EVALUATING RELATIONSHIPS BETWEEN VECTOR SPACES OF WORD EMBEDDINGS

by

Safia Hassan

A Senior Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the Degree

Bachelor of Computer Science

School of Computing

The University of Utah

December 2017

Approved:

Jeff Phillips / 12.18.2017

Ross Whitaker / 12-18-2017
Director School of Computing

H. James de St. Germain / 18-Dec-17
Director of Undergraduate Studies
School of Computing

ABSTRACT

Several techniques within Natural Language Processing rely on the representation of text data to enable computers to better "understand" human language. Useful techniques to be able to effectively model text data can result in much greater accuracy in understanding information extracted from such text. Word emdeddings, a popular framework to represent text data, is commonly achieved through a neural-network based approach to represent words as d-dimensional vectors. These vectors can further be used to analyze text. This paper will focus on evaluating the vector spaces of two types of word embeddings through an absolute orientation technique of their respective coordinates.

1 Introduction

One important facet of Computer Science is teaching computers to correctly deal with human language as it is spoken. Due to the inherent ambiguous nature of natural language, it is a difficult, yet important, task to teach a computer to learn how to understand text in an unambiguous manner. Several text representations have been studied within Natural Language Processing to achieve better performance in understanding text, a form of data that is ubiquitous in our day-to-day lives.

Many techniques for text representation in Natural Language Processing present words as atomic units. This sort of representation encodes words to symbols that don't attempt to capture the relationships between words. An example of such technique is the N-gram model used for statistical language modeling [1]. Although training with N-gram models can go up to 1 trillion words, performance has reached its limits with many tasks such as understanding relevant in-domain data.

Alternatives to simpler models used within Natural Language Processing, like N-gram, rely heavily on machine learning techniques. These techniques give rise to word embeddings, or a distributed representation of words. Word embeddings at their core capture rich semantic information about words and their meanings [2]. Each vocabulary word is encoded as a *d*-dimensional word vector, where a given dimension represents a particular feature of a reference word. The learning algorithm, usually a neural net, is responsible for figuring out continuous valued features that best represent a vocabulary word.

2 Background

2.1 Understanding Word Embeddings

New techniques for learning high-quality word vectors have revealed that along with finding similarities between words in a vector space, it is possible to perform simple algebraic operations to answer syntactic analogies between words. For example, the vector "King – Man + Woman" results a vector that is closest to the word "Queen" [3]. One technique introduced by Mikolov et al. uses neural networks to form word embeddings, and these embeddings encode many linguistic regularities and patterns.

Figure 1 shows an example of the resulting vector space of two types of word embeddings. Both of these embeddings map words onto 20-dimensional vectors. After performing principal component analysis and projecting the points onto a 2-dimensional space, the results reveal the valuable substructure obtained from the embeddings. Words that are similar are closer together, and arithmetic operations within the vector space preserves relationships.

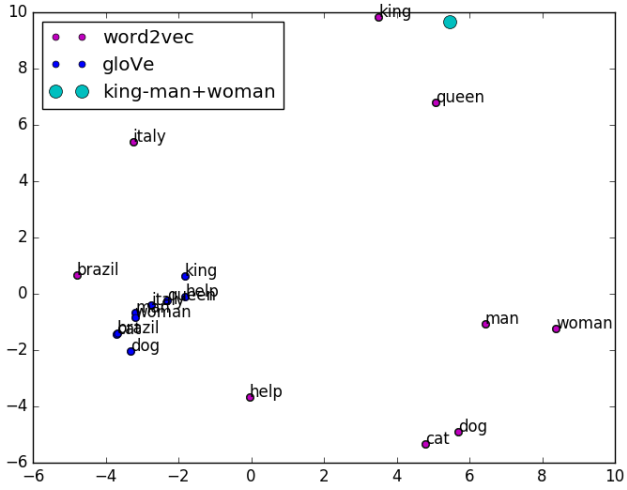

Figure 1: Word2Vec and GloVe embeddings of nine words.

