PRACTICAL OBSERVATIONS IN APPLYING NEURAL WORD EMBEDDINGS TO MACHINE TRANSLATION

Michael Seeber
Skymind Labs and GalvanizeU
San Francisco, CA

Abstract - Machine translation is accomplished by creating a translation matrix to map the neural word embeddings from one language to those in another. This paper takes a practitioner's approach, comparing embeddings, discussing obstacles, finding personal motivation for additional methods, and analyzing results.

I. INTRODUCTION

Deep learning, natural language processing, and machine translation are rapidly evolving fields. Often, the best approach to a particular problem is the one that practitioners have found works well in practice.

By attempting to apply neural word embeddings to the task of machine translation, we can start to develop our own intuitions for what works well and where we run into limitations.

I completed this project under the direction of Mike Tamir, PhD at Skymind Labs while a student at GalvanizeU. Skymind Labs is interested in applying state-of-the-art data analysis and machine intelligence, including natural language processing and machine translation. All of the code and reports created by this project are available to view at Github, https://github.com/mike-seeber/lsomantics.

II. OVERVIEW

A. Background

Neural word embeddings are dense representations of words in a low-dimensional vector space, where the word representations are learned by a neural network. They are usually trained on a large corpus such as Wikipedia and "encode general semantic relationships, which are beneficial to many downstream tasks (Ruder, 2016)." A common algorithm for achieving neural word embeddings is word2vec (Mikolov et al., 2013a).

Neural word embeddings can be learned for different languages. Currently, embeddings are learned separately for each language on a corpus of text specific to that language. This results in different embedding spaces for each language learned.

B. Hypothesis

Consider a word, such as "blueberry" and its associated vector representation. We know that another language will have a different word for "blueberry" and a different word vector representation. However, in each language, "blueberry" is a fruit, is blue, and grows on a bush. Underlying our hypothesis, is the belief that the word vectors in each language capture similar semantic relationships between items and ideas, despite using different words and vectors to represent them.

The hypothesis used in our experiments is that we will be able to leverage word vector representations to successfully train a translation matrix that will be able to translate words between two languages.

C. Prior Literature

In (Mikolov et al., 2013b), the authors take a translation matrix approach using neural word embeddings for machine translation. For their specific data and approach, their results include achieving 33% accuracy in translation from English to Spanish. It is worth noting that they trained their translation matrix on the top 5,000 most frequent words in the vocabulary and tested on the next 1,000.

(Xing et al., 2015) largely follows the work of (Mikolov et al., 2013b) but takes an approach where they constrain the word vectors to unit length during the training of the embedding. They also constrain the linear transform as an orthogonal transform. Their results show an increase in accuracy up to around 40%.

(Dinu et al., 2014) also builds on the work of (Mikolov et al., 2013b). They discuss a practical problem of working with high-dimensional spaces where certain vectors, called hubs, become the nearest neighbor to many other vectors. Their approach down-weights these hub vectors when finding the nearest neighbor and results in improved translation accuracy over their baseline by a few points. Their results also include an interesting chart, which shows that accuracy decreases significantly as they attempt to translate words that occur less frequently in the embedding corpus.

III. Pre-trained Embeddings

A. *Embedding Overview*

Training neural word embeddings requires a large corpus and a lot of training time. There are also various algorithm and hyper-parameter options at the discretion of the modeler. At the onset of this project, I was aware of two sources to obtain pre-trained embeddings and decided to use those as a jumping off point for the machine translation task. As of now, I have used pre-trained embeddings from four different sources and will discuss each in this section.

B. *Kyubyong*

Kyubyong Park provides links to pre-trained word embeddings for 29 languages on Github, https://github.com/Kyubyong/wordvectors. The vocabulary size is typically near 50,000 but sometimes lower, and the vectors are size 300. He has vectors available for both word2vec and fasttext algorithms, but I used word2vec. His training code is also available on Github, and you can see exactly how he performed the embeddings.

C. *Polyglot*

Rami Al-Rfou provides embeddings for 136 languages at https://sites.google.com/site/rmyeid/projects/polyglot. The training methodology is discussed in (Al-Rfou' et al., 2013). The vocabulary size is typically either 100,000 or lower, and the vectors are size 64. He also provides code for those interested in training similar embeddings on their own data.

