Comparing Hebbian Semantic Vectors Across Language

Kiran Vodrahalli

Princeton University

May 12, 2015

1. Introduction

Meaning is a relatively ill-defined term when discussing language. We could suppose that the meaning of a word is a map from seen text or spoken sound to a concept existing in the physical world. For instance, a "tree" refers to the leafy, tall green-and-brown thing outside of my window. But the difficulty becomes more apparent as we attempt to describe what an "ideal" is, or perhaps "love".

How then, do our brains represent meaning in such a manner that people understand what other people are talking about, particularly when discussing abstract concepts? Furthermore, is it possible to use processes similar to those of the brain to represent abstract concepts in a computer? In fact, ideas for solving this representation problem from computer science are justified approaches in neuroscience as well.

1.1. Semantic Vectors

One prevalent idea from natural language processing is the semantic vector approach (also referred to as vector space model, or VSM), which hypothesizes that the meaning of words can be expressed as vectors in $\mathbb{R}^d$, typically for $d \in [10^2, 10^3]$. The origin of this idea is from the late 1990s when word-context matrices were defined in the field of information retrieval. The goal is to exploit the distributional hypothesis of meaning, which roughly says that words which have similar co-occurrence patterns in a corpus have similar meaning. For instance, the words "dog" and "cat" might appear in a lot of similar contexts: "The owner petted the dog/cat.", "The owner fed the dog/cat.", and so on. These popular pets are both domestic mammals of similar size and are in a rough sense similar, at least compared to whales or cars.

The original approach was to build a word-context matrix for a corpus: rows are words, columns are "contexts", essentially settings in which the words appear in the corpus. The co-occurrence frequencies go in the entries, and are usually normalized, smoothed, and transformed. Typically some dimension reduction process (for instance, singular value decomposition (SVD)) is applied to this matrix, and the vectors from the resulting process are termed semantic vectors. Simple linear algebraic operations are then applied to these vectors to solve linguistic problems. As an example, the cosine distance between two vectors is often applied to tell how similar they are. k-means clustering is also often used to find groups of words which are similar. There is a large literature on the application of word-context matrices to topic modeling, word sense disambiguation, and other tasks. More information about vector space models is presented in detail in the comprehensive survey by Turney and Pantel [Turney2010].

Another approach to creating semantic vectors came about from Bengio's study of neural networks intended to learn a language model of a corpus. Words in a fixed-size vocabulary $\mathcal{V}$ are represented as one-hot vectors and are fed as inputs into a neural net. An intermediate layer $\mathcal{C}$ represents each of the words with a small number of units, which are then fully connected to a hidden layer $\mathcal{H}$. $\mathcal{H}$ finally produces a softmax output layer of size $|\mathcal{V}|$, where each unit represents the probability that the word correponding to it occurs next in the corpus [Bengio2003]. More recently, Mikolov et. al. developed a much simpler and easier to train log-linear model (known as the skip-gram model) the sole goal of which is to learn semantic vector representations. A central idea in this approach is to throw out the complexity of a fully-connected neural net with nonlinearities, and instead use a barebones structure to learn the vectors [Mikolov2013]. Somewhat surprisingly, this method (known as word2vec) works a lot better than other word vector representations in the word analogy task. A better model known as GloVe was introduced the following year by Pennington et. al. [Pennington2014]. However, the results were entirely empirical for all of these approaches. A few months ago (March 2015), Professor Arora (who is at Princeton) published a simple unsupervised learning model with empirical guarantees as to the performance of word vectors on an analogy task [Arora2015].

From the point of view of the computer scientists, semantic vectors applied to natural language problems in machine learning have been wildly successful empirically. The theoretical grounding behind why this approach works is an exciting field that is developing rapidly. However, while some semantic vector approaches to representing meaning do rely on the architecture of a neural net, these neural nets are not biologically inspired. Typically, the cost function relates to a language model and the objective is to learn word vectors which maximize the probability of predicting the correct next word. Our goal in this paper is to introduce and evaluate a more clearly biologically-justified neural network which learns semantic vectors in an unsupervised fashion.

1.2. Semantic Vectors in Neuroscience

Semantic vectors have also attracted interest in the neuroscience community as an approach to correlating fMRI, EEG, and MEG data from a person with what a person is actually thinking. First we must ask if fMRI, EEG, or MEG data actually correspond with semantic meaning. Pulvermüller did some work to demonstrate that the brain encodes representations which distinguish words in separate classes: For instance, grammatical function words, concrete content words, and words referring to visual stimuli. He analyzes fMRI data as well as temporal dynamics (through EEG and MEG data) to come to these conclusions. fMRI studies revealed that cortical areas devoted to motor function were activated upon being presented with words associated with motor movement, and visual perception cortical areas were activated upon being presented with visually-associated words [Pulvermüller1999]. Furthermore, Pulvermüller's work supports a Hebbian model of word representation, which roughly summarized is that every concept or word has a separate neuronal assembly. A neuronal assembly refers to a group of cells that are strongly connected, and activate together ("fire together, wire together"). Particularly, Pulvermüller observed that concrete content and abstract function (i.e., grammar) words had neuronal assemblies which were lateralized differently across both brain hemispheres. Laterality refers to the number of cross-hemisphere connections. Abstract function words had assemblies with high degree of laterality while concrete content words had a low degree of laterality. Pulvermüller also advocates support for the hypothesis that grammatical knowledge is represented in connections between neuron assemblies and in the activation dynamics exhibited by the cell assemblies - in other words, it is not only which neurons activate, but also the intensity of their activation which determines representations of meaning [Pulvermüller1999].

Further evidence for neural signatures of concepts embedded in brain activity data comes from multidiscplinary work on the intersection between neuroscience and machine learning. Tom Mitchell, Robert Mason, Svetlana Shinkareva, Vincente Malave, and Francisco Pereira collaborated on a project to learn which neural activation patterns correspond to various semantic categories by using classifiers like logistic regression to fit the data. They found that neural representations of concepts contain perceptual and motor information relevant to that concept. Also notably, these neural representations span all four lobes in both hemispheres as well as the cerebellum. This approach also was able to classify the category of word being read, and even more impressively, the precise word that was in a person's mind at a given time [Just2008].

More recent work has corroborated evidence for these neural encodings of language meaning by comparing semantic vectors to brain activity data, and even incorporating brain activity data in the construction of semantic vectors. Fyshe et. al. (2013) developed a vector space model based on a large (16 billion word) corpus. Vectors in this corpus were 2000 dimensions: The first 1000 dimensions were termed "Document" features and the last 1000 were termed "Dependency" features. Document features are built from co-occurence data with a single word's presence in a document, while Dependency features are built from co-occurence data with contexts of the type "eat ___" or "a ___ television screen", where the blank represents the word in question. The objective was to model adjective-noun phrases. This paper also analyzed the resulting semantic vectors with respect to brain activity data. Specifically, Fyshe et. al. analyzed the following task: A person is presented with a phrase while MEG data is collected. Then, the authors formed a training set $\{(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x}$ is input and $\mathbf{y}$ is the label. The input was taken to be the averaged MEG data for a subject, while the label is the sentence associated with the data. They then defined a mapping from sentences to their VSM-based semantic vectoral representation of the phrase and trained a regressor to predict semantic vector representations from MEG data, which when trained on 36 phrases was able to predict the correct semantic vector representation for 2 sentences in the test set with 0.9440 accuracy [Fyshe2013].

