vectors we have examined so far: It is used to represent many frequent verbs with somewhat-related meanings in a fairly ordinary way. For example, it is very heavily used in ‘carries’, ‘puts’, ‘limits’, and ‘introduces’. ‘Cruft’ is used to represent both many technical terms and many fantasy terms, as well as various acronyms.

Overall, most of these vectors can be considered an artifact of the filter in our basis selection process. Recall that we exclude potential basis vectors that are not part of a standard English vocabulary. Some words, such as ‘la’, pass this filter because they have a more obscure meaning, and then they can code for many words that have a relative scarcity of basis vectors. Other words, such as ‘namespace’, are standard English words that are related to many non-words such as ‘.com’.

5.2.3 Basis Vectors with Negative Coefficients

Unlike some sparse coding approaches, we allow both negative and positive coefficients. However, negative coefficients are much rarer than positive ones. When $\alpha = 0.35$, only 16% of the coefficients are negative. When $\alpha = 0.1$, there are more negative coefficients, but most of those are very small in magnitude.

Though there is no bias in our model against negative coefficients, the distribution of vectors in the dense vector space biases against negative coefficients. Despite the fact that we

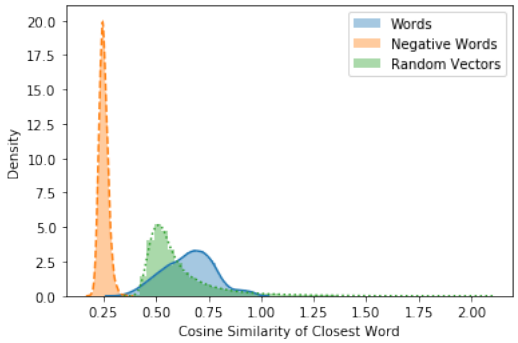

Figure 5.2: The distribution of distances to their closest neighbors. Shown for vectors corresponding to words (solid), the negation of vectors corresponding to words (dashed), and randomly distributed vectors (dotted). The randomly distributed vectors were sampled from a Gaussian distribution.

centered the vector space so that it has a mean of 0, vectors are still unevenly distributed and clustered in the dense space. Figure 5.2 shows that words are closer to their nearest neighbor in the vector space than would be expected by random chance while the negations of words are much farther from any other words. This means that the negations of words make bad basis vectors in general.

The five words with the most negative coefficients for $\alpha = 0.35$ are the following: 'which', 'province', 'pretty', 'manner', and 'apartheid'. 'which' is often used as one of the only semantic basis vectors to represent words such as 'Not', 'Even' or 'Used' (note the capitalization), combined with many large syntactic coefficients. 'province' codes for the names of many celebrities, though there is no obvious semantic reason for this. Similarly, 'apartheid' is used to represent various common European last names. 'pretty' with a negative coefficient codes for generic verbs and adverbs that cannot describe objects, such as 'Please', 'Occasionally', and 'Often'. 'manner' with a negative coefficient codes for many large numbers for no easily discernable reason (see Chapter 5.5).

5.2.4 Basis Vectors that were Selected Early

Recall that we use an iterative process to select basis vectors, suggesting that the 'best', or the most broadly applicable, basis vectors may be chosen first. This fact does not necessarily imply that vectors that are chosen earlier are more useful. In fact, the opposite appears to be true: When $\alpha = 0.1$, vectors chosen last are significantly more likely to be used more often. We measure this relationship with a linear regression. The r^2 value of this linear regression, or the percentage of variance in usage of basis vectors explained by this association, is $r^2 = .25$. The p-value, or the probability that a similarly strong result would occur by chance if there is no true relationship, is $p < 1e - 199$. When $\alpha = 0.35$, there is a slight association in the opposite direction ($r^2 = 0.02,\ p < 2e - 17$).

Though there is no clear relationship in frequency of usage, the first vectors to be selected do appear to be remarkably intuitively good concepts to use as a basis. The first ten selected basis vectors are 'beautiful', 'young', 'murdering', 'government', 'gorgeous', 'leaders', 'optimized', 'clergyman', 'antics', and 'charming'. Furthermore, all of these dimensions are used in extremely intuitive ways. For instance, 'beautiful' is used to represent 'attractive', 'scenic', and 'Marilyn' (the latter presumably refers to Marilyn Monroe). 'young' is used to represent various age-related terms, 'murdering' is used to represent various crimes, and the remainder of the first basis vectors have similarly intuitive uses.

5.3 An Analysis of Individual Word Representations

In the previous section, we examined and sought to understand individual dimensions by looking at their associated words. In this section, we reverse this procedure. We randomly select ten words, examine their entire representation (in the $\alpha = 0.35$ vector space), and holistically evaluate this representation.

In particular, we will focus on the coefficients that may be initially unintuitive, and attempt

to find an explanation for any unintuitive behavior of our system.

5.3.1 Carbon

$$\begin{aligned}\text{carbon} = {} & 0.79 * \text{nitrogen} - 0.38 * \text{CAPITALIZATION} + 0.30 * \text{fossil} \\ & - 0.21 * \text{POS-NOUN} + 0.16 * \text{POS-ADJ} + 0.14 * \text{C0} \\ & - 0.14 * \text{PAST-TENSE} + 0.13 * \text{wood} + 0.11 * \text{global} \\ & + 0.10 * \text{atoms} - 0.09 * \text{POS-ADV} + 0.09 * \text{aluminum} \\ & - 0.08 * \text{PLURAL-NOUN} + 0.07 * \text{greenhouse} - 0.07 * \text{POS-PROPN} \\ & - 0.05 * \text{POS-VERB} + 0.05 * \text{forestry} + 0.03 * \text{PARTICIPLE} \\ & + 0.02 * \text{sink} + 0.01 * \text{POS-NUM} \end{aligned} \tag{5.3}$$

The most unintuitive aspect of this representation is the syntactic coefficients, particularly the part-of-speech coefficients. Though the first entry in the Oxford English Dictionary lists 'carbon' as a noun, our system seems to consider 'carbon' to be an adjective. Of course, 'carbon' has a secondary use as an adjective (such as its use in the phrase 'carbon dioxide'). Given the news-focused dataset the dense vectors are trained on, this is likely a frequent use of the word. Here, it is worth noting that none of the part-of-speech coefficients is particularly high in magnitude (compare to the distribution shown in Figure 5.1). The best interpretation may be that the system is unsure of the correct part-of-speech for a word that is often used as both a noun and an adjective.

The semantic coefficients represent 'carbon' as it is used in practice. 'nitrogen', 'aluminum' and 'atoms' represent the chemistry-related aspect. Though carbon is clearly not a combination of nitrogen and aluminum, the coefficients nevertheless display some limited knowledge of chemistry: carbon is chemically much closer to nitrogen than to aluminum, and nitrogen has a much higher coefficient. Other basis dimensions represent not the meaning of 'carbon' itself but other contexts in which the word 'carbon' is used (such as 'fossil', 'wood', and

'greenhouse'). Note that some of the nonzero basis dimensions are associated with 'carbon' through their secondary meanings, such as 'global' (as in 'global warming') and 'sink' (as in 'carbon sink').