D. *Fasttext*

Facebook research provides embeddings for 90 languages on Github, https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md. The methodology is discussed in (Bojanowski et al., 2016). The vocabulary size is over 1,000,000 words and the vectors are size 300. The Fasttext pre-trained embeddings had the largest file sizes that I had to work with due to the greater vocabulary size. At this point, I have downloaded and compared embeddings for 3 of these languages with additional research planned.

E. *Zeroshot*

Georgiana Dinu provides embeddings for English and Italian at http://clic.cimec.unitn.it/~georgiana.dinu/down/. The embeddings were learned using

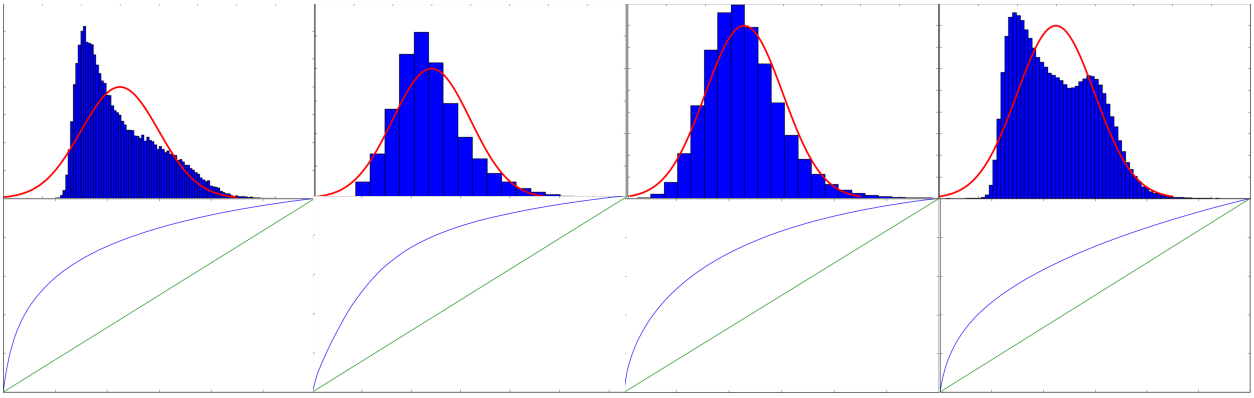

Figure 1

Left-to-Right:	Kyubyong French, Polyglot English, Fasttext German, Zeroshot Italian
Top:	Distribution of L2 norms of the vector embeddings.
Bottom:	Principal components analysis on the embeddings; explained variance (y-axis) versus number of eigenvectors (x-axis).

word2vec, the vocabulary size is 200,000, and the vector size is 300.

F. *Embedding Comparisons*

In addition to looking at the vocabulary size, vector length, and training details, I calculated both the norms of the vectors and their principal components analysis (PCA). Figure 1 shows a sample of corresponding plots for a single language for each type of embedding. I computed "isotropy" as the area between the blue curve of the PCA plots and 1 (top of the plot), divided by 0.5. Isotropy gives us a measure between 0 and 1 for how well the embedding utilizes each of the dimensions in the embedding space. If the embedding fully utilizes the dimension space, we should not be able to reduce the vector dimensionality using PCA and the isotropy measure should be 1.

I first started by performing the exploratory data analysis on the Kyubyong vectors. As you can see in Figure 1, the Kyubyong vector norms are right-skewed. Based on past experience, we anticipated that the distribution of norms would look closer to a normal distribution. This is one reason I began looking at the next embedding source, Polyglot. It is also worth noting that the average isotropy was 0.39, with a max of 0.57 across the 29 languages.

The Polyglot vectors are embedded in fewer dimensions (64), have the most languages, and have many different vocabulary sizes. Upon visual inspection, I observed greater skewing of the distribution of vector norms as the vocabulary size decreased. For larger vocabularies, the distribution appeared more normal than the Kyubyong vector norms. A potential hypothesis I formed is that this could be due to the lower dimensionality of the embedding space relative to the vocabulary size. However, it could also be caused by any of the other differences in the embedding procedures. The average isotropy for Polyglot was lower at 0.27, and had a max of 0.60 across 136 languages.

Kyubyong and Polyglot were the two pre-trained embedding sources that I was aware of when I began this project. However, I wanted to see if a larger vocabulary would result in more normal vector norms and higher isotropy. In my research, I discovered the Fasttext pre-trained embeddings from Facebook with more than 1 million words. The analysis shows that the Fasttext vector norms, like

Polyglot, are less skewed than Kyubyong, while having the higher number of dimensions, 300, like Kyubyong. While certainly inconclusive, this new information doesn't detract from my prior observation that the distribution of the vector norms is perhaps impacted by both the size of the vocabulary and embedding space. Fasttext also had a higher average isotropy of 0.49 with a max of 0.51 across 3 languages.