2.2 Types of Word Embedding Models

Two main methods for computing word embeddings, count-based models and context-predicting models, rely on distributional semantic models (DSMs). DSMs utilize the important notion in computational linguistics that shows that contextual information provides a good approximation to the meaning of a word. Count-based models include DSMs built in the traditional manner of initializing vectors with co-occurrence counts. Context-predicting models rely on neural networks for supervised context prediction training [4]. Baroni et al. performed the first comparative evaluation of count-based and prediction-based vectors. Prior to their paper, context-predicting models were widely used due to their performance, but there was no literature support highlighting their comparison with count-based models. The comparison report of the performance of count-based models and their counterpart, prediction-based models, regarded several benchmarks for types of tasks including:

- Semantic Relatedness – Relatedness between two words.
- Synonyms – Selecting the correct synonym category for a target word.
- Concept Categorization – Categorize words into natural categories (cat, dog → mammals)
- Analogy –Test syntactic analogy introduced by Mikolov et al.

The results showed that prediction-based models consistently outperform count-based models.

2.3 Learning Methods

Due to the fact that context-predicting models outperform count-based models, this paper focuses on evaluating the vector spaces of prediction-based models. The two types of prediction-based word embeddings that we will consider are: Word2Vec, and GloVe [3,5].

2.3.1 Word2Vec

The prediction based learning methods proposed by Mikolov et al., Word2Vec, rely either on a Continuous Bag-of-Words Model (CBOW) or a Skip-gram Model. The primary difference between these models is the method for predicting a particular word. For the CBOW architecture, a word is predicted based off of the context in which it appears. So the input for this model is a set of words, and the output is the word that is predicted. For the Skip-gram Model, the reverse of CBOW occurs. Given an input word, the model predicts words surrounding that input.

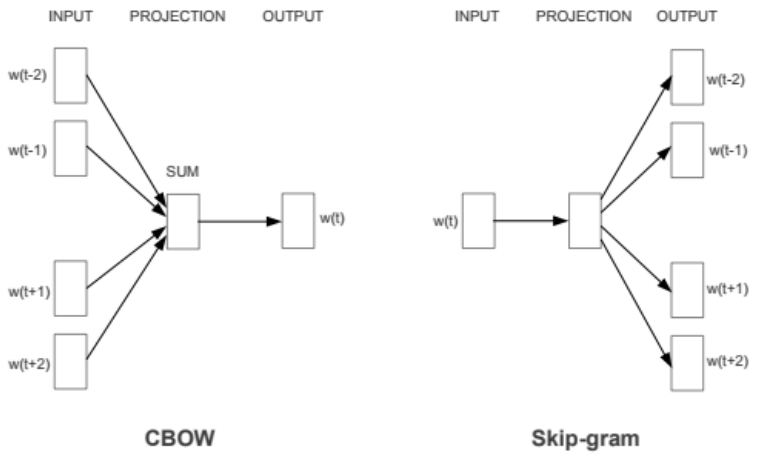

Figure 2: Example of CBOW and Skip-gram models [3].

2.3.2 GloVe

The Global Vectors for Word Representation (GloVe) model aims to use the information about the global statistics of a corpus while still capturing meaningful linear substructures provided by prediction-based methods [5]. The motivation for the GloVe model is to create resulting vectors that represent the meaning generated by the statistics of word occurrences.

2.4 Absolute Orientation

Finding a relationship between two coordinate systems is an important task prevalent in photogrammetry and robotics. Absolute orientation, a technique proposed by Horn et al, helps to accomplish this task [6]. This technique recovers the transformation between points on the respective coordinates by finding the best translational offset along with the best rotation. In particular this minimizes the sum of squared distances between aligned words.

The interest of absolute orientation in evaluating vector spaces of word embeddings is due to the fact that the mapping techniques present in word embeddings do not rely on a fixed rotation and translation. As embeddings prove to be powerful in providing vector spaces that exhibit a meaningful substructure, recovering rotations between them may unveil additional underlying relationships. Variants of absolute orientation presented in *ICP: A Users Guide* [7] include:

- Absolute Orientation with Rotation Matrices and Singular Value Decomposition [10]
- Absolute Orientation with Rotation Matrices and Eigenvalue Decomposition [6, 11]
- Absolute Orientation with Unit Quaternions [12,13]
- Absolute Orientation with Dual Number Quaternions [14]

These variants all study the technique within a 3-dimensional space. Additionally, scaling is not introduced in any of the variants.