5.3.2 Reefs

$$\begin{aligned}\text{reefs} = {} & 0.68 * \text{islands} - 0.66 * \text{CAPITALIZATION} + 0.40 * \text{C0} \\ & + 0.35 * \text{PLURAL-NOUN} + 0.28 * \text{POS-VERB} + 0.25 * \text{rocks} \\ & + 0.19 * \text{dredging} + 0.18 * \text{oysters} + 0.12 * \text{POS-ADJ} \\ & + 0.10 * \text{POS-NUM} + 0.10 * \text{POS-ADV} + 0.09 * \text{POS-PROPN} \\ & + 0.09 * \text{tropical} + 0.08 * \text{underwater} + 0.07 * \text{dunes} \\ & + 0.06 * \text{seas} + 0.06 * \text{diver} - 0.06 * \text{PAST-TENSE} \\ & + 0.04 * \text{sandstone} - 0.02 * \text{demon} + 0.02 * \text{marine} \\ & - 0.02 * \text{PARTICIPLE} - 0.01 * \text{POS-NOUN} - 0.01 * \text{french} \\ & - 0.00 * \text{witches} \end{aligned} \tag{5.4}$$

The syntactic coefficients around part-of-speech are, again, ambiguous. In particular, 'reefs' has a moderate coefficient for 'POS-VERB' and an almost-zero coefficient for 'POS-NOUN'. However, there is no clear satisfying reason for this ambiguity.

The semantic coefficients are even more unintuitive. The most significant basis dimensions are generally intuitive, mostly focusing on water-related concepts. 'dredging' is a word often used in the sense of 'dredging' coral reefs. However, there is a series of less-significant basis dimensions, including 'demon', 'french', and 'witches' that have little discernable connection to 'reefs'.

5.3.3 Coulson

The next randomly selected word, 'Coulson', is a proper noun referring to a character from the Marvel cinematic universe:

$$\begin{aligned}\text{Coulson} = {} & 0.85 * \text{hacking} + 0.72 * \text{C0} + 0.59 * \text{POS-PROPN} \\ & + 0.25 * \text{CAPITALIZATION} - 0.23 * \text{POS-ADJ} - 0.19 * \text{POS-NOUN} \\ & + 0.17 * \text{butler} - 0.17 * \text{southern} + 0.15 * \text{POS-VERB} \\ & - 0.14 * \text{website} - 0.12 * \text{com} - 0.12 * \text{roaring} \\ & + 0.10 * \text{solicitors} - 0.09 * \text{POS-ADV} + 0.07 * \text{oats} \\ & - 0.07 * \text{cathedral} - 0.06 * \text{PARTICIPLE} + 0.06 * \text{inquiry} \\ & - 0.06 * \text{dances} - 0.06 * \text{fan} + 0.04 * \text{POS-NUM} \\ & - 0.03 * \text{provinces} - 0.03 * \text{finals} - 0.02 * \text{dance} \\ & - 0.02 * \text{waters} - 0.01 * \text{tango} - 0.01 * \text{shame} \\ & - 0.01 * \text{PAST-TENSE} - 0.00 * \text{PLURAL-NOUN} \end{aligned} \tag{5.5}$$

The representation for 'Coulson' is much noisier than previous representations we have examined. Though the most significant term, 'hacking', is somewhat appropriate, and the syntactic coefficients are broadly intuitive, the lower order semantic terms are very chaotic. This effect may be a result of the relatively low frequency of 'Coulson' in the corpus, which could cause the dense vector to be noisier.

5.3.4 Roundabout

$$\begin{aligned}\text{roundabout} = {} & 0.72 * \text{bypass} + 0.40 * \text{roadway} - 0.28 * \text{CAPITALIZATION} \\ & + 0.22 * \text{plaza} - 0.16 * \text{PLURAL-NOUN} + 0.11 * \text{POS-ADJ}\end{aligned}$$

$$
\begin{aligned}
&+ 0.11 * \text{clumsy} + 0.10 * \text{airfield} + 0.09 * \text{POS-ADV} \\
&- 0.08 * \text{biological} + 0.08 * \text{C0} + 0.05 * \text{POS-NOUN} \\
&+ 0.04 * \text{PAST-TENSE} \ + 0.04 * \text{PARTICIPLE} \ + 0.03 * \text{caravan} \\
&+ 0.02 * \text{ironic} + 0.02 * \text{POS-VERB} - 0.02 * \text{POS-NUM} \\
&- 0.00 * \text{POS-PROPN} + 0.00 * \text{nonsensical}
\end{aligned}
$$

Again, we have a word that is sometimes used as a noun and sometimes used as an adjective. Similarly to 'carbon', the coefficients indicate that it is more adjective than noun, though neither of the coefficients is strong. This observation is particularly curious since most of the semantic coefficients seem to indicate that the road-related usage dominates the corpus.

For semantic vectors, the road-related usage is represented fairly well, but the other abstract adjective is described less well. The adjective form appears to have two related basis terms: clumsy and ironic. Similar to the 'Coulson' example above, this observation implies that our system might produce worse representations for infrequently-used words or senses of words.

We also see an example of a high magnitude negative semantic coefficient, 'biological'. This coefficient is used in an interesting way here: a negative coefficient on 'biological' may mean something close to 'non-biological'.

5.3.5 Hub

$$
\begin{aligned}
\text{Hub} = {} & 0.49 * \text{bustling} + 0.47 * \text{C0} + 0.40 * \text{portal} \\
&+ 0.39 * \text{infrastructure} + 0.32 * \text{POS-NOUN} + 0.31 * \text{CAPITALIZATION} \\
&+ 0.31 * \text{central} - 0.13 * \text{POS-PROPN} - 0.10 * \text{PLURAL-NOUN} \\
&+ 0.07 * \text{outage} + 0.07 * \text{centre} + 0.06 * \text{POS-NUM} \\
&+ 0.06 * \text{connectivity} + 0.06 * \text{PARTICIPLE} \ - 0.06 * \text{POS-ADJ} \\
&+ 0.04 * \text{PAST-TENSE} \ - 0.04 * \text{POS-VERB} + 0.03 * \text{POS-ADV}
\end{aligned}
$$

This example demonstrates an intuitively-represented word. Syntactic coefficients correctly indicate that this word is a capitalized singular noun. Different semantic coefficients contribute slightly different senses of the word ('bustling' vs. 'connectivity').

5.3.6 Environmental

$$\begin{aligned}\text{environmental} = {} & 0.43 * \text{sustainability} + 0.43 * \text{economic} + 0.38 * \text{POS-ADJ} \\ & - 0.30 * \text{CAPITALIZATION} - 0.27 * \text{POS-VERB} + 0.27 * \text{regulatory} \\ & - 0.20 * \text{PAST-TENSE} + 0.18 * \text{biological} + 0.17 * \text{campaigner} \\ & + 0.17 * \text{POS-NUM} + 0.14 * \text{thermal} - 0.14 * \text{POS-ADV} \\ & - 0.12 * \text{POS-NOUN} + 0.10 * \text{health} - 0.10 * \text{C0} \\ & + 0.09 * \text{PARTICIPLE} + 0.09 * \text{POS-PROPN} - 0.08 * \text{PLURAL-NOUN} \\ & + 0.07 * \text{outdoor} + 0.06 * \text{chemical} + 0.01 * \text{cultural}\end{aligned}$$

The larger syntactic coefficients are appropriate and intuitive. Generally, the semantic coefficients are also intuitive, though there are some properties of the representation that are worth discussing. Again, note that many basis dimensions, such as 'economic', do not correspond to the dictionary definition of 'environmental' but instead correspond to contexts in which the word 'environmental' would be used (such as economic news). This explanation describes an especially sizable chunk of the basis dimensions in this case. In particular, it is worth pointing out the 'thermal' dimension. However, the word 'thermal' is presumably included in this representation because both 'thermal' and 'environmental' are related to global warming, even if they do not co-occur frequently. This effect implies that the basis vectors used to represent a word may not just be words that frequently co-occur with that word.