While there is a lot more to explore and understand as it relates to the embedding of a single language, the attempt of this project is to train a translation matrix. I decided to begin that process while leveraging the Fasttext vectors, which had the largest vocabulary, 300 dimensions, less-skewed distribution of vector norms given the dimensionality, and higher isotropy. Later, as I was working through the methodology and attempting to achieve meaningful results, I discovered the Zeroshot vectors.

The Zeroshot vectors were embedded in 300 dimensions and have the highest isotropy of 0.59 for both languages. The distributions of vector norms also included an unanticipated bi-modal shape shown in Figure 1.

You can find the many hundreds of plots for the L2 norm distributions and principal components analyses, as well as additional statistics and data for all of the embeddings and languages located in the reports folder of the projects Github repository.

IV. Bilingual Dictionaries

In order to train and test a translation matrix, we need access to a bilingual dictionary that allows us to map the words and vectors from one language to the corresponding words and vectors in the other language. As a practical matter, sourcing the required translations was the most limiting factor in the work I completed at the time of this paper.

Google Translate has a developer API that allows you to submit a word and return the translation in a foreign language of your choice. This was the most promising option for fast, simple translations between languages of our choice. However, we found the cost for the developer API to be too prohibitive for the large vocabulary sizes we required for this project.

In my search, I was able to find and download a free English to Russian text file from http://www.dict.cc. Unfortunately, I found the vocabulary size to be less than 20,00, which was far less than our vocabularies that exceeded a million words.

Finally, I found that I was able to request translations from Google Translate using a url and Python's Urllib. This option was free, but quite slow. Over the course of more than a week, I used several virtual machines from Amazon Web Services to obtain full vocabulary translations for English to Russian, Russian to English, English to German, and German to English.

V. Translation Matrix

A. Methodology

In this section we formalize the notion of a translation matrix. Let $v_{1,i}$ be the i^{th} word vector from the first vocabulary and let $v_{2,i}$ be its correctly translated word vector from the second vocabulary. The objective is to find a translation matrix, T, such that the product $v_{1,i} T$ approximates $v_{2,i}$. I treat this as a supervised learning problem by training T on a subset of n_{Train} number of words from our vocabulary and testing on n_{Test} number of different words. During training, I seek to minimize the mean squared error:

$$min_T \sum_{i=1}^{n_{Train}} ||v_{1,i} T - v_{2,i}||^2$$

The mean squared error loss function makes sense under the hypothesis that neural word embeddings of the same language are equivalent under a change of basis. For example, this means that if we run word2vec on the same training set but

	Embedding	Translation	Train/Test Size	Train/Test Sampling	Max Neighbors
Experiment Label					
A	Fasttext	Enlish -> Russian	5,000 / 1,500	Random sample	944,211
B	Fasttext	English -> Russian	5,000 / 1,500	Random sample most frequent words	944,211
C	Fasttext	English -> German	5,000 / 1,500	Random sample	1,137,616
D	Fasttext	English -> German	5,000 / 1,500	Random sample most frequent words	1,137,616
E	Zeroshot	English -> Italian	5,000 / 1,869	Stratified sample word-frequency buckets	200,000

Table 1

initialize the learned parameters slightly differently then we expect to end up with equivalent matrices up to a change of basis. This suggests that radially symmetric loss terms like L2 and cosine are appropriate.

B. Optimization

I use Python's Keras package running Tensorflow in the back-end to train *T*, and leverage the "Adam" optimizer, which is a first-order gradient-based optimization technique (Kingma, Ba, 2014).

C. Unambiguous Translations

For the Fasttext embeddings, for which I created bilingual dictionaries using Google Translate, I decided to train and test on words where I had direct, unambiguous translations between the two languages. I determined this by checking to see if a word, when translated into the other language and then translated back again to the first language, would yield the original word. For English to Russian, I had around 50,000 words that met this bi-directional translation criteria, and for English to German I had around 150,000 such words.

VI. Experimental Results

A. Experimental Design

Table 1 lists each experiment, labeled *A* to *E*, with its key attributes. It is worth noting that (Dinu et al., 2014) show no improvement in accuracy when their training size is greater than 5,000 words, and my own ad hoc experimentation (not shown) suggested similar conclusions.