In a second paper by Fyshe et. al. (2014), brain activation data recorded while people read words is incorprated into building the semantic vector space model. They introduce a new matrix factorization method called JNNSE which creates a VSM that is more correlated to word semantics, produces semantic vectors that are more predictable from brain activity data across recording technologies (i.e., fMRI, MEG), and maps semantic concepts directly onto the brain. In other words, there is a mapping between brain representations of meaning and a vector living in $\mathbb{R}^n$ which also represents the same information. Fyshe et. al. also suggest that their findings indicate that there is semantic information available in brain activation data that is not present in corpus data which text-based VSMs lack [Fyshe2014].

Therefore, there seems to be sufficient evidence for a semantic vector-like representation of concepts in the brain. Following Pulvermüller's support of the Hebbian hypothesis, we will build a Hebbian network-based semantic vector model as a greatly simplified representation of what may be occurring in the human brain.

1.3. Evaluating Semantic Vectors

Most methods are concerned with the performance of word vectors on tasks within a single language; for instance, a word analogy task or a sentiment analysis task. The aspect of word meaning that semantic vectors are evaluted on capturing is primarily relational within a single language. For instance, we might produce the k closest vectors according to some distance metric to a given semantic vector, and check a thesaurus or a concept database to evaluate precision and recall - do the surrounding vectors show up as synonyms, or at least as highly related words? Another example of a single language task we can use to evaluate word vector performance is analogy. We might desire word vectors to provide a mapping from words to $\mathbb{R}^n$ solving the equation $vec(\text{“king”}) - vec(\text{“man”}) + vec(\text{“woman”}) = vec(\text{“queen”})$. Arora et. al. show that word vectors which solve an equation of this type actually encode distributional properties along the lines of

$$\frac{\mathbf{P}\{\chi|\text{king}\}}{\mathbf{P}\{\chi|\text{man}\}} \approx \frac{\mathbf{P}\{\chi|\text{queen}\}}{\mathbf{P}\{\chi|\text{woman}\}} \tag{1}$$

where χ is a word [Arora2015].

At the very least, most papers publishing new approaches to building word vectors within the past few years assessed performance on single-language tasks.

Sutskever et. al (2014) apply word vectors to machine translation between English and French. Sequence-to-sequence learning works by learning weights for an LSTM-based neural network which translates from one language to another by compressing natural English language into vector representations, which are then passed to a second LSTM network which extracts French words out of it, indicating that the community recognizes that semantic vectors should encode meaning transferable across language [Sutskever2014]. Hassan et. al. (2012) use multilingual representations of words to improve quantification of the strength of semantic connections between textual units (i.e. semantic relatedness). However, these papers do not evaluate methods which produce word vectors by evaluating how similarly they perform across languages. Rather, the goals of these approaches is either to improve machine translation algorithms or make use of multilingual information to improve performance on some linguistic task [Hassan2012]. To the knowledge of this author, there have been no published approaches which evaluated a given semantic vector representation by assessing its cross-language performance.

In this paper, we will evaluate a method of creating semantic vectors by assessing the similarity of their performance across English and French. The justification for this evaluation metric stems from the dictum that "meaning is invariant across language" for the most part. Therefore, since semantic vectors are supposed to represent meaning, language structure needs to be accounted for in the construction of word representations so that interactions between semantic vectors do not change across languages.

1.4. The Task

The goal of this paper is to build and evaluate a Hebbian network vector space model of language. Our parallel texts are the English and French translations of J.K. Rowling's *Harry Potter and the Philosopher's Stone* ([Rowling1997], [Ménard1998]). We learn semantic vectors for matching subsets of the words in the text and introduce some new metrics for assessing word vector performance that depend on cross-lingual corpora.

2. Learning Semantic Vectors with Hebbian Learning

Given a corpus, we wish to define a neural network that applies Hebbian learning to learn low-dimensional representations of the semantics of each word. Recall that the main principle of Hebbian learning is "fire together, wire together". Neurons that are active at the same time strengthen their connection, while neurons that are not active simultaneously weaken their connection.

2.1. The Hebbian Network

In Figure 1, we present an illustration of the network we use to learn the word vectors. It contains an input layer representing the vocabulary $\mathcal{V}$ and a single sigmoidal hidden layer $\mathcal{H}$ with k-winner-take-all inhibition. $\mathcal{V}$ represents a one-hot encoding of all the words in the vocabulary: Each word is assigned an index from 1 to $|\mathcal{V}|$. $\mathcal{H}$ is fully connected to $\mathcal{V}$. We denote these weights between the two layers as $\mathcal{W} \in \mathbb{R}^{|\mathcal{H}| \times |\mathcal{V}|}$, where these weights are updated in a Hebbian fashion. The basic scheme is that we take a sliding window of some size $2r + 1$ across the corpus. For all words present in the same window, we "fire" the neuron and compute the activations of the hidden layer. Letting $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$ be the activation input and $\mathbf{y} \in \mathbb{R}^{|\mathcal{H}|}$ be the activation output, we have

$$\mathbf{y} = \sigma\left(\mathcal{W}\mathbf{x}\right) \tag{2}$$

where σ is the sigmoid function $\sigma(\mathbf{z}) = \frac{1}{1+e^{-\mathbf{z}}}$, where the function is applied elementwise in the vector setting. These inputs and outputs are then used to calculate the weight updates. In the k-winner-take-all setting, the equation is slightly different. Let $\hat{\mathbf{y}}$ be $\mathbf{y}$ in sorted order from largest to smallest, and $\mathbf{a}_i$ denote the i^{th} element of the array $\mathbf{a}$.

$$\mathbf{y}_j = \begin{cases} \sigma\left(\mathcal{W}\mathbf{x}\right)_j & \text{if } \sigma\left(\mathcal{W}\mathbf{x}\right)_j \geq \hat{\mathbf{y}}_k \\ 0 & \text{otherwise .} \end{cases} \tag{3}$$

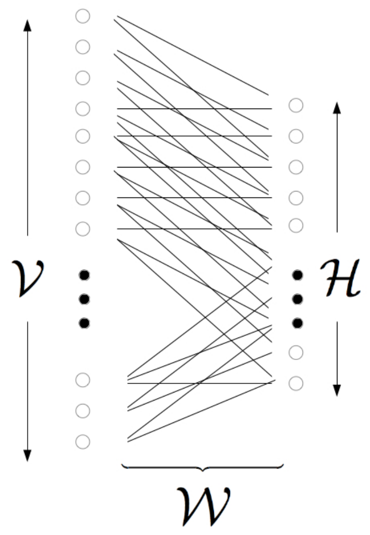


Figure 1: *Diagram of the Hebbian Network*

2.2. Theoretical Interpretations of Hebbian Learning

The basic Hebbian learning update is given by

$$\triangle \mathcal{W}_{ij} = \eta \mathbf{y}_j \mathbf{x}_i \tag{4}$$

where $\mathbf{x}_i$ is the activation of a unit corresponding to the i^{th} word in $\mathcal{V}$, $\mathbf{y}_j$ is the activation of a unit corresponding to the j^{th} unit in $\mathcal{H}$, and $\mathcal{W}_{ij}$ is the weight of the edge between $\mathbf{x}_i$ and $\mathbf{y}_j$. η is the learning rate.