5.3.7 Churchill

$$\begin{aligned}\text{Churchill} = {} & 0.84 * \text{wartime} + 0.60 * \text{C0} + 0.41 * \text{CAPITALIZATION} \\ & + 0.40 * \text{quotation} + 0.38 * \text{POS-PROPN} + 0.36 * \text{statesman} \\ & - 0.21 * \text{PARTICIPLE} \ - 0.14 * \text{astronomer} - 0.14 * \text{POS-NOUN} \\ & - 0.11 * \text{POS-ADJ} - 0.10 * \text{PAST-TENSE} \ + 0.08 * \text{POS-VERB} \\ & + 0.08 * \text{POS-NUM} + 0.06 * \text{POS-ADV} + 0.04 * \text{advising} \\ & - 0.03 * \text{architectures} + 0.02 * \text{PLURAL-NOUN} \ + 0.02 * \text{pint} \\ & + 0.01 * \text{fascism} \end{aligned} \tag{5.6}$$

Unlike the representation for 'Coulson' above, the representation for 'Churchill' appears relatively noise-free. The syntactic coefficients are broadly correct, and the semantic coefficients are interpretable. The representation of Churchill as a 'wartime statesman' conveys a remarkable level of detail about a specific person, considering the coarse nature of our system. The high 'quotation' component is perhaps the best example of this system's reliance on context. If a human was trying to describe 'Churchill', 'quotation' would rarely be one of the words used. Of course, Churchill's name appears quite frequently in quotation-related concepts, and this fact is heavily noticed by the dense vectors, and, therefore, our system.

The most curious basis dimension is the highly negative coefficient on 'astronomer'. Negative coefficients are generally rare (see Chapter 5.2.3). While Churchill is not an astronomer, it is not immediately clear why there is a large negative coefficient here. While 'Churchill' has a low negative cosine similarity with 'astronomer' (-0.14), 2.7% of words are further away from Churchill than 'astronomer' is, and this distance is likely not enough by itself to make astronomer useful as a basis vector for Churchill. In addition, consider the relationship between 'statesman' and 'astronomer'. These two vectors are strongly related (cosine similarity = 0.33), and their projection along each other is uncorrelated with 'Churchill' (cosine similarity = 0.037). In intuitive terms, while 'statesman' represents some connotations

that are present in 'Churchill', it also represents some connotations that are not present in 'Churchill'. Including 'statesman' in the representation of 'Churchill' brings those negative connotations, so our system cancels out those connotations by including a negative coefficient on astronomer.

5.3.8 Resident

$$\begin{aligned}\text{resident} = {} & 0.54 * \text{citizens} + 0.49 * \text{native} + 0.37 * \text{visiting} \\ & - 0.19 * \text{PLURAL-NOUN} + 0.12 * \text{PAST-TENSE} + 0.11 * \text{caretaker} \\ & - 0.10 * \text{CAPITALIZATION} - 0.09 * \text{C0} + 0.08 * \text{PARTICIPLE} \\ & + 0.08 * \text{ward} + 0.08 * \text{POS-NOUN} - 0.04 * \text{POS-ADV} \\ & + 0.02 * \text{proprietor} + 0.02 * \text{POS-VERB} - 0.01 * \text{POS-NUM} \\ & + 0.00 * \text{POS-PROPN} + 0.00 * \text{POS-ADJ} \end{aligned} \tag{5.7}$$

'resident' is another example of an intuitively represented word. The syntactic coefficients, though low and ambiguous, are broadly correct. The only notable coefficient is the high weight on 'visiting', a word that is in many ways the opposite of 'resident'. This coefficient points to the inability, likely traceable to the original dense vectors, to distinguish between many types of semantic opposites.

5.3.9 Backers

$$\begin{aligned}\text{backers} = {} & 0.64 * \text{sponsors} + 0.40 * \text{POS-NOUN} - 0.40 * \text{CAPITALIZATION} \\ & + 0.33 * \text{advocates} + 0.28 * \text{PLURAL-NOUN} + 0.19 * \text{POS-PROPN} \\ & + 0.18 * \text{businessman} + 0.18 * \text{businessmen} + 0.16 * \text{fans} \\ & - 0.15 * \text{POS-ADJ} + 0.12 * \text{PARTICIPLE} + 0.12 * \text{PAST-TENSE}\end{aligned}$$

$$
\begin{aligned}
&- 0.12 * \text{POS-ADV} + 0.09 * \text{whose} + 0.08 * \text{opposition} \\
&+ 0.07 * \text{POS-VERB} + 0.06 * \text{candidacy} + 0.05 * \text{touted} \\
&+ 0.05 * \text{startups} - 0.02 * \text{POS-NUM} + 0.02 * \text{rebels} \\
&+ 0.01 * \text{reformist} + 0.01 * \text{investment} - 0.00 * \text{C0}
\end{aligned} \tag{5.8}
$$

Here, the most significant coefficients are very interpretable. The biggest syntactic coefficients are both correct and unambiguous. However, some of the lower-value semantic coefficients have unintuitive interpretations. For instance, look at the high coefficient on 'whose', a dimension that is hard to interpret in any context. This coefficient is especially unintuitive given that, among other words with a high coefficient on 'whose', most are very different from 'backers'. The top five words along the 'whose' dimension are the following: 'His', 'Those', 'Her', 'Its', and 'Their'. In fact, 'backers' is the word with the 85th highest coefficient along this dimension, and only one of the words higher-up, 'bearing', is somewhat similar to 'backers'. The problem does not seem to be that 'whose' codes for some unintuitive dimension. In fact, the set of words coded for by 'whose' are quite intuitive. 'backers' could potentially be associated with 'whose' because both words are associated with ownership. Even so, 'backers' is an unusual inclusion in this dimension.

5.3.10 Rudimentary

$$
\begin{aligned}
\text{rudimentary} = {} & 0.84 * \text{basics} - 0.65 * \text{C0} + 0.49 * \text{POS-ADJ} \\
&- 0.41 * \text{POS-VERB} + 0.41 * \text{apparatus} - 0.36 * \text{POS-NOUN} \\
&+ 0.35 * \text{improvised} + 0.15 * \text{POS-ADV} + 0.10 * \text{CAPITALIZATION} \\
&+ 0.07 * \text{PARTICIPLE} + 0.07 * \text{PLURAL-NOUN} + 0.06 * \text{POS-NUM} \\
&+ 0.06 * \text{develop} + 0.05 * \text{PAST-TENSE} + 0.04 * \text{POS-PROPN}
\end{aligned} \tag{5.9}
$$

Aside from the unusually low coefficient on C0, the syntactic coefficients with large magnitude

are correct and very intuitive. The semantic coefficients are less intuitive. While the highest coefficient, 'basics', makes sense, the second-highest, 'apparatus', is more mysterious. In this case, the 'apparatus' basis itself is surprisingly complex. Aside from the words we expect, such as 'machinery', 'equipment', or 'gear', the top twenty words along the 'apparatus' basis dimension also include words such as 'judiciary', 'Editorial', 'Ch', 'elaborate', and '1896'. Some of these words appear to be bureaucracy-related terms, capturing another sense of the work 'apparatus'.

Overall, examining these randomly selected individual words gives us a detailed low-level intuition about how our system works in practice. We can take away a few concrete lessons from this analysis. First, the vectors produced by our system are much more representative of the context in which words are used than they are of the dictionary meaning of those words. Second, our system has trouble disentangling closely related concepts that are, in some sense, direct opposites of each other. Finally, although our system is broadly intuitive, there are still many representations or dimensions that do not seem meaningful.