3 Related Work

3.1 Uses of Word Embeddings

Word embeddings are used in many Natural Language Processing applications due to significant performance prediction-based learning methods provide. Improvements in sentiment classification have occurred through the use of such embeddings due to learned word vectors capturing relational structures present in the vocabulary [8]. Similar progress is present in other applications of NLP such as named entity recognition, word sense disambiguation, relational extraction, semantic role labeling, and machine translation [9].

Due to the popularity of prediction-based word embeddings, Bollegala et al. [9] explored the relationship between prediction-based embeddings and count-based embeddings through linear transformation techniques. The approach for their findings was through an iterative procedure that converges to a local minimum of a regularized version of the sum of squared errors. The approach presented in this paper relies on a noniterative technique that is guaranteed to find the local minimum. Finding this sort of relationship can highlight characteristics of an embedding that result in higher performance. This can thereafter be used to improve existing learning methods.

Similar to Bollegala et al., this paper will not focus on proposing new word embedding learning methods, but rather on evaluating the relationship between two predictive word-

embeddings. We begin by first extending the technique of absolute orientation to higher dimensions. Because this technique has commonly been used with low-dimensions (usually 3-d). In our work we explore this technique in higher dimensions, up to a 150 dimensional space.

4 Method

4.1 Absolute Orientation Algorithm

The goal of absolute orientation is minimizing the sum of square errors. Arun et al. [10] propose a solution using the singular value decomposition (SVD) that perfectly minimizes the error, thus finding the best existing rotation. We will rely on this variant of the technique due to its scalability to higher inherent stability [7].

Consider two point sets A and B. Absolute orientation of aligning B onto A AbsoluteOrientation(A,B) requires:

Centering each dimension in the point sets of A and B with respect to their centroids:

$$\bar{a} = \frac{1}{n} \sum_{a \in A} a$$

$$\bar{b} = \frac{1}{n} \sum_{b \in B} b$$

and then updating each set of vectors by subtracting off the mean:

$$a'_i = a_i - \bar{a}_i$$

$$b'_i = b_i - \bar{b}_i$$

Next perform a vector outer-product of points in the rotating point set with respect to the other point set:

$$\mathrm{N} = \sum_{i=1}^{n} b_i \, a_i^{\mathrm{T}}$$

Finally, call the SVD(N) which results in orthogonal matrices that the rotation can be derived from:

$$N = U \Sigma V^{\mathrm{T}}$$

$$R = V U^{\mathrm{T}}$$

Finally, to apply this rotation to all of the points in B:

$$B' = \sum_{i=1}^{n} R \, b_i^{\mathrm{T}}$$

The final result requires adding back the mean of the point sets that the rotation is aligning to:

$$B' = B' + \bar{a}$$

Figure 3 is an example of absolute orientation on a set of four points in 3-dimensions. We have an initial point set A, a random reflection and/or rotation by an orthogonal matrix of the same dimensions of the initial point set which yields A', and a resulting point set after performing absolute orientation.