A more stable learning rule is Hebbian learning with weight decay, which is modified slightly from the original Hebb rule:

$$\triangle \mathcal{W}_{ij} = \eta \mathbf{y}_j (\mathbf{x}_i - \mathcal{W}_{ij}) \tag{5}$$

We now give a theoretical justification for why these rules work.

2.2.1 Hebbian Learning as PCA

To provide intuition, we consider the simple case of a linear threshold neuron: Essentially, Figure 1 where $|\mathcal{H}| = 1$ and σ is a linear function instead of sigmoid. Here we denote $\mathbf{y}$ as y since there is only one output unit, and $\mathcal{W}$ as simply $\mathbf{w}$, since again, there is only one output. We claim that in this setting,

Theorem 2.1. *The first Hebbian learning update (Equation 4) computes the first principal component of the data matrix $\mathcal{X}$ for a linear-threshold neuron.*

Here, $\mathcal{X}$ can be considered as a concatenation columnwise of all inputs $\mathbf{x}$ to the network. As a reminder, the first principal component can be interpreted as the eigenvector associated with the largest singular value after singular value decomposition (SVD) of $\mathcal{X}$. Another interpretation is as the direction with most variance in the data.

Following the treatment in Chapter 4.5 of [O'Reilly2000], we show that Theorem 2.1 is indeed the case:

Proof. First, we consider the weight-change rule analagously to velocity in physics [Seung2015], and make a simple average velocity approximation for the weights by assuming that the number of input patterns is $\frac{1}{\eta}$:

$$\triangle \mathbf{w}_i \approx \mathbf{E}_t\left[\mathbf{x}_i y\right] \tag{6}$$

Substituting Equation 2 for y,

$$\begin{aligned} \triangle \mathbf{w}_i &\approx \mathbf{E}_t\left[\mathbf{x}_i\left(\mathbf{x}\cdot\mathbf{w}\right)\right] \\ &= \mathbf{E}_t\left[\mathbf{x}_i\mathbf{x}\right]\cdot\mathbf{E}_t\left[\mathbf{w}\right] \\ &= \mathcal{C}_i\cdot\mathbf{E}_t\left[\mathbf{w}\right] \end{aligned} \tag{7}$$

where $\mathcal{C}$ is interpreted as a correlation matrix if the inputs $\mathbf{x}$ have mean 0 and unit variance. Re-writing in vector notation, we have that

$$\triangle \mathbf{w} \approx \mathcal{C}\mathbf{w} \tag{8}$$

where $\mathbf{w}$ is a valid approximation of $\mathbf{E}_t\left[\mathbf{w}\right]$ if we assume that the weight matrix changes slowly. Now we have the formulation of the power iteration algorithm to find the largest eigenvalue (taking $\mathcal{C}^t\mathbf{w}$ for t large and normalizing will give the largest eigenvector of $\mathcal{C}$ assuming that $\mathbf{w}$ as a non-zero component in the direction of the largest eigenvector). Thus as $t \to \infty$, $\triangle\mathbf{w}$ will converge to a vector in the direction of the largest eigenvector, and the update will keep stepping in the direction of the principal component, which will dominate the finite number of steps in directions other than the principal component. $\square$

2.2.2 Hebbian Learning as CPCA

However, Equation 4 diverges as $t \to \infty$ since we keep stepping in the direction of the strongest principal component. This problem inspires us to come up with Equation 5, which includes a weight-decay term to ensure that $\mathbf{w}$ converges. We can see this regularization forces convergence by assuming that the output is always active, i.e. $y = 1$. Then, Equation 5 becomes

$$\triangle \mathbf{w} = \eta \left(\mathbf{x} - \mathbf{w}\right) \tag{9}$$

Again using the average velocity approximation and letting $\hat{\mathbf{w}}$ be the value $\mathbf{w}$ converges to, we get that

$$\begin{aligned} \triangle \hat{\mathbf{w}} &\approx \mathbf{E}_t\left[\triangle \hat{\mathbf{w}}\right] \\ &= \eta \mathbf{E}_t\left[\mathbf{x} - \hat{\mathbf{w}}\right] \\ &= \eta\left(\mathbf{E}_t\left[\mathbf{x}\right] - \hat{\mathbf{w}}\right) \end{aligned} \tag{10}$$

and therefore $\hat{\mathbf{w}} = \mathbf{E}_t\left[\mathbf{x}\right]$ when $\triangle \hat{\mathbf{w}} = \mathbf{0}$. Thus with regularization, the weight vector $\mathbf{w}$ converges to something sensible with constant activation, the average of the inputs [Seung2015].

We can also interpret this learning rule in terms of Conditional PCA, or CPCA [O'Reilly2000]. By conditioning on a given input pattern occurring at time t, and treating activations as probabilities, we can write the update of Equation 5 as

$$\begin{aligned} \triangle \mathbf{w}_i &= \eta\left(\sum_t \mathbf{P}\{y|t\}\mathbf{P}\{\mathbf{x}_i|t\}\mathbf{P}\{t\} - \sum_t \mathbf{P}\{y|t\}\mathbf{P}\{t\}\mathbf{w}_i\right) \\ &= \eta\left(\sum_t \mathbf{P}\{\mathbf{x}_i, y|t\}\mathbf{P}\{t\} - \mathbf{w}_i \sum_t \mathbf{P}\{y|t\}\mathbf{P}\{t\}\right) \end{aligned} \tag{11}$$

Setting this update equal to 0 to find the equilibrium yields

$$\mathbf{w}_i = \frac{\mathbf{P}\{\mathbf{x}_i, y\}}{\mathbf{P}\{y\}} = \mathbf{P}\{\mathbf{x}_i|y\} \tag{12}$$

by the definition of conditional probability. For the case where $y = 1$ all the time, we recover our previous analysis: $\mathbf{P}\{\mathbf{x}_i|y\} = \mathbf{P}\{\mathbf{x}_i\}$, since y never changes. Since we are interpreting $\mathbf{x}_i$ as a probability, $\mathbf{w}_i = \mathbf{E}_t\left[\mathbf{x}_i\right]$.

2.2.3 Adding *k*-Winner-Take-All Inhibition

Thus far we have only considered linear neurons with a single output unit. However, we would like to use Hebbian learning to build semantic vectors with many features to represent words in language. The solution to this problem is to use competitive learning, i.e. interneuronal inhibition. Our method of choice is to use *k*-winner-take-all inhibition as in Equation 3. We can think of *k*-winner-take-all as finding the mean vectors of various subsets of the data. Essentially, we are diving the data into k clusters, each summarized by a single vector. We can see this result by applying the average velocity approximation to $\mathcal{W}_\mathbf{j}$, where j corresponds to a unit in the hidden layer $\mathcal{H}$ (thus, $j \in [|\mathcal{H}])$. The activation of this unit is $\mathbf{y}_j$. Here, $\mathcal{W}_\mathbf{j}$ is a vector since we no longer have one output. Following [Seung2015],

$$\triangle \mathcal{W}_\mathbf{j} \approx \eta \left(\mathbf{E}_t \left[\mathbf{y}_j \mathbf{x} \right] - \mathbf{E}_t \left[\mathbf{y}_j \right] \mathcal{W}_\mathbf{j} \right) \tag{13}$$

Again setting the update to $\mathbf{0}$ to find the steady state, we get

$$\hat{\mathcal{W}}_\mathbf{j} = \frac{\mathbf{E}_t \left[\mathbf{y}_j \mathbf{x} \right]}{\mathbf{E}_t \left[\mathbf{y}_j \right]} \tag{14}$$

If we simplify Equation 3 so that $\mathbf{y}_j$ is either 0 or 1, then we get that $\hat{\mathcal{W}}_\mathbf{j}$ is the normalized average for the subset of vectors for which $\mathbf{y}_j$ is non-zero. If we let $\mathbf{y}_j \in [0,1]$, then we have a weighted average of sorts.