5.4 Errors in the Representation of Syntactic Concepts

In this section, we examine the use of syntactic basis dimensions to represent various syntactic concepts. In particular, we examine the difference between representations of words that only differ syntactically. As an example, consider the difference between the representation of words 'swim' and 'swimming' in the final version of our system:

$$\begin{aligned}\text{swim} = &-0.55 * \text{PARTICIPLE} \ + 0.37 * \text{walk} + 0.36 * \text{sink} \\ &+ 0.34 * \text{dolphins} + 0.29 * \text{surf} - 0.26 * \text{PLURAL-NOUN} \\ &- 0.26 * \text{PAST-TENSE} \ + 0.24 * \text{POS-VERB} - 0.21 * \text{C0} \\ &+ 0.17 * \text{fish} + 0.17 * \text{underwater} - 0.14 * \text{CAPITALIZATION}\end{aligned}$$

$$
\begin{aligned}
&- 0.12 * \text{POS-NUM} + 0.11 * \text{waters} + 0.10 * \text{POS-ADV} \\
&+ 0.10 * \text{diver} + 0.06 * \text{nude} + 0.06 * \text{POS-PROPN} \\
&+ 0.05 * \text{POS-NOUN} + 0.04 * \text{gym} + 0.02 * \text{POS-ADJ} \\
&- 0.01 * \text{timber} + 0.01 * \text{sailing} + 0.01 * \text{water} \\
&+ 0.00 * \text{synchronized}
\end{aligned} \tag{5.10}
$$

$$
\begin{aligned}
\text{swimming} = {} & 0.28 * \text{POS-VERB} + 0.28 * \text{water} + 0.26 * \text{walking} \\
&- 0.26 * \text{CAPITALIZATION} + 0.26 * \text{surfing} + 0.23 * \text{volleyball} \\
&- 0.23 * \text{PAST-TENSE} + 0.22 * \text{POS-ADJ} + 0.22 * \text{underwater} \\
&+ 0.19 * \text{dolphins} + 0.17 * \text{PARTICIPLE} - 0.16 * \text{PLURAL-NOUN} \\
&- 0.16 * \text{POS-NOUN} + 0.15 * \text{diver} + 0.11 * \text{fish} \\
&+ 0.10 * \text{synchronized} + 0.10 * \text{dancing} + 0.08 * \text{sailing} \\
&+ 0.07 * \text{POS-NUM} + 0.07 * \text{POS-ADV} + 0.07 * \text{POS-PROPN} \\
&- 0.07 * \text{C0} + 0.04 * \text{waters} + 0.04 * \text{skiing} \\
&+ 0.00 * \text{badminton} + 0.00 * \text{basketball}
\end{aligned} \tag{5.11}
$$

These representations correctly differ by a large coefficient in the 'PARTICIPLE' dimension. Ideally, this should be the only difference between the two representations. In practice, this is not the case.

There are two types of problematic differences between these two representations. The first are differences apparently caused by noise. For example, 'dolphins' has a high coefficient in 'swim' and a low coefficient in 'swimming', while 'dolphin' has a high coefficient in 'swimming' and a low coefficient in 'swim'. There is little apparent reason why these coefficients are so different, although subject-verb agreement means that 'dolphins swim' has fewer grammatical senses than 'dolphin swim'. This noise, however, reveals a potential flaw in our system. Slight

noise in the original dense vector may be greatly exaggerated in the transformed sparse vector because this slight noise will cause the vector to be coded using a different set of basis dimensions. We call this error *relationship noise*.

There is also a second, more pernicious, type of error. In the representation for 'swim' above, 'walk' is used as one of the basis dimensions while 'walking' is used for 'swimming'. That is, the syntactic properties of the semantic basis chosen may relate to the syntactic properties of the encoded word. This type of error is a particular problem because our model includes a substantial attempt to mitigate it: We removed the projection of all semantic basis vectors along all syntactic basis vectors. In theory, this should have removed all syntactic concepts from the semantic vectors (or, at least, all the syntactic concepts we considered). Clearly, this is not the case. We call this error *systematic relationship error*.

Note that the combined effect of both types of errors is best measured by the analogy tests (see Chapter 4.5). Specifically, an upper bound on these errors can be measured by the difference between the sparse and reconstructed vectors in the syntactic analogy subtasks. This error is significant: going from reconstructed dense vectors to sparse vectors reduces performance from 0.80 to 0.46 for $\alpha = 0.1$ (Table 4.1). Even so, we expect that the analogy task is more sensitive to these errors than tasks such as classification.

Relationship noise is hard to quantify for a variety of reasons. First, we have no way to automatically match words that differ only by a syntactic difference. While various techniques could produce words with identical stems, it is hard to automatically identify cases where a change in a syntactic property is confounded by a change in meaning. For instance, in the representations displayed above, 'swimming' is associated with 'synchronized', while 'swim' is not. Recall that, due to similar issues, we manually filtered such confounders from the list of word pairs for our syntactic basis vectors (see Chapter 3.3).

Instead, we will directly measure only systematic relationship error, and estimate (an upper bound on) the combined error by looking at the degradation in performance in the analogy tasks. To estimate systematic relationship error, we look at words that have a certain

Feature	Baseline	Positive
Noun	0.22	0.32
Verb	0.07	0.17
Adjective	0.08	0.18
Plural (Noun)	0.07	0.15
Participle	0.02	0.08
Past Tense	0.00	0.06

Table 5.3: For a variety of syntactic features, the first column shows the fraction of basis vectors that have this feature in the representations for words that do not have this feature. For example, the fraction of the nonzero terms that are nouns in the representations of nouns. The second column shows the fraction of basis vectors that have this feature for words that do have this feature. Words that are themselves basis vectors are excluded from this analysis. Syntactic features were estimated using the spaCy library [15]. Some syntactic features (such as proper nouns) were excluded because they are filtered from the potential basis vectors.

syntactic feature and see how many of the basis vectors used to represent those words have that same feature. As a baseline, we compare to the representations of words that do not have these syntactic features. Ideally, these proportions would be the same. The results are shown in Table 5.3.

We can see that none of the syntactic features are completely controlled. Basis vectors with a particular part-of-speech were used to code for words with the same part-of-speech around 10 percentage points more than the baseline. For categories other than nouns, this is a bit more than a 2x increase. Non-part-of-speech features, for which the syntactic basis vectors were created using a different method, are rarer but have a much greater proportional increase. Though the comparison depends on whether we are looking at relative or absolute

differences, it appears that non-part-of-speech features display a greater imbalance. This suggests that our construction of the syntactic basis vectors for these features may not be as accurate.

Overall, though we attempted to prevent semantic basis vectors from representing syntactic components, the controls we used do not appear to be completely effective.

5.5 Representation of Numbers

Natural language text contains many numbers, and these numbers are considered tokens and are assigned their own embeddings in the original FastText dense vectors. Therefore, our system also creates sparse representations for the most frequently occurring numbers. These numbers form a distinct and nuanced subset of the vector space that may not obey the implicit or explicit assumptions that we used when designing our system. For instance, describing numbers with syntactic basis vectors may lead to unintuitive results. Therefore, we want to analyze these representations to understand how our system behaves in edge cases.

The top 30,000 tokens contain 925 tokens that are tagged as numbers by spaCy [15]. Most of these tokens are decimal representations of integers. A substantial portion of those are four-digit integers that primarily represent years. A smaller number of these tokens are integers written out as words, such as 'one', 'twenty-four', or 'billion'. A few of these tokens are other forms, such as '3.5', 'two dimensional', '2-1', or '9-11'.