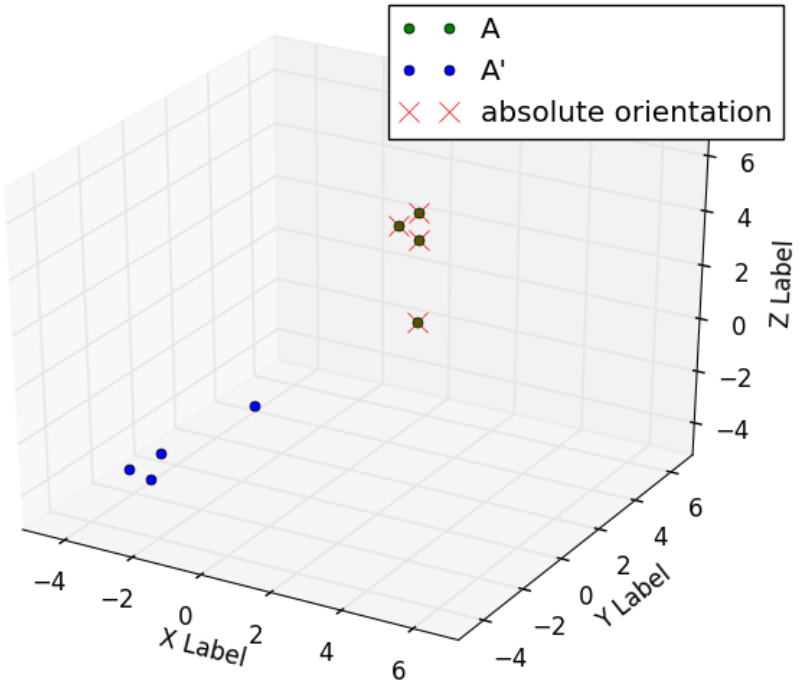

Figure 3: Absolute orientation in 3-dimensions

5 Results

5.1 Setup

The training data set, *First billion characters from Wikipedia,* used for both learning methods was obtained through (https://code.google.com/archive/p/word2vec/). To generate embeddings for Word2Vec we relied on the python library, Gensim. To generate embeddings for GloVe we relied on the implementation provided by Pennington et al. [5].

5.2 Absolute Orientation Applied to Word Embeddings

The aligning decision we made for this paper entails aligning the point sets in the vector space obtained through GloVe to the point sets in the vector space obtained through Word2Vec. Our experiments deal with handling dimensions of sizes 20, 50, and 150.

We will begin by first examining absolute orientation on the same set of words from Figure 1. As displayed in Figure 4, individual embeddings of Word2Vec and GloVe represent similar words close together in the vector space. This is an expected result of a predicative word embedding. In terms of "closeness" for the respective embeddings, clearly there isn't a great relationship throughout both point sets.

In order to quantifying “closeness”, we will calculate the Frobenius norm of the two vector spaces where Aw represents the vector space obtained through Word2Vec, and Bg represents the vector space obtained through gloVe. Similarly to evaluate “closeness” of the vector space as a result of absolute orientation we will calculate the Frobenius norm of Aw and the result of absolute orientation of Bg onto Aw.