With this competitive learning inhibition, multiple hidden units give additional information about the inputs. We can take $\mathcal{W}_\mathbf{i}$ as the semantic vector for each input unit i.

2.3. Interpreting Hebbian Semantic Vectors

We can use the analysis in the previous section to better understand what it is the word vectors really are. Keep in mind that our inputs are sequential blocks of words of size $2r + 1$, where r is a window radius. In the simplest case, we express these blocks in vector form by letting $\mathbf{x}_i = 1$ if word i in $\mathcal{V}$ is present and 0 otherwise (we will later refer to this setup as the uniform case, because we do not distinguish between locations in the window).

Now, Equation 14 tells us that each unit in $\mathcal{H}$ is associated with an average vector over a subset of words that are co-activated simultaneously fairly often. Let $\mathcal{S}_j$ be the word subset associated with hidden unit j. Denote $\mathbf{v}_i$ as the semantic vector of word $\mathcal{V}(i)$. Then $\mathbf{v}_i$'s largest feature values are located at indices j such that $\mathcal{V}(i) \in \mathcal{S}_j$.

We can also get intuition about the extent to which we are restricting the representation. Since there are k possible units active at a given time, there are $\leq \binom{|\mathcal{H}|}{k}$ different possible subsets of units that can be learned. The parameter k represents how many units we use to represent a given subset of words. However, we are limited by the need to allow for "dead cells" which do not learn in order to find an optimal representation, which is why $\binom{|\mathcal{H}|}{k}$ is only an upper bound [O'Reilly2000]. There are $\binom{|\mathcal{V}|}{2r+1}$ possible windows of length $2r + 1$ (since in the uniform case, word order does not matter and these windows are essentially bags of words). If $|\mathcal{V}|$ is much larger than $|\mathcal{H}|$, then we are imposing a restriction on the relatedness of various words. There may be some optimal value of $|\mathcal{H}|$ such that the number of subsets of words is very close to the actual value of meaningful subsets of size k.

2.4. Beyond Bag-of-Words

The uniform Hebbian approach is invariant to sentence structure and grammar. In fact, in the implementation given in [O'Reilly2012], language structure is only scrutinized at a paragraph level in a bag-of-words style. We would like to see if we can modify the basic model to learn semantic vectors that take into account the structure of sentences.

2.4.1 Context Window Distribution

The modification we make to the previously given model lies in the representation of the input vectors $\mathbf{x}$. Instead of letting $\mathbf{x}_i = 1$ if word $\mathcal{V}(i)$ is present in the window and 0 otherwise, we attach a distribution $\mathcal{D}$ so that $\mathbf{x}_i = \mathcal{D}(f(i))$ and 0 otherwise, where $f(i)$ is a map from the input indices which are activated to the ordered set $[2r+1]$ (essentially, we want the distribution to be applied in order over the sliding context window).

The idea behind this setup is that word order should matter. If our context window size is small enough, we will essentially be scanning over sentences or subsentence contexts. As a simple example, perhaps the closer a word is to the middle of the window, the more important it is in relation to the other words in the window. Here, a reasonable model would be choosing $\mathcal{D}$ to be Gaussian $\mathcal{N}(r+1, \sigma^2)$ centered at the middle index of the window (a middle is guaranteed since $2r+1$ is odd). A more novel distribution might be bimodal, peaking at the edges of the context window with a minimum at the center. The influence behind this choice of distribution would be the hypothesis that words spaced $2r+1$ words apart are particularly related with each other, and only mildly related to words inbetween this distance. We could consider this to be a skim-reading model of language: A skim-reader who glances at every $2r+1^{st}$ word to get the gist of a document would tie representations of these words strongly together.

In this paper, we analyze three different context window distributions:

1. uniform, where $\mathcal{D}(i) = 1$ for all $i \in [2r+1]$.

2. unimodal, where $\mathcal{D} = \mathcal{N}(r+1, \sigma^2)$, where $r+1$ is the center of the window.

3. bimodal, where $\mathcal{D}(i) = \begin{cases} \mathcal{N}(0,\sigma^2)(i) & \text{if } i \le r+1 \\ \mathcal{N}(2r+1,\sigma^2)(i) & \text{if } i \ge r+1. \end{cases}$. We take care to choose σ so that th distributions agree at the center, $r+1$.

   We can also parametrize our models by r - this translates to varying context window size.

3. Cross-Lingual Evaluation Metrics for Semantic Vectors

We introduce a few novel metrics for evaluating semantic vector spaces. Generally, the mantra to remember is "meaning is invariant across languages". This phrase guides our intuition that a set of word vectors trained on a corpus from language $\mathcal{L}_1$ should behave very similarly to a corpus trained on a corpus from a language $\mathcal{L}_2$. We will use the quantitative metrics to compare semantic vector models parametrized by tuples $(\mathcal{L}_i, \mathcal{D}, r)$. Basically, each attempts to determine a "closeness" between two semantic vector models. We also give a visual metric so that we can gain intuition about the overall shape of the distribution of the semantic vectors.

3.1. Quantitative Metrics

3.1.1 Language Similarity Distance

Let $v_i^{\mathcal{L}_1}$ be the word vector in language $\mathcal{L}_2$ for a given word $\mathcal{V}_{\mathcal{L}_1}(i)$, and $v_j^{\mathcal{L}_2}$ be the word vector for the translation of the word $\mathcal{T}\left(\mathcal{V}_{\mathcal{L}_1}(i)\right) = \mathcal{V}_{\mathcal{L}_2}(j)$. Note that we assume $\mathcal{T} : \mathcal{V}_{\mathcal{L}_1} \to \mathcal{V}_{\mathcal{L}_2}$ is bijective, and therefore that $|\mathcal{V}_{\mathcal{L}_1}| = |\mathcal{V}_{\mathcal{L}_2}|$. Then, we want to compute how well the vectors for $\mathcal{V}_{\mathcal{L}_1}(i)$ and $\mathcal{V}_{\mathcal{L}_2}(j)$ represent the language-invariant meaning behind the word. Note that we first normalize all vectors. Let word $i \in \mathcal{V}_{\mathcal{L}_1}$.

$$\text{LSD}(i, \mathcal{V}_{\mathcal{L}_1}, \mathcal{T}) = \sqrt{\sum_{j=1}^{|\mathcal{V}_{\mathcal{L}_1}|} \left((v_i^{\mathcal{L}_1} \cdot v_j^{\mathcal{L}_1}) - (v_i^{\mathcal{L}_2} \cdot v_j^{\mathcal{L}_2}) \right)^2} \tag{15}$$

The justification behind this is as follows: First we find the cosine distances between the i^{th} and j^{th} word vectors in both languages. Then we take the difference in the cosine distances - this value roughly tells us the difference in rotatation it would take to get from one of the vectors to the other. We ignore the direction we have to rotate, so we take a square sum. We conclude that the word vector representation is good across languages for a given word i if $\text{LSD}(i, \cdots)$ is close to zero, and thus that the word vectors do a good job of capturing meaning of i with respect to the other words.