First, we analyze the basis dimensions that are used to encode these numbers, starting with syntactic basis dimensions. One of the syntactic basis dimensions encodes whether or not a word is a number, and most numbers have a high coefficient along this dimension (Figure 5.1). This syntactic dimension has the highest separation between words that do and do not match this feature (see page 55). Ideally, the coefficients of numbers on all other syntactic

basis dimensions should be small and centered around zero.

The actual distributions of syntactic basis coefficients within numbers is presented in Figure 5.3. First, we can see that a few dimensions identify numbers in general. Numbers tend to have strong negative coefficients on the noun and proper-noun dimensions, and slightly less strong negative coefficients on the adjective dimension. Here, recall the order in which syntactic basis orthogonalization happens: both nouns and proper nouns occur before numbers, so the vector components associated with both numbers and (negatively) associated with these parts of speech become a part of the corresponding part-of-speech vector and not the number vector. Note that vectors that appear after the number vector are orthogonalized with respect to the number vector and are almost perfectly centered at zero. These numbers are indeed not nouns, proper nouns or adjectives and correctly receive negative coefficients.

The sharp split between years and non-year numbers is more nuanced. This split occurs in some, though not all, of the syntactic basis dimensions. They appear in a variety of syntactic basis dimensions that have no intuitive relation to the distinction between dates and other numbers. This result seriously questions the ability of our system to intuitively represent subtle meanings, such as numbers.

Next, we consider semantic loadings. In total, 3,449 semantic basis coefficients, spread across 892 distinct semantic dimensions, are used to describe these 925 numbers. This is comparatively a low number: Only 3.73 coefficients per word, compared to 8.85 coefficients per word overall. Recall that semantic basis vectors may pose a particular problem for numbers because we excluded any number from being used as a potential basis vector.

We manually examine the semantic basis vectors used to code for more than 20 numbers. Some do not have any intuitive relationship to numbers, such as ‘noted’ or ‘consequence’. But many of these semantic basis vectors are either related to numeric concepts, such as ‘multi’, or are number-adjacent tokens that were not identified as numbers by spaCy, such as ‘1930s’ or ‘tenth’. Manually examining the usage of these vectors, we find that semantic basis vectors strongly distinguish the different categories of numbers that we discussed above, though the

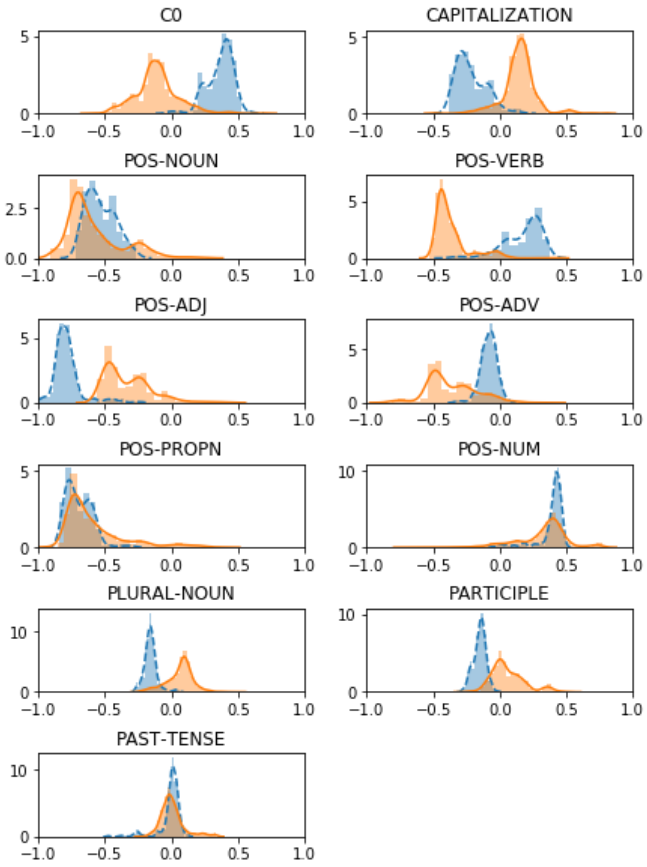

Figure 5.3: The distribution of syntactic basis coefficients for all 925 words that correspond to numbers. The dashed distributions are the distribution of numbers that are not years, while the solid distributions are the distribution of numbers that are years. Years are defined as decimal representations of integers between 1,000 and 2,500. Height of these plots is a kernel density estimate.

semantic basis vectors sometimes bear little relation to the categories they identify. For instance, years are represented by vectors such as 'noted', 'consequence', or 'asserts'. Round integers in the hundreds or thousands, such as '5000', often have a strong negative coefficient with these dimensions. Small integers are represented by basis dimensions such as 'should' or 'can'. 'quarterfinals' is used to represent numbers such as '2-1' or '6-0', which presumably refer to sports results. Non-integer decimals such as '3.1' or '6.5' are represented by vectors such as 'averaged' or 'upgrade'. Numbers spelled out using alphabetic characters, such as 'twenty' or 'eleven', are represented by vectors such as 'few' or 'tenth'.

Overall, our model is able to distinguish different subcategories of numbers, preserving the relationships that existed in the dense vector space. This structure holds even in a subcategory that is not well represented by the allowed set of semantic or, to a lesser degree, syntactic, basis vectors. However, this structure comes at the cost of intuitive representations. Most (though not all) of the subclasses of numbers were not represented in a meaningful way.

5.6 Systematic Reconstruction Error

Recall that we defined reconstruction error as the difference between the original dense vector and the dense vector created by multiplying the sparse vector by the basis matrix. Conceptually, the sparse vector exactly corresponds to a vector in the dense space that is an approximation to the original vector. Reconstruction error is the cosine dissimilarity between the approximation and the original vector.

In Chapter 4.4, we examined the degree that this reconstruction error affects the performance of our system. We previously assumed than the reconstruction error is simply a loss of precision caused by the introduction of noise, instead of a systemic error that introduces various statistical biases into our system. Because the reconstruction error is a significant drawback of our system, a detailed understanding of it is crucial.

We conduct a variety of tests to determine whether the direction of the reconstruction error is uniformly distributed. For this analysis, we normalize the reconstruction error, so that we only consider the vector direction. Taking the mean of normalized vectors tests if the vectors are pointing in the same direction. We consider a simple test on the mean of the reconstruction errors. The mean normalized reconstruction error is a vector with magnitude 0.056.

To test if this value is small or large, we run simulations for this test statistic. We assume as the null hypothesis that reconstruction error is Gaussian distributed, with mean zero and symmetric variance, and then normalized. This process produces a uniform distribution on the surface of the hypersphere of unit vectors. In each simulation, we generate 30,000 vectors according to this process and take the mean. We repeat the simulation 1,000 times. This process gives us the expected range of test statistics under the null hypothesis. Though the mean reconstruction error of 0.057 might appear small, a difference this large did not appear in any of the 1,000 simulations we ran, implying that the difference is statistically significant with $p < .001$. This difference was 8.6 times as large as the largest mean observed by pure chance in our simulations. We conclude that reconstruction error has a slight but significant directionality.

Though the mean reconstruction error is meaningful, we also care about the extent to which the variance of the reconstruction error is evenly distributed throughout all dimensions of the dense vector space. If the variance is unevenly distributed, then that would imply that our system is less able to represent certain dimensions in the dense vector space, which may correspond to an inability to represent certain concepts as accurately.