$||Aw - Bg||_F = 27.8173751416$

$||Aw - AbsoluteOrientation(Aw, Bg)||_F = 15.4417315303$

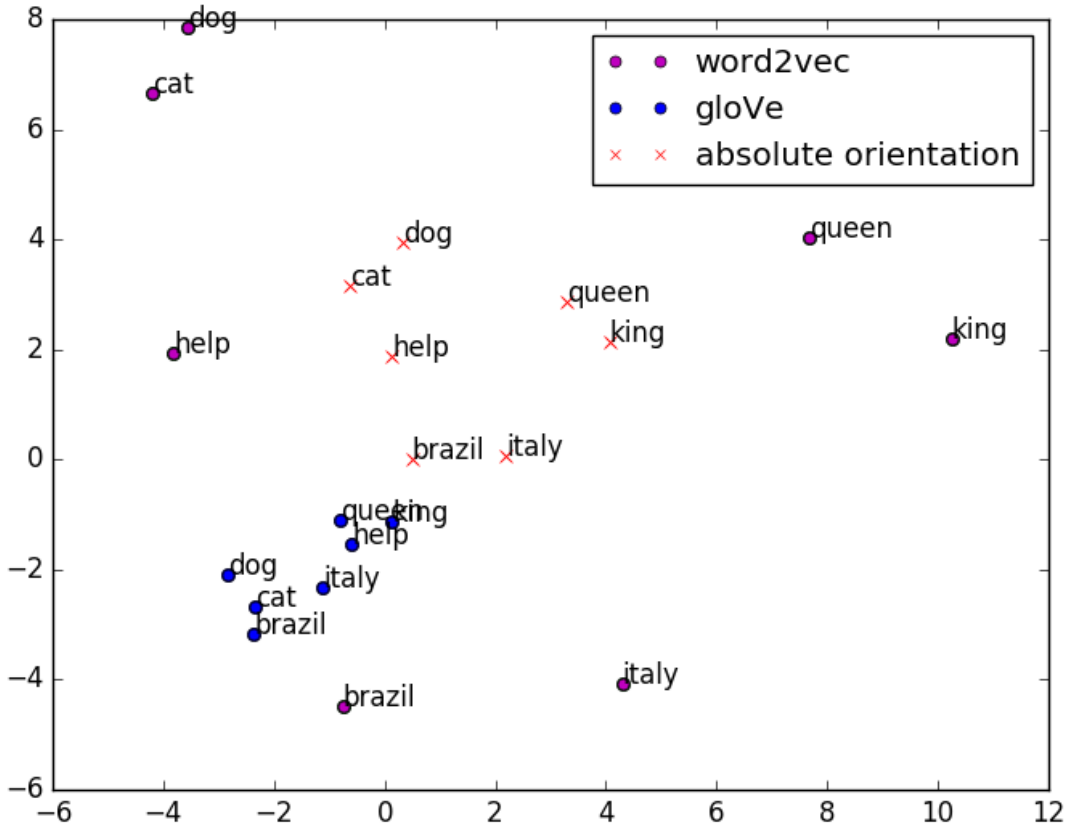

Figure 4: Embeddings of Word2Vec, GloVe, and absolute orientation of GloVe onto Word2Vec

The importance that the error values provide is giving a context of how absolute orientation performs. If the error after performing absolute orientation were greater, then that would indicate that the technique does not align coordinates of high dimensional vectors well. Because the error after absolute orientation smaller, we can conclude that the technique works well to help align high dimensional vector spaces.

Extending this experiment to higher dimension and more words exhibits the same behavior as discussed. Absolute orientation consistently results in moving points from the GloVe embedding vector space closer to the Word2Vec embedding vector space as seen in Figures (5-7). It is important to note that the point cluster resulting from alignment seems to better represent the shape of the initial vector space.

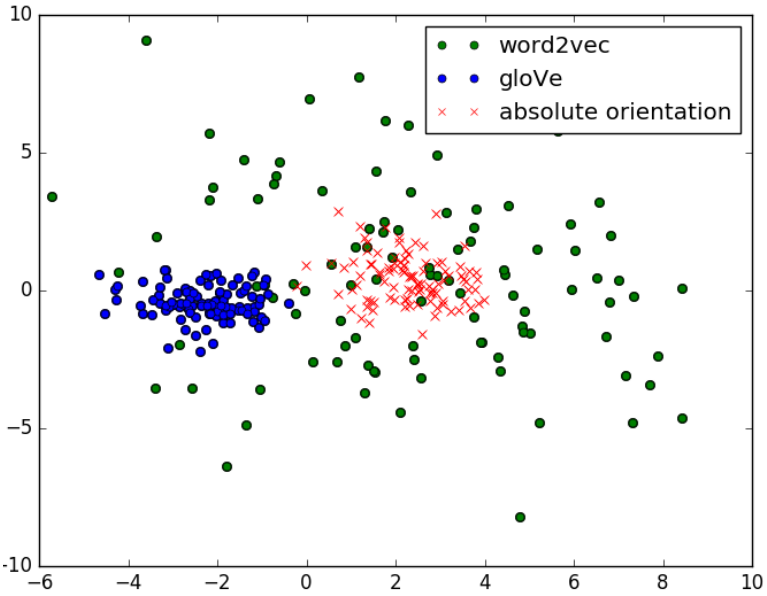

Figure 5: 20 dimensional vector space.