Using this metric, we can define metrics for a pair of word vector sets in terms of the total language similarity distance (TLSD) and average language similarity distance (ALSD):

$$\begin{aligned} \text{TLSD}\left(\mathcal{V}_{\mathcal{L}_1}, \mathcal{T} : \mathcal{V}_{\mathcal{L}_1} \to \mathcal{V}_{\mathcal{L}_2}\right) &= \sum_{i}^{|\mathcal{V}_{\mathcal{L}_1}|} \text{LSD}\left(i, \mathcal{V}_{\mathcal{L}_1}, \mathcal{T}\right) \\ \text{ALSD}\left(\mathcal{V}_{\mathcal{L}_1}, \mathcal{T} : \mathcal{V}_{\mathcal{L}_1} \to \mathcal{V}_{\mathcal{L}_2}\right) &= \frac{1}{|\mathcal{V}_{\mathcal{L}_1}|} \text{TLSD}\left(\mathcal{V}_{\mathcal{L}_1}, \mathcal{T}\right) \end{aligned} \tag{16}$$

Again, the smaller these metrics are, the more invariant the word vectors are to languages $\mathcal{L}_1, \mathcal{L}_2$.

3.1.2 Procrustes' Transformation

If the languages $\mathcal{L}_1, \mathcal{L}_2$ are similar enough in origin, we may expect the word vectors that result from them to behave similarly already. If they are truly nice, we may expect that we can define a linear transformation (scaling, rotation, translation) from one word vector set to the other. The Procrustes' transformation gives the optimal linear transformation $\mathcal{P}$ from matrix $\mathcal{M}_1$ to $\mathcal{M}_2$ in the sense of the Frobenius distance $\left(\text{defined as } \|\mathcal{M}\|_{\mathcal{F}} = \sqrt{\textbf{tr}\left(\mathcal{M}\mathcal{M}^{\top}\right)}\right)$. That is, $\mathcal{P} : \mathcal{M}_1 \to \mathcal{M}_2$ minimizes $\|\mathcal{P}\left(\mathcal{M}_1\right) - \mathcal{M}_2\|_{\mathcal{F}}$.

Here, we represent the semantic vector sets as matrices $\mathcal{X}_{\mathcal{L}_1}, \mathcal{X}_{\mathcal{L}_2} \in \mathbb{R}^{|\mathcal{H}| \times |\mathcal{L}_1|}$ for each language since $|\mathcal{L}_1| = |\mathcal{L}_2|$. We can define three different metrics based on Frobenius distance.

1. We can evaluate the starting distance $\|\mathcal{X}_{\mathcal{L}_1} - \mathcal{X}_{\mathcal{L}_2}\|_{\mathcal{F}}$.
2. We can evaluate the Procrustes' distance after a Procrustes' transform
$$\|\mathcal{P}\left(\mathcal{X}_{\mathcal{L}_1}\right) - \mathcal{X}_{\mathcal{L}_2}\|_{\mathcal{F}} \tag{17}$$
3. We can evaluate the Procrustes' ratio
$$\frac{\|\mathcal{X}_{\mathcal{L}_1} - \mathcal{X}_{\mathcal{L}_2}\|_{\mathcal{F}}}{\|\mathcal{P}\left(\mathcal{X}_{\mathcal{L}_1}\right) - \mathcal{X}_{\mathcal{L}_2}\|_{\mathcal{F}}} \tag{18}$$
which would be large for word vector sets for which a very close Procrustes' transform existed. A large score is better for the third metric since a larger decrease ratio in Frobenius distance implies that the Procrustes' transform found more linear structure in the map, which we define to represent a better semantic vector set.

3.1.3 *k*-Nearest Neighbors

It is also useful to develop a tool to find the closest words to a given word in both languages to see how different the returned words are. The larger the overlap between the k-closest word sets in $\mathcal{L}_1$ and $\mathcal{L}_2$, the better the word vectors.

We compare across languages by seeing how many of the words are corresponding translation pairs, thus enabling the calculation of recall for several vector pairs to evaluate performance. Recall that recall is defined as $\frac{|\text{Desired Retrieved}|}{|\text{Desired Total}|}$. We will call this metric k-recall in order to embed the parameter k in their definition. Defined explicitly, for each word in $\mathcal{L}_1$, we will produce the k closest words in the sense of Euclidean distance in the semantic vector space for $\mathcal{L}_1$. For each translation we do the same thing. Then for each translation pair, we have a set of closest words. We can translate the $\mathcal{L}_2$ words to their $\mathcal{L}_1$ forms and evaluate k-recall (which, since there are k points for both French and English, is the same as k-precision).

We can also use k-Nearest Neighbors as a more visual tool to assess how good the word vectors are qualitatively. For instance, if we know two characters are very strongly related in a story, we would expect the angle between the words to be small (i.e. the cosine is large). We find the closest vectors to a few other interesting words and see if they make sense.

3.2. *t*-SNE Projection

We would also like to visualize our word vectors in a more qualitative way to see if there is interesting structure.

t-SNE projection is a method of projecting words in higher dimensional space down to $\mathbb{R}^2$, so that they can be plotted and visualized [van der Maaten2008]. The basic idea is that we define a distribution over pairwise distances between vectors in the high dimensional space, as well as a distribution over pairwise distances for vectors in $\mathbb{R}^2$. The object of t-SNE is to minimize the Kullback-Leibler divergence between these two distributions, where the KL-divergence is an asymmetric pseudometric that satisfies $\mathcal{D}_{KL}(\mathcal{P}\|\mathcal{Q}) \geq 0$ for any distributions $\mathcal{P}, \mathcal{Q}$. It is essentially the expectation of the difference in log probabilties and is defined directly as

$$\mathcal{D}_{KL}(\mathcal{P}\|\mathcal{Q}) = \sum_i \mathcal{P}(i) \ln \left(\frac{\mathcal{P}(i)}{\mathcal{Q}(i)} \right) \tag{19}$$

Another interesting interpretation of the KL-divergence is as the Bregman distance over the simplex, which is relevant in more technical machine learning theorems.

t-SNE projection has become popular as a means of visualizing high-dimensional spaces over the past several years, and most of the newer papers involving word vectors cited in the References use this approach to make plots.

4. Implementation Details

4.1. Dataset

The process of narrowing down parallel corpora took some time. First we decided upon the languages $\mathcal{L}_1, \mathcal{L}_2$. In this paper, $\mathcal{L}_1 =$ English and $\mathcal{L}_2 =$ French. We chose English and French because they have similar roots and most importantly, the author is able to read both languages.

Originally, the parallel corpus was to come from the publicly available EuroParl corpus (EuroParl). However, the EuroParl corpus was very large and the vocabulary was also very large for both English and French, which imposed an issue on time constraints and methods of choosing an appropriate subset. We wanted a vocabulary that was small enough so we could include the whole vocabulary in the input layer of the neural net. Ideally, the corpus will not be too large so that we can train on the whole corpus in a reasonable amount of time. We also required that the semantic content of the windows should be very similar, which is relatively true across book translations. We then decided that a reasonable length chapter book would provide all of these properties.