We test this property in two ways. First, we calculate the standard deviation for each dimension. Figure 5.4 visualizes these standard deviations. We can see that the distribution of per-dimension standard deviations is close to the reference random distribution. Though the distribution under the null hypothesis consistently slightly underestimates the observed standard deviation, it is important to remember here that this distribution makes many as-

sumptions about how closely the distribution of the reconstruction errors and the distribution of word embeddings are related. Overall, this distribution does not show significantly more variance than the assumed distribution under the null hypothesis, so we cannot conclude that the reconstruction error favors certain dimensions.

The analysis above assumes that any differences in variance would occur only along the axis-aligned dimensions of the dense vector space. However, there is no reason to assume that is the case, since the dimensions of the dense vector space could be arbitrarily rotated. To further examine this hypothesis, we analyze the principal components of the reconstruction error. The amount of variance explained by each principal component is displayed in Figure 5.5.

With some minor deviations, the distribution of variances of principal components is approximately the same. Again, the minor deviations are likely the results of the assumptions we made in picking an appropriate distribution under the null hypothesis.

While we have found evidence that the distribution of reconstruction errors has nonzero mean, this difference is slight and we have not found any evidence that the reconstruction errors are distributed unevenly across dimensions in the dense vector space.

5.7 Words with Large Reconstruction Error

We focus next on characterizing those words that have unusually high reconstruction error. We will start by characterizing classes of words with particularly high reconstruction error, before analyzing individual words.

First, we analyze reconstruction error by part-of-speech. Distributions for each part-of-speech tag are displayed in Figure 5.6.

Generally, we do see significant differences in reconstruction error across different part-of-

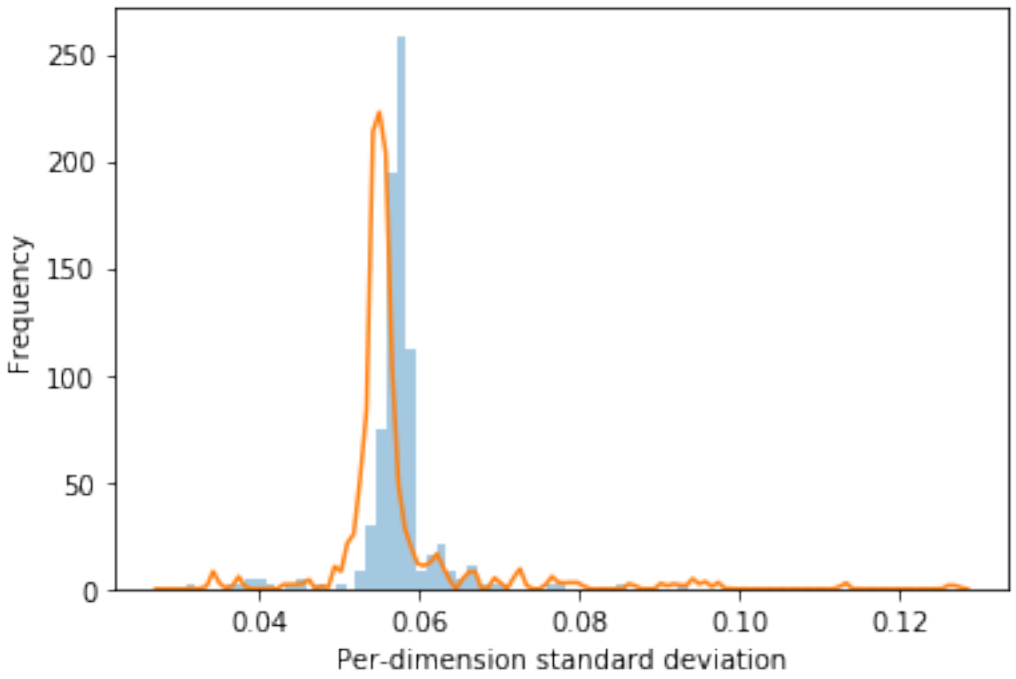

Figure 5.4: The dark grey (blue) histogram shows the distribution of standard deviations of dimensions of the reconstruction error. The light grey (orange) line shows a kernel density estimate of the distribution expected if the reconstruction error was distributed according to a reference distribution. We generate the reference distribution by the following process: we start with a Gaussian distribution, scale according to the per-dimension standard deviations of all word embeddings in the original vector space, and normalize to unit length.

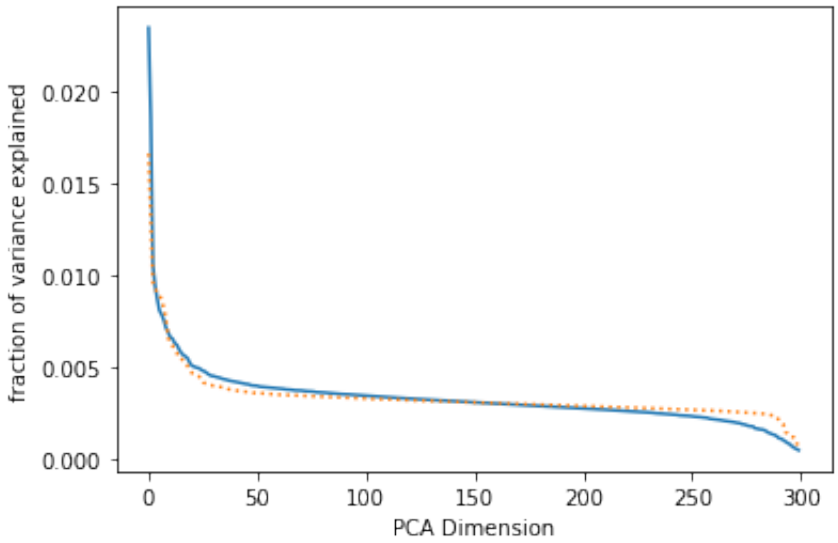

Figure 5.5: The line shows the fraction of variance explained by each successive principal component of the reconstruction error. The dotted orange line shows the fraction of variance explained by each successive principal component of data generated from a reference distribution. We generate the reference distribution by the following process. We start with a standard Gaussian distribution, scale according to the per-dimension standard deviations of all word embeddings in the original vector space, and normalize to unit length.