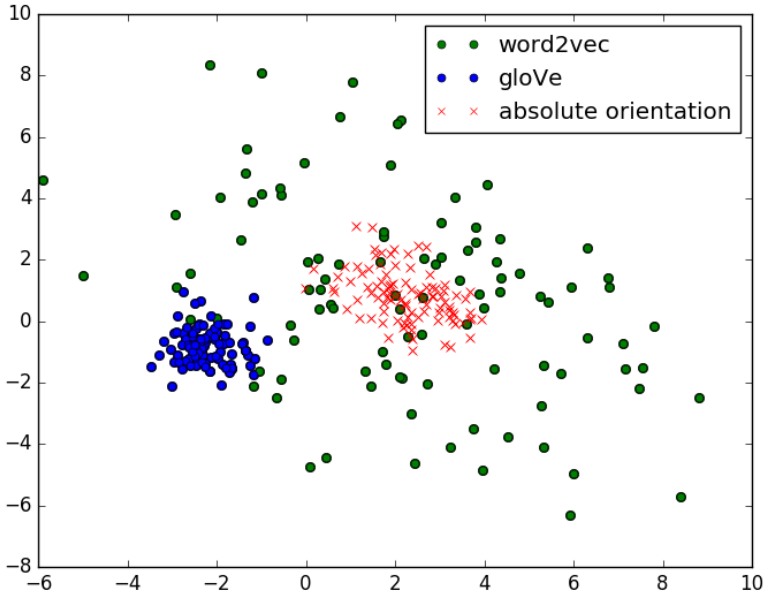

Figure 6: 50 dimensional vector space.

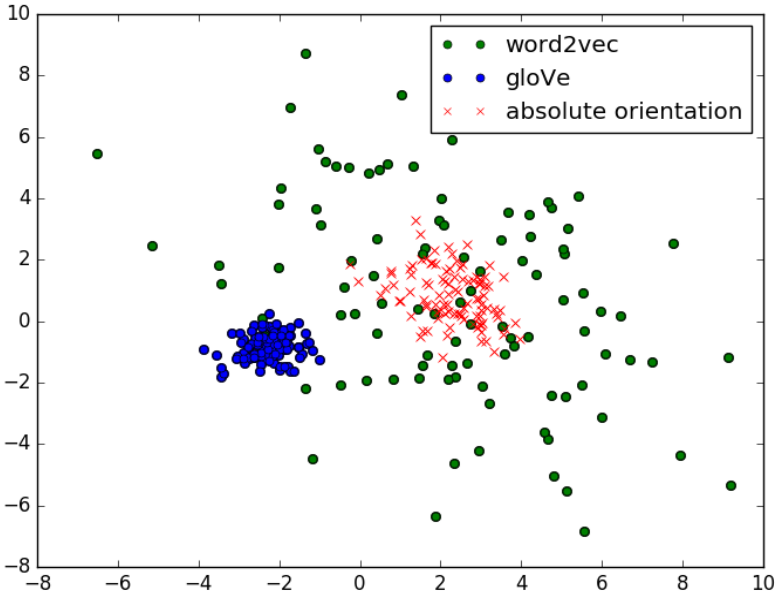

Figure 7: 150 dimensional vector space.

5.3 Measuring Error

The goal for absolute orientation is to better align points between two coordinates. The error results obtained by measuring distances between the vector spaces depict good results. Looking at the error ratio, it is clear that the alignment error is constant with respect to increasing dimensionality. This makes sense because as dimensionality of the vector spaces increase the differences between two embeddings will follow.