We also thought it would be useful to be particularly familiar with the chosen book, so as to have an intuitive sense for word distributions in the corpus. Finally, we thought it would be interesting to have a book with some words that are specific to the book to see how they interact. A chapter book with a plot also allows us to see interesting interactions between characters. Here, words can be names - names represent a whole character, more than just the meaning of a word. Therefore, we choose *Harry Potter and the Philosopher's Stone* ([Rowling1997], [Ménard1998]), one of the author's own favorite books. The English version of the text used has 81536 words , and the number of words after preprocessing is 77744. The English vocabulary size is 5982. The French version of the text used has 85472 words, and after preprocessing has 89709 words (due to expansion of some apostrophe-joined words). The French vocabulary size is 8152.

In order to transform the books into an analyzable format, we converted owned PDFs of both versions of the book and used a freely available online tool (convert-to-txt) to convert it to a text file.

4.2. Preprocessing the Corpora

As a result of the method of obtaining text file versions of the two books, language-specific cleanup was necessary before performing any training or analysis. First of all, some typos were made in the English version of the text due to a few failures of the OCR PDF-to-TXT technology. Specifically, the largest problem was the usage of non-alphanumeric characters to replace alphanumeric characters. For example, several instances of the letter combination "fi" were replaced by another (single) character that looked similar to the combination. Similar issues were presented in the French corpus. In order to ensure that words did not have separate spelling representations when they should not, we used an English and a French dictionary (via the Python Enchant module) to check if each token that we found was a word. To ensure tokens were words, we had to strip periods, commas, colons, semicolons, questionmarks, parentheses, brackets, and slashes from both texts. We also lowercased all text. Specific to the English text were quotation marks (i.e. ""), and specific to the French text were French quotation marks (i.e. «»). More care was required for single quotes (i.e. ′′) in both texts, since some words include a single quote inside of them (for instance, "aujourd′hui" in French). Of course, contractions also contain these single quotes. In French, it was also important to handle the dash "-" correctly: Dashes are used to denote speech, and are also used in verb conjugations in questions. We also took care to ensure that these modifications did not result in words getting pulled together. Considerable hand-checking was performed whenever a word was not identified by the dictionary, usually requiring specialized Python scripts to check specific patterns in text.

Because Harry Potter is a fantasy series, there are also several words that are not in normal English and French dictionaries. The Enchant module has a function to add your own dictionary to the module, and Harry Potter-specific words were added by hand throughout the process. Some multi-word objects were encoded as single words (i.e. "You Know Who", a common phrase referring to Voldemort, the villain of the book). For French, translation of Harry Potter-specific words had to be performed carefully since the author had not read the whole French book before. To the end of ensuring translations were accurate, a context-checker was implemented to search the book for all appearances of a specific word and then provide the surrouding context so that the translation was obvious. In cases of serious doubt, the word was looked up on the internet.

After this process was carried out, a string containing the entire novel was split up by spaces and stored in an array that fit entirely in RAM. For easy access later on, this array was saved into a file that can be re-loaded easily.

One possible issue with our approach might be that we did not preprocess with consideration to lemmatization. Instead, we just preprocessed at the specific word level, which means that different tenses of verbs are considered different words and that plural and singular nouns are considered different as well. We did not consider this too much of a problem since effectively, we will be duplicating word vectors if (for instance) the plural and singular of a noun are used in the same settings. In some settings, singular and plural nouns may be used in different ways: "the Weasleys" refers to the family as a group, while "Weasley" may refer to only one member of the family. In these cases, we desire different word vectors anyways.

Another possible issue is that since we remove periods, when running the sliding window across the corpora, we ignore sentence boundaries, meaning that some windows might have semantic content that is not as self-contained. However, there is also an argument that we should consider windows across sentences. Even if there is not as clear a grammatical relationship between such words, we are more interested in semantics and adjacent sentences may have similar content. A more profound problem is across paragraphs, or potentially chapters. The approach we took was the simplest in the interest of time. Further work could experiment with more specific boundaries on window context.

4.3. Restricting the Vocabularies

Originally, the English vocabulary size after preprocessing was 5982 and the French vocabulary size was 8152. We wanted to analyze a subset of the vectors produced for these words, and ensure they were interesting. After poking through the word lists ordered by frequency in corpus, we decided to throw out the most common 100 words except for 14 of them, since most of the words were articles, conjunctions, and pronouns. We then kept the 1900 words following for a total of 1914 words.

Then we had to create a bijective translation dictionary in order to evaluate the metrics defined in Section 3. To perform this task automatically, we programmatically accessed Google Translate and checked if the English-French translation result was in the set of valid French Harry Potter vocabulary words (standard French + the handcrafted Harry Potter-specific dictionary). We made a list of words that did not check out, and then checked those by hand. From this process, we ended up with 1368 English words and 1187 French words. We noted that the set of French words is smaller than the English set, since sometimes there are two English words with the same French translation: For instance, in plural cases ("Gryffindor" and "Gryffindors" both become "Gryffondor" in French). Another example is synonymity: "Warm" and "hot" both map to "chaud". To ensure we can compare word-to-word, we removed these duplicates so that there is a one-to-one mapping between French and English words. The final vocabulary size was therefore $|\mathcal{V}| = 1187$. Therefore, the number of word pairs in each vocabulary is $\binom{1187}{2} = 703891$ different comparisons to make.

We restricted the vocabulary size so that computations were feasible (we may have had to hand-translate a lot more otherwise). Furthermore, smaller frequency words probably have worse word vectors. Since we have a relatively small corpus, we should focus more on the words which have good representation. Also, if we had analyzed more words, picking out individual words to analyze may have been more difficult, as the number of word pairs increases roughly as the square of the vocabulary size.

4.4. Training the Networks

We trained 18 separate Hebbian networks of the type described in Section 2 to compare. We describe the set of parameters that maps to each network. Here, "en" refers to English and "fr" refers to French. Recall that the tuple refers to language, distribution, and window radius r (windows are size $2r+1$). The set of the networks $\mathcal{N}$ is therefore given by

$$\mathcal{N} = \{\text{"en"}, \text{"fr"}\} \times \{\text{uniform}, \text{unimodal}, \text{bimodal}\} \times \{2, 3, 4\} \tag{20}$$

The weight matrix $\mathcal{W}$ was uniformly randomly initialized with values in the range $[0, 1]$. We chose $|\mathcal{H}| = 100$ and $k = 10$ for k-winner-take-all inhibition. Thus the semantic vectors live in $\mathbb{R}^{100}$. We chose $\eta = 0.1$ for the learning rate. We enforced a shared vocabulary size across corpuses of $|\mathcal{V}| = 1187$ words, and had a bijective translation dictionary for the vocabularies. The number of connections in the network was therefore $|\mathcal{W}| = |\mathcal{H}| * |\mathcal{V}| = 118700$, which is comparatively small to more recent networks in the literature. We trained each network on only one run across the corpus in the interest of time. It took roughly 2 hours for a single network to train on the English text, and closer to 3 hours for a single network to train on the French version of the text. In order to train networks efficiently, we used three computers to simultaneously train networks. It took a total of around 24 hours to complete all training (sans debugging).