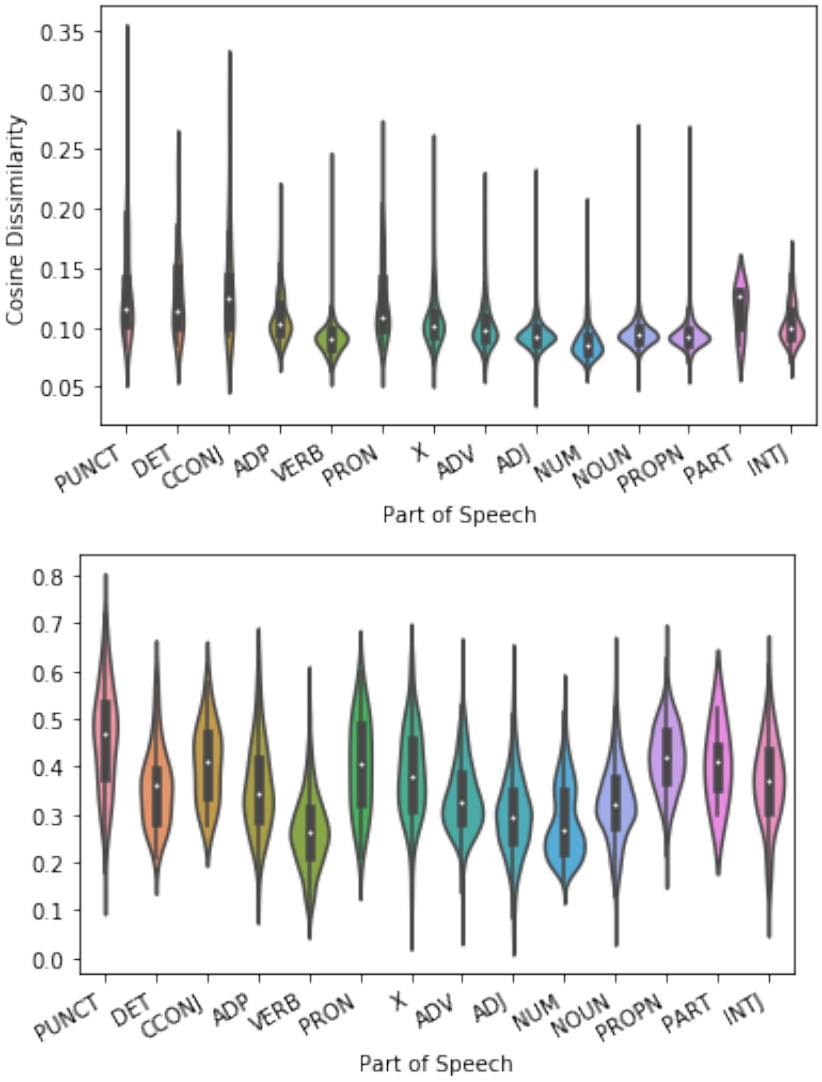

Figure 5.6: Violin plots of reconstruction error for words, grouped by part-of-speech. Results are shown for $\alpha = 0.10$ (top) and $\alpha = 0.35$ (bottom). Part-of-speech tags are estimated using the spaCy library. The 'X' category contains words with unknown part of speech.

speech tags. Overall, we see unusually high reconstruction error in syntax-related part-of-speech tags such as punctuation ('PUNCT'), pronouns ('PRON'), or interjections ('INTJ'). Recall that these parts of speech were excluded from the potential semantic basis vectors. We also see high reconstruction error for proper nouns ('PROPN'), likely because they were also excluded from potential semantic basis vectors.

There is a complicated relationship between the reconstruction errors for the two values of α that we examine. Looking at reconstruction error by part-of-speech, the two levels of α display similar, though not the same, patterns. For instance, proper nouns have relatively low reconstruction error for $\alpha = 0.1$ but relatively high reconstruction error for $\alpha = 0.35$. The relationship between reconstruction errors of individual words is even more tenuous. Figure 5.7 displays the relationship between reconstruction error distributions for the different levels of α.

We can see that, although there is a statistically significant positive relationship between reconstruction errors across the two levels of sparsity, this relationship explains almost none of the variance ($r = 0.103$, $p = 4.8e - 71$).

The relationship between frequency and reconstruction error depends on the value of α. When $\alpha = 0.35$, more frequent words have lower reconstruction error ($r = 0.096$)[1]. But when $\alpha = 0.10$, more frequent words have higher reconstruction error ($r = -0.10$).

However, the trend is nonlinear, and estimates from linear regression may give misleading values. These conflicting trends appears to be related to the part-of-speech trends examined above. For the most frequent 2,000 words, which contain many syntax-related words such as 'the', frequent words have higher reconstruction error for both levels of α. This effect is stronger for $\alpha = 0.10$ ($r = -0.45$) but is also present for $\alpha = 0.35$ ($r = -0.15$). For words outside the most frequent 2,000 words, the relationships again point in the same direction but with a much higher strength for $\alpha = 0.35$ ($r = 0.101$ for $\alpha = 0.35$, $r = 0.02$ for $\alpha = 0.10$).

[1]The FastText vectors provide only relative, not absolute, frequencies. The correlation statistics we compute are based on the ranks of frequencies of words.

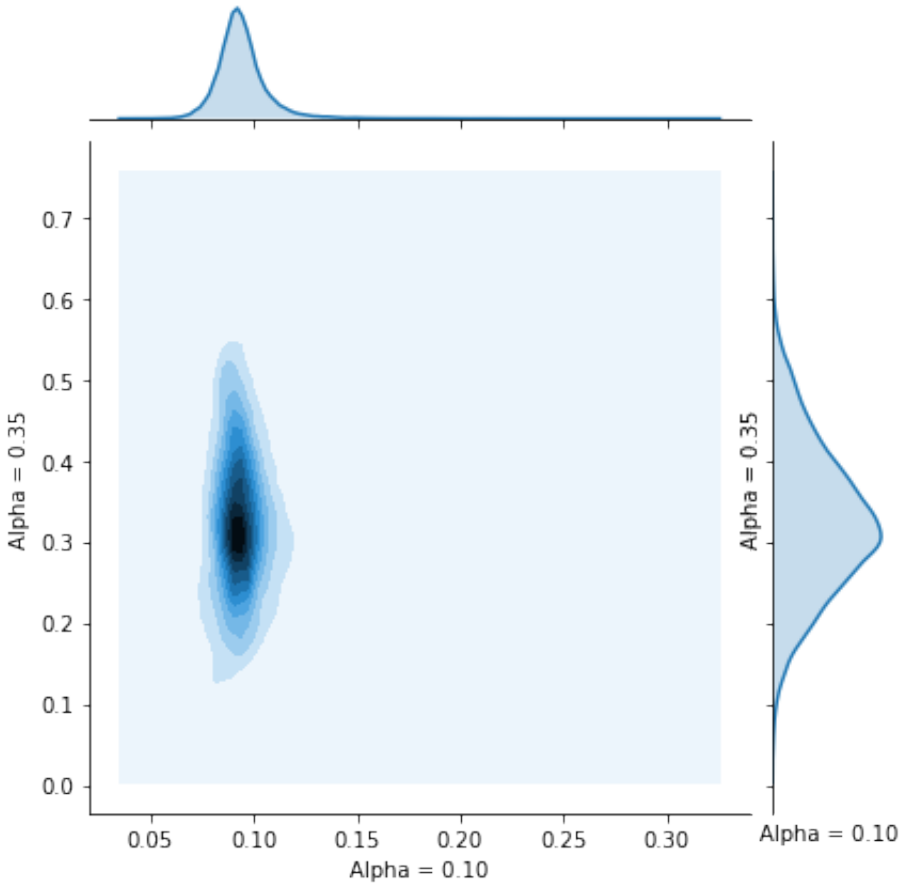

Figure 5.7: A kernel density estimate of the joint and marginal distributions for per-word reconstruction error for $\alpha = 0.10$ and $\alpha = 0.35$

Overall, we do not see any clear and consistent relationship between word frequency and reconstruction error.

Next, we individually examine words with particularly high reconstruction errors in an attempt to discover fine-grained patterns. For each level of α, we randomly sampled 15 words from the 2% of words with the highest reconstruction error.

For $\alpha = 0.1$, the randomly sampled words are: 'Vettel', 'him', 'think', 'trio', 'reference', 'a', 'Pistorius', 'NGC', 'Wilshere', 'c', 'after', 'и', 'rather', 'wonder', and a Bengali glyph. For $\alpha = 0.35$, the randomly sampled words are: 'Levin', 'Preston', 'Lawrence', 'Post', 'Carlo', 'Kingsley', 'Flora', 'Harold', 'Willard', 'Don', 'O.', 'W.', 'Rey', 'Kamal', 'Carey'.