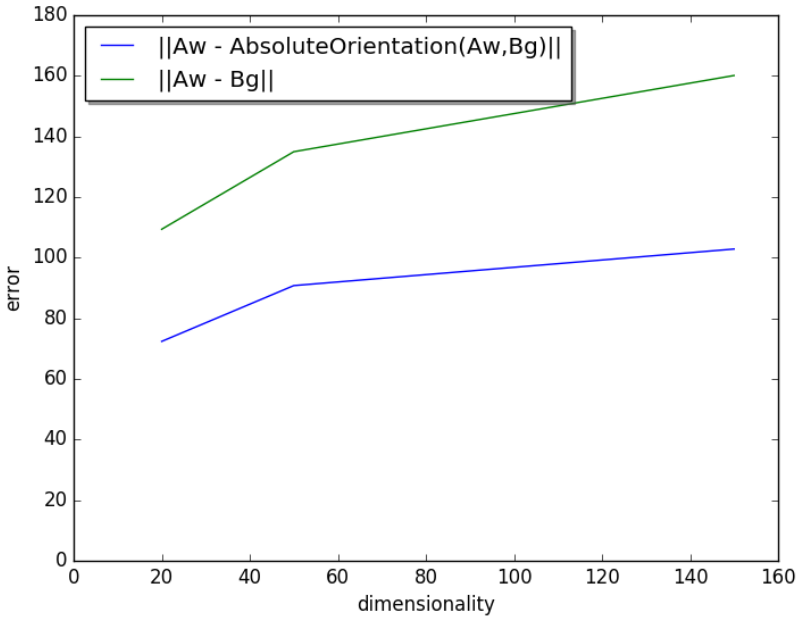

Figure 8: Comparing distance between Word2Vec and GloVe as well as Word2Vec and AbsoluteOrientation(Word2Vec,GloVe).

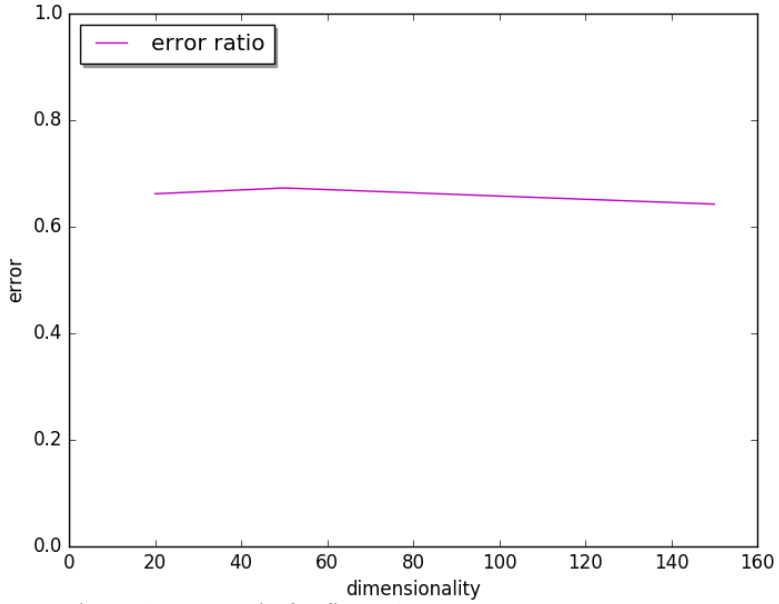

Figure 9: Error ratio for figure 8.

In regards to the numerical value of the error it is unclear as to what constitutes large error. To answer this question, we explored error patterns for absolute orientation. To do so, we began by beginning with two identical 50-dimensional vector spaces and slowly introduced Gaussian noise onto one space. The results from Figures (10-11) reflect the effect noise has on absolute orientation of the respective 50-dimensional embeddings. From this we can see that our error rate for initial embeddings imitates the behavior of adding 1.5-2.0 units of Gaussian noise, whereas error rate for absolute orientation depicts the behavior of adding 0.8-1.2 units of Gaussian noise. As a majority of the points in the respective embeddings are assigned small values for each dimension, typically ranging from [-2,2], the noise introduced is significant. This leads us to conclude that error values over about 100 should be considered high.