5. Results and Analysis

5.1. Performance of Semantic Vector Space Language Pairs

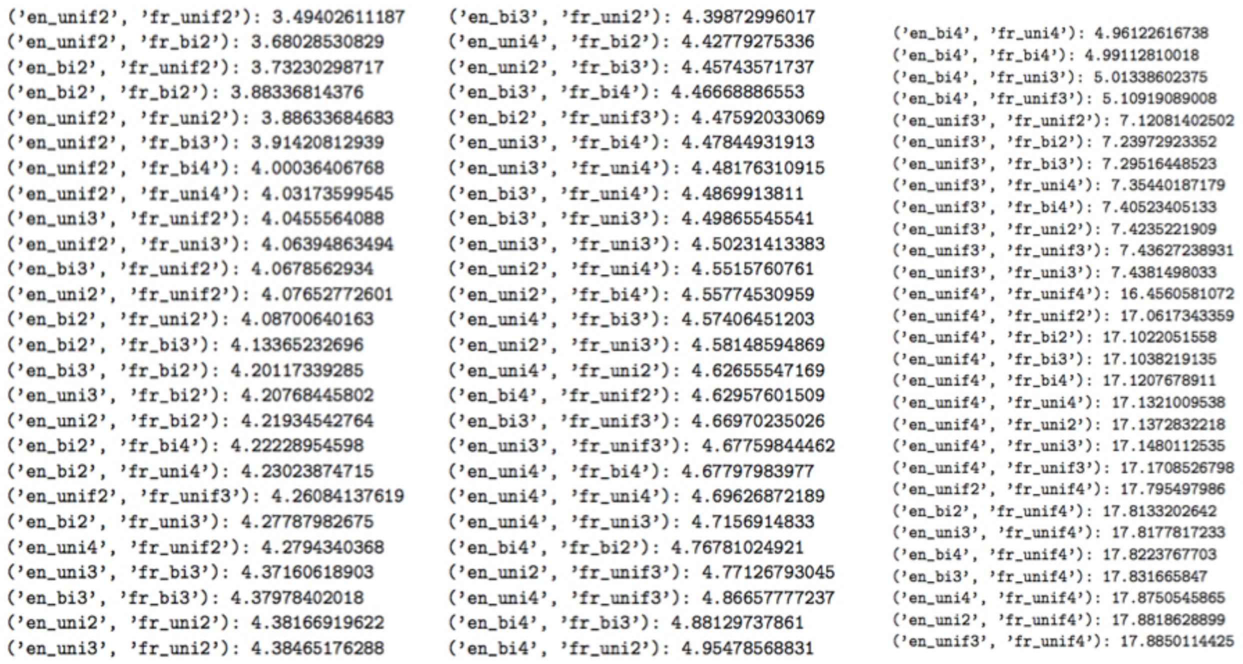
5.1.1 Language Similarity Distance

We use the Average Language Similarity Distance (ALSD) (see Equation 16) score to sort the language pairs of semantic vector sets. Each of the 81 pairs and its score are presented in Figure 2. We also plot the distribution of the scores in Figure 3. Recall that the smaller scores imply more similarity across semantic vector sets. Each comparison takes an average over roughly 1187 LSD comparisons (each of which is calculated as the square root of a sum of 1187 squared distances). So $1187^2 = 1408969 \approx 1.4 \times 10^6$ cosine distance comparisons are being made to calculate the ALSD score for one pair of parallel semantic vector sets.

First of all, we notice a large dropoff in ALSD quality at semantic vector set language pair 65. Interestingly, this dropoff contains all vector space models which use (uniform, $r = 4$) for both the English and French models. Therefore, the (uniform, $r = 4$) setting is terrible for this metric: Perhaps the content window is too large, suggesting that smaller content windows are sufficient. Indeed, the English (uniform, $r = 3$) distribution also does not perform very well. $r = 2$ is drastically better than the other uniform distributions for English. The trend of low r performing better seems to hold across distributions and languages as well: The top 10 model pairs have $\frac{15}{20}$ models with $r = 2$, $\frac{3}{20}$ models with $r = 3$, and only $\frac{2}{20}$ models with $r = 4$. Perhaps not too surprisingly the best model pair is the same model, simply compared across language. It is also arguably the simplest model with the fewest assumptions: uniform weights and the smallest r possible. The (bimodal, $r = 2$) set of word vectors seems to be roughly the second best in terms of ranking.

5.1.2 Procrustes' Distance and Procrustes' Ratio

For every semantic vector set pair, we give the Procrustes' distance scores in Figure 4, where the smaller the score, the better the pair. We give the Procrustes' ratio scores for every pair in Figure 5: Here, larger scores imply a better pair. We also provide plots of the values for the Procrustes' Distance and Procrustes' Ratio scores in Figure 6 so that the distribution of values is clear. The first aspect of the Procrustes metrics that we notice is the highly similar shape of the Procrustes' distance plot to the ALSD plot – they have practically the same shape, just slightly translated by maybe 5 semantic vector set pairs. Particularly, note the slight bump before the values shoot up. This phenomenon would lead us to expect that the pairs that do well in ALSD also do well with the Procrustes' distance metric. However, briefly scanning Figure 2 and Figure 4 demonstrates that (English, uniform, $r = 2$) performs badly with Procrustes' distance and very well with ALSD (we count several pairs for which this vector set is at the end of the list of pairs for Procrustes' distance, and several pairs for which this vector set is at the top of the list of pairs for ALSD). However, it is clear that several of the top performing ALSD vector set pairs are top performing on Procrustes' distance as well; for instance, (French, bimodal, $r = 2, 3$). We also note that for



```
(’en_unif2’, ’fr_unif2’): 3.49402611187
(’en_unif2’, ’fr_bi2’): 3.68028530829
(’en_bi2’, ’fr_unif2’): 3.73230298717
(’en_bi2’, ’fr_bi2’): 3.88336814376
(’en_unif2’, ’fr_uni2’): 3.88633684683
(’en_unif2’, ’fr_bi3’): 3.91420812939
(’en_unif2’, ’fr_bi4’): 4.00036406768
(’en_unif2’, ’fr_uni4’): 4.03173599545
(’en_uni3’, ’fr_unif2’): 4.0455564088
(’en_unif2’, ’fr_uni3’): 4.06394863494
(’en_bi3’, ’fr_unif2’): 4.0678562934
(’en_uni2’, ’fr_unif2’): 4.07652772601
(’en_bi2’, ’fr_uni2’): 4.08700640163
(’en_bi2’, ’fr_bi3’): 4.13365232696
(’en_bi3’, ’fr_bi2’): 4.20117339285
(’en_uni3’, ’fr_bi2’): 4.20768445802
(’en_uni2’, ’fr_bi2’): 4.21934542764
(’en_bi2’, ’fr_bi4’): 4.22228954598
(’en_bi2’, ’fr_uni4’): 4.23023874715
(’en_unif2’, ’fr_unif3’): 4.26084137619
(’en_bi2’, ’fr_uni3’): 4.27787982675
(’en_uni4’, ’fr_unif2’): 4.2794340368
(’en_uni3’, ’fr_bi3’): 4.37160618903
(’en_bi3’, ’fr_bi3’): 4.37978402018
(’en_uni2’, ’fr_uni2’): 4.38166919622
(’en_uni3’, ’fr_uni2’): 4.38465176288
```

```
(’en_bi3’, ’fr_uni2’): 4.39872996017
(’en_uni4’, ’fr_bi2’): 4.42779275336
(’en_uni2’, ’fr_bi3’): 4.45743571737
(’en_bi3’, ’fr_bi4’): 4.46668886553
(’en_bi2’, ’fr_unif3’): 4.47592033069
(’en_uni3’, ’fr_bi4’): 4.47844931913
(’en_uni3’, ’fr_uni4’): 4.48176310915
(’en_bi3’, ’fr_uni4’): 4.4869913811
(’en_bi3’, ’fr_uni3’): 4.49865545541
(’en_uni3’, ’fr_uni3’): 4.50231413383
(’en_uni2’, ’fr_uni4’): 4.5515760761
(’en_uni2’, ’fr_bi4’): 4.55774530959
(’en_uni4’, ’fr_bi3’): 4.57406451203
(’en_uni2’, ’fr_uni3’): 4.58148594869
(’en_uni4’, ’fr_uni2’): 4.62655547169
(’en_bi4’, ’fr_unif2’): 4.62957601509
(’en_bi3’, ’fr_unif3’): 4.66970235026
(’en_uni3’, ’fr_unif3’): 4.67759844462
(’en_uni4’, ’fr_bi4’): 4.67797983977
(’en_uni4’, ’fr_uni4’): 4.69626872189
(’en_uni4’, ’fr_uni3’): 4.7156914833
(’en_bi4’, ’fr_bi2’): 4.76781024921
(’en_uni2’, ’fr_unif3’): 4.77126793045
(’en_uni4’, ’fr_unif3’): 4.86657777237
(’en_bi4’, ’fr_bi3’): 4.88129737861
(’en_bi4’, ’fr_uni2’): 4.95478568831
```

```
(’en_bi4’, ’fr_uni4’): 4.96122616738
(’en_bi4’, ’fr_bi4’): 4.99112810018
(’en_bi4’, ’fr_uni3’): 5.01338602375
(’en_bi4’, ’fr_unif3’): 5.10919089008
(’en_unif3’, ’fr_unif2’): 7.12081402502
(’en_unif3’, ’fr_bi2’): 7.23972923352
(’en_unif3’, ’fr_bi3’): 7.29516448523
(’en_unif3’, ’fr_uni4’): 7.35440187179
(’en_unif3’, ’fr_bi4’): 7.40523405133
(’en_unif3’, ’fr_uni2’): 7.4235221909
(’en_unif3’, ’fr_unif3’): 7.43627238931
(’en_unif3’, ’fr_uni3’): 7.4381498033
(’en_unif4’, ’fr_unif4’): 16.4560581072
(’en_unif4’, ’fr_unif2’): 17.0617343359
(’en_unif4’, ’fr_bi2’): 17.1022051558
(’en_unif4’, ’fr_bi3’): 17.1038219135
(’en_unif4’, ’fr_bi4’): 17.1207678911
(’en_unif4’, ’fr_uni4’): 17.1321009538
(’en_unif4’, ’fr_uni2’): 17.1372832218
(’en_unif4’, ’fr_uni3’): 17.1480112535
(’en_unif4’, ’fr_unif3’): 17.1708526798
(’en_unif2’, ’fr_unif4’): 17.795497986
(’en_bi2’, ’fr_unif4’): 17.8133202642
(’en_uni3’, ’fr_unif4’): 17.8177817233
(’en_bi4’, ’fr_unif4’): 17.8223767703
(’en_bi3’, ’fr_unif4’): 17.831665847
(’en_uni4’, ’fr_unif4’): 17.8750545865
(’en_uni2’, ’fr_unif4’): 17.8818628899
(’en_unif3’, ’fr_unif4’): 17.8850114425
```