B. Accuracy

To gauge the performance of the translation matrix, *T*, I wanted to measure how often the correct translation could be recovered after applying *T* to the vector of a given word.

Consider $p_{2,i}$ to be the predicted vector for the i^{th} word obtained from the product $v_{1,i}T$. I use cosine similarity to find the nearest neighbor to $p_{2,i}$ among all vectors in the second language $v_{2,:}$ and use it as the predicted word. Table 1 shows the number of neighbors considered in each experiment. For example, experiment *A* searched for the nearest neighbor among our full Russian vocabulary of 944,211 words.

Using the nearest neighbor approach, I obtained the following accuracies for each of the five experiments on the test data:

- *A*: 3.9%
- *B*: 46.4%
- *C*: 21.9%
- *D*: 63.6%
- *E*: 27.9%

My analysis suggests that the accuracy is effected just as much by our ability to learn *T* as by the embedding processes ability to reliably encode each word in the embedding space. For example, consider a low-frequency word that only

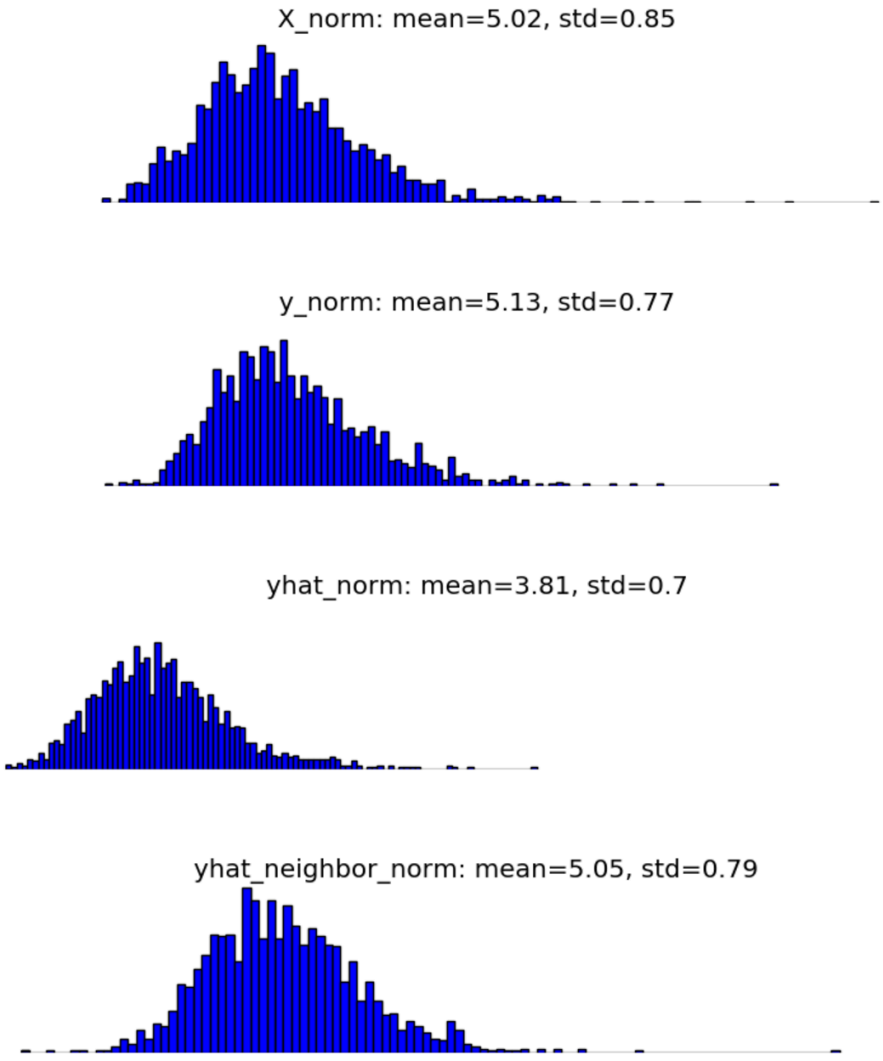

Figure 2

Distribution of L2 vector norms for the test vectors of Experiment *C*.
X_norm: English
y_norm: German
yhat_norm: German predictions obtained from *XT*
yhat_neighbor_norm: Nearest German neighbors to *XT*

shows up one time in the corpus for each language. The embedding procedure will have a more difficult time reliably embedding this word than it would a word that appears many times in the corpus. Since this low frequency word is embedded separately for each language in its respective embedding space, we can assume it is less likely to get a reliable embedding in each space than a more frequent word. This obviously has implications for our ability to find the correct nearest neighbor, and as (Dinu et al., 2014) also showed, translation accuracy decreases for less frequent words. Similarly, this also has implications for training *T*. To the extent we use less frequent words with unreliable embeddings as the supervision to the learning process, *T* will be less able to

generalize to words not seen during training. Our own experiments *B* and *D* show 40%+ improvements in accuracy over *A* and *C* when we train and test on the most frequent words rather than simply on random words.