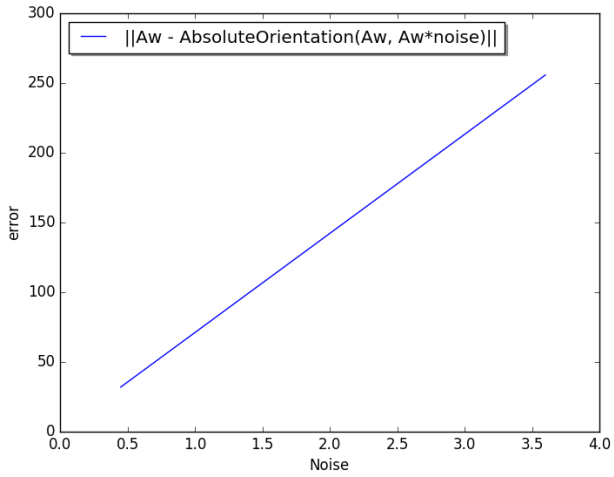

Figure 10: Evaluating noise on Word2Vec

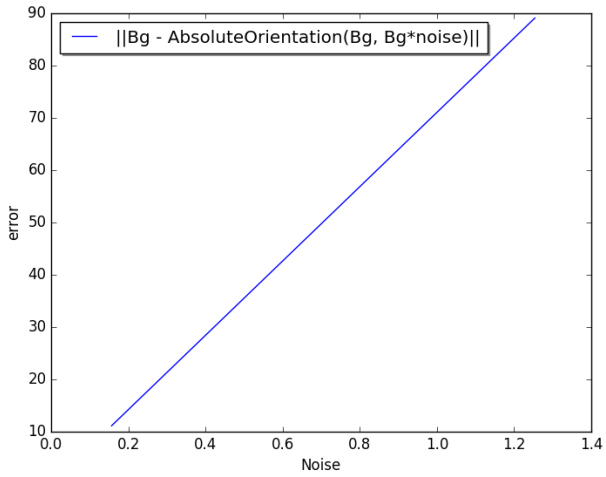

Figure 11: Evaluating noise on GloVe

6 Conclusion

6.1 Summary of Results

We extended the use of absolute orientation, a technique usually implemented on 3-dimensional vector spaces, to higher dimensions. These vector dimensions were obtained through two context-predicting models, Word2Vec and GloVe. Our results indicate that this alignment technique consistently matches points sets obtained from GloVe to be closer to those Word2Vec. As dimensionality increases, embedding results begin to differ as well. This change is reflected in the alignment error of our technique. Finally, absolute orientation guarantees to find the best rotation that exists between two coordinate points, and in high dimensions our results reflect just that.

6.2 Future Work

The goal for performing absolute orientation with two predictive models was to better understand the vector spaces the respective embeddings produced. Applying absolute orientation on higher dimensions proved to help align embeddings. Using this alignment technique, stability

of embeddings can be measured. More specifically, we now have a way to measure the effects of adding or removing words from training dataset.

Scaling points in our vector spaces was an area we did not explore but feel like would help in reducing alignment errors. For many of the results, like in Figure 4, although the rotation correctly aligned points, a scale factor would have brought respective points even closer together.

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop, 2013.

[2] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Quantifying and Reducing Stereotypes in Word Embeddings. CoRR abs/1606.06121 (2016).

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. Accepted to NIPS 2013.

[4] Marco Baroni, Georgiana Dinu, and German´ Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In ACL.

[5] Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12.

[6] Berthold K.P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. Journal of the Optical Society of America A, 5(7):1127–1135, 1988.

[7] Jeff M. Phillips. ICP: A Users Guide. 2007.

[8] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In Proceedings of ACL, 2011.

[9] Bollegala D, Hayashi K, Kawarabayashi Ki (2017) Learning linear transformations between counting-based and prediction-based word embeddings. PLOS ONE 12(9): e0184544. https://doi.org/10.1371/journal.pone.0184544

[10] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d points sets. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5):698–700, 1987.

[11] Jacob T. Schwartz and Micha Sharir. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. International Journal on Robotics Research, 6(2), Summer 1987.

[12] O. D. Faugeras and M. Hebert. A 3-d recognition and positioning algorithm using geometric matching between primitive surfaces. In Proceedings International Joint Conference on Artificial Inteligence, volume 8, pages 996–1002, August 1983.

[13] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America A, 4, April 1987.

[14] Michael W. Walker, Lejun Shao, and Richard A. Volz. Estimating 3-D location parameters using dual number quaternions. CVGIP: Image Understanding, 54(3):358–367, November 1991.