Generally, these words match our intuitions from the part-of-speech analysis above. We see many proper nouns and punctuation marks. However, looking at the part-of-speech tags assigned to the words above suggests that inaccuracies in the part-of-speech tagging may be underestimating the fraction of high reconstruction error words that fall into the categories above. For instance, 'Pistorius' was tagged as an adjective, while 'Levin', 'Willard', 'O.', and 'W.' were tagged as nouns, among other errors. It is possible that a significant number of the very-high reconstruction error words seen in Figure 5.6 that are tagged as nouns, verbs, or adjectives (according to the spaCy part-of-speech taxonomy used to create the syntactic basis vectors) may be misclassified.

The high reconstruction error words we see above that are not proper nouns or various non-word tokens are all very context neutral words, or words that we would expect to appear frequently across a wide range of contexts. Even many words that were correctly tagged as nouns or verbs, such as 'think' and 'wonder', are extremely context neutral.

To explore the two hypotheses above, we repeat the same process to generate words randomly sampled from the top 1% of verbs with the highest reconstruction error.

For $\alpha = 0.1$, the randomly sampled verbs are: 'incurred', 'Granted', 'needed', 'Eating', 'appalled', 'Agreed', 'Am', 'imaging', 'Getting', 'Look', 'Give', 'said', 'Awarded', 'meant',

'constitutes'. For $\alpha = 0.35$, the randomly sampled verbs are: 'Townsend', 'Show', 'tipping', 'ADD', 'Clash', 'Says', 'Asked', 'Added', 'Bring', 'Agreed', 'Illustrated', 'Loving', 'Add', 'Granted' and 'shadow'.

We can see that the high reconstruction error verbs for $\alpha = 0.10$ are all very context neutral verbs, as we expected from our analysis above. However, the same does not hold for $\alpha = 0.35$, where, in addition to context neutral verbs, we see words like 'Clash' or 'Loving'.

Recall that we believe the C0 syntactic basis vector represents words that appear frequently in every context. Assuming that this interpretation is correct, we can use this to measure the extent to which context neutral words have higher reconstruction error. We see a strong correlation, especially for $\alpha = 0.35$, For $\alpha = 0.35$, the correlation between reconstruction error and the C0 coefficient has correlation coefficient $r = 0.46$, much higher than the next more correlated syntactic basis dimension (Capitalization, with $r = 0.34$). The correlation is much less strong for $\alpha = 0.10$: $r = 0.19$, while the Number syntactic basis vector has correlation $r = -0.31$. Note that the ordering of effect size between the two levels of α is reversed from the analysis above.

Overall, we can conclude that words with high reconstruction error are often proper nouns, non-word tokens, or context neutral words. The latter is both the most difficult category to define and the category for which the evidence is most mixed. Here, we must pay particular attention to the low relationship between reconstruction errors across different levels of α. The high variance suggests that either the criteria for high reconstruction error differs substantially between different levels of α (as supported by some or our data) or that most of the variation in reconstruction error can be attributed to noise and not to innate properties of a given word.

Chapter 6

Conclusions and Future Work

In this work, we presented a method to create word embeddings that are interpretable by construction. Each dimension of these embeddings corresponds precisely to a natural language word. These embeddings can be presented in a human readable form, and we have shown that most of these representations are intuitive. We have also shown that these embeddings can be used to produce an extremely interpretable classification model that still delivers performance comparable to or better than a classification model based on the original embeddings.

Unlike most previous work on interpretable word embeddings, our method does not require humans to interpret and label each dimension. We have previously seen how this feature allows us to easily create interpretable classification models. It also allows us to gain a deeper understanding of the original dense vector space. Previous approaches may have obscured nuanced or hard to interpret behavior. In particular, a human manually interpreting a dimension may not appreciate subtle behavior of the system. Several sections of this work, which have manually examined individual word representations in our system, have revealed the nuanced behavior that our system demonstrates.

Our method still has some serious drawbacks. While we have examined a number of these

flaws, many are tied closely to the sparse coding method we have chosen to use. Sparse coding, by its nature, introduces a substantial amount of noise in the form of reconstruction error. In addition to the reconstruction error, sparse coding has the potential to assign very different sparse vectors to similar dense vectors. We hope that future work will produce sparse embeddings that are interpretable by construction without some of the shortcomings of our work.

Much of the promise of sparse coding methods remains to be proved. In particular, we believe it will be fruitful to study the representation of syntactic concepts. We have seen that our attempts to disentangle syntactic concepts from our semantic basis vectors were not entirely successful. We would also like to better understand how these methods are applicable in deep learning models.

There is still a large amount of analytical work left to be done on evaluation. The word intrusion task, while an effective quantitative method, does not offer a complete view of interpretability. Part of this problem is that we do not have any way to quantify interpretability where it is most useful: when building downstream classification models. More fundamentally, we do not have any underlying framework for understanding what it means for a word embedding to be interpretable.

We believe that interpretable word embeddings have great potential for helping us understand and interpret models in a wide range of NLP tasks.

Bibliography

[1] Mathieu Blondel and Fabian Pedregosa. Lightning: large-scale linear classification, regression and ranking in Python, 2016.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] Rakesh Chalasani, Jose C Principe, and Naveen Ramakrishnan. A fast proximal method for convolutional sparse coding. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5. IEEE, 2013.

[4] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296, 2009.

[5] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, 2011.

[6] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[7] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[8] Philipp Dufter and Hinrich Schütze. Analytical methods for interpretable ultradense word embeddings. *arXiv preprint arXiv:1904.08654*, 2019.

[9] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. Sparse overcomplete word vector representations. *arXiv preprint arXiv:1506.02004*, 2015.

[10] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[11] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. The lottery ticket hypothesis at scale. *CoRR*, abs/1903.01611, 2019.

[12] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[15] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[16] Patrik O. Hoyer. Non-negative sparse coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565. IEEE, 2002.

[17] Durk P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[18] Aykut Koç, Lutfi Kerem Senel, Ihsan Utlu, and Haldun M. Ozaktas. Imparting interpretability to word embeddings while preserving semantic structure. *arXiv preprint arXiv:1807.07279*, 2018.

[19] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[20] Thomas K. Landauer and Susan T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211, 1997.

[21] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808, 2007.

[22] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[23] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[24] Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. In *Guide to Big Data Applications*, pages 83–104. Springer, 2018.

[25] Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[26] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[27] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human*

Language Technologies, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[28] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.

[29] Binny Mathew, Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. The polar framework: Polar opposites enable interpretability of pre-trained word embeddings. *arXiv preprint arXiv:2001.09876*, 2020.

[30] Oren Melamud and Jacob Goldberger. Information-theory interpretation of the skip-gram negative-sampling objective function. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 167–171, 2017.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[32] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17, pages 2498–2507. JMLR.org, 2017.

[33] Ari S. Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv preprint arXiv:1906.02773*, 2019.

[34] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. CuPy: A NumPy-compatible library for nvidia gpu calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[35] Travis E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[36] Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. Word2sense: Sparse interpretable word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5692–5705, 2019.

[37] Sungjoon Park, JinYeong Bak, and Alice Oh. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411, 2017.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[39] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[40] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.

[41] Sascha Rothe and Hinrich Schütze. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–517, 2016.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[43] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and

Eduard Hovy. Spine: Sparse interpretable neural embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[44] Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems*, pages 887–898, 2018.

[45] Juexiao Zhang, Yubei Chen, Brian Cheung, and Bruno A. Olshausen. Word embedding visualization via dictionary learning. *arXiv preprint arXiv:1910.03833*, 2019.

[46] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *arXiv preprint arXiv:1905.01067*, 2019.