C. Vector Norms

Figure 2 shows the distribution of vector norms for the test vectors of Experiment *C*. Notice that the distribution of *yhat_norm* (obtained from the L2 vector norms of the inner product of *X* and *T*) is shifted to the left. This result was observed in all of our experiments. In other words, one impact of *T* on the vectors is that of reducing their magnitude.

During development, I also tried using euclidean distance in place of cosine similarity in the nearest neighbor procedure. However, we have just shown that the vectors predicted by *T* have lower magnitudes, meaning they are closer to zero. As a result, I observed extreme amounts of hubness as the nearest neighbors for all of the test words were from among a handful of words that had low magnitudes themselves. In other words, the predicted vectors are closer to zero, and, hence, closer to a small number of hub vectors with low magnitude, than they are to the vectors of correct translation. The result was accuracies observed near 0%.

These observations on hubness and reduced prediction magnitudes in my own experiments provide some personal motivation for understanding the (Xing et al., 2015) approach of normalizing the word embeddings and training an orthogonal transform, because an orthogonal transform will not change the magnitude of the predicted vectors. In my reading of their paper, I took away the argument for considering the way in which the matrix will be used, i.e. with cosine similarity, into account by training in a consistent manner. I think their approach makes sense both from the motivation and logic they provide and in light of the diminishing magnitudes demonstrated above.

D. Translation Matrix Isotropy

I computed the isotropy of *T* for each experiment, which ranged between 0.32 and and 0.47. These values less than one lead me to believe that a translation with similar performance could be achieved with fewer parameters than the 90,000 used in our 300 by 300 matrix *T*. However, I haven't conducted any experiments yet in this direction.

Please note that the PCA explained variance plots, vector norm plots, isotropy values, and additional data for each experiment can be found in the reports folder of the projects Github repository.

V. Conclusion and Future Work

Running these experiments highlighted many of the nuances involved in working with neural word embeddings and provided some practical intuitions for comparing different embedding spaces. By analyzing the vector norms of the translation, we observe the diminishing magnitudes and develop a better understanding of hubness and metric choice in the nearest neighbor algorithm.

Overall, I discovered for myself how this approach for machine translation works much better for higher frequency words, which is an important understanding that hadn't jumped out as saliently to me during prior studies.

As a practical matter, we were constrained in time and complexity on the number of languages we translated for this project. Having now worked through a complete cycle from start to finish, we see that in future experiments we can constrain the number of translated words to only those needed for training and testing.

Finally, we saw very different results depending on our experiment choices, and even different results between languages

when following the same procedure. In future work, I would like to translate between several additional languages and see if there are additional trends, perhaps based on how linguistically similar we know two languages to be.

References

[1] Sebastian Ruder, *On word embeddings - Part I*, 2016, http://sebastianruder.com/word-embeddings-1/

[2] Tomas Mikolov et al., *Distributed Representations of Words and Phrases and their Compositionality*, 2013a, https://arxiv.org/pdf/1310.4546.pdf

[3] Tomas Mikolov et al., *Exploiting Similarities among Languages for Machine Translation*, 2013b, https://arxiv.org/abs/1309.4168

[4] Chao Xing et al., *Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation*, 2015, http://www.aclweb.org/website/old_anthology/N/N15/N15-1104.pdf

[5] Georgiana Dinu et al., *Improving zero-shot learning by mitigating the hubness problem*, 2014, https://arxiv.org/abs/1412.6568

[6] Rami Al-Rfou' et al., *Polyglot: Distributed Word Representations for Multilingual NLP*, 2013, https://arxiv.org/pdf/1307.1662.pdf

[7] Piotr Bojanowski et al., *Enriching Word Vectors with Subword Information*, 2016, https://arxiv.org/abs/1607.04606

[8] Diederik P. Kingma, Jimmy Lei Ba, *Adam: A Method for Stochastic Optimization*, 2014, https://arxiv.org/pdf/1412.6980v8.pdf