Figure 2: *Each of the Semantic Vector Set Pairs and Associated ALSD Score*

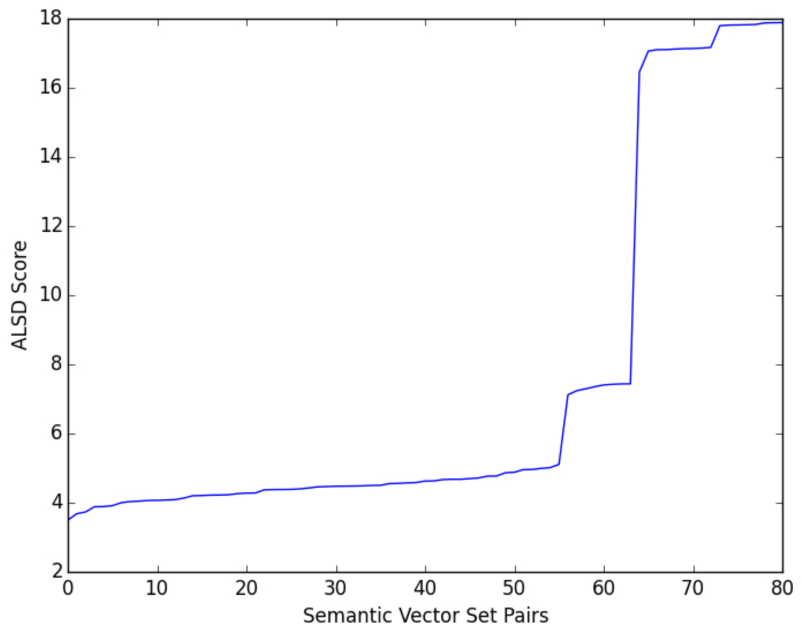
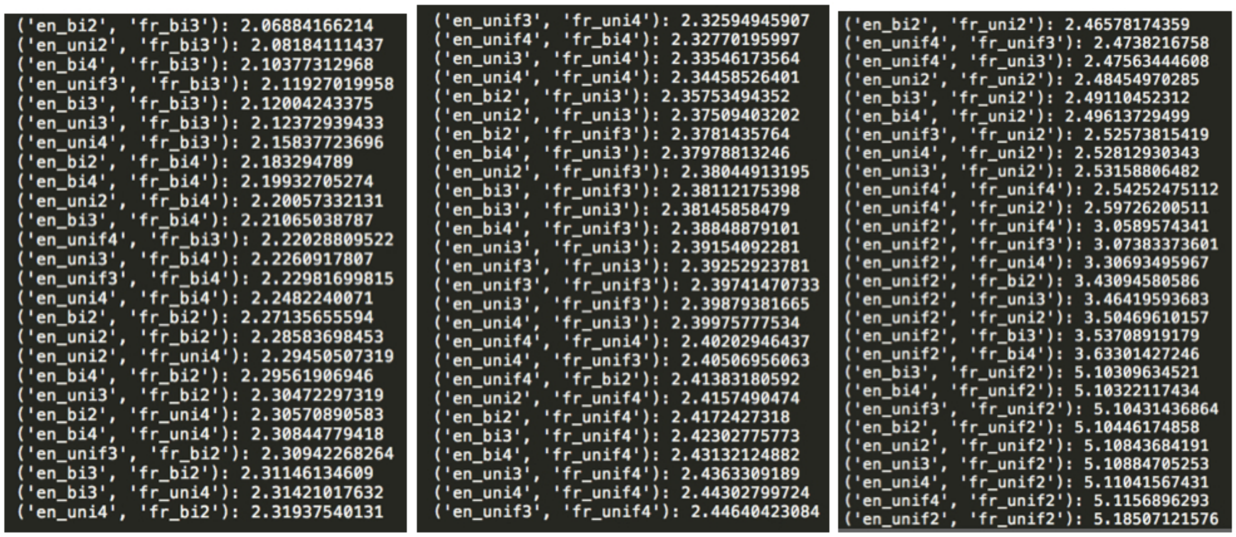


Figure 3: *Distribution of the ALSD Scores (Smallest to Largest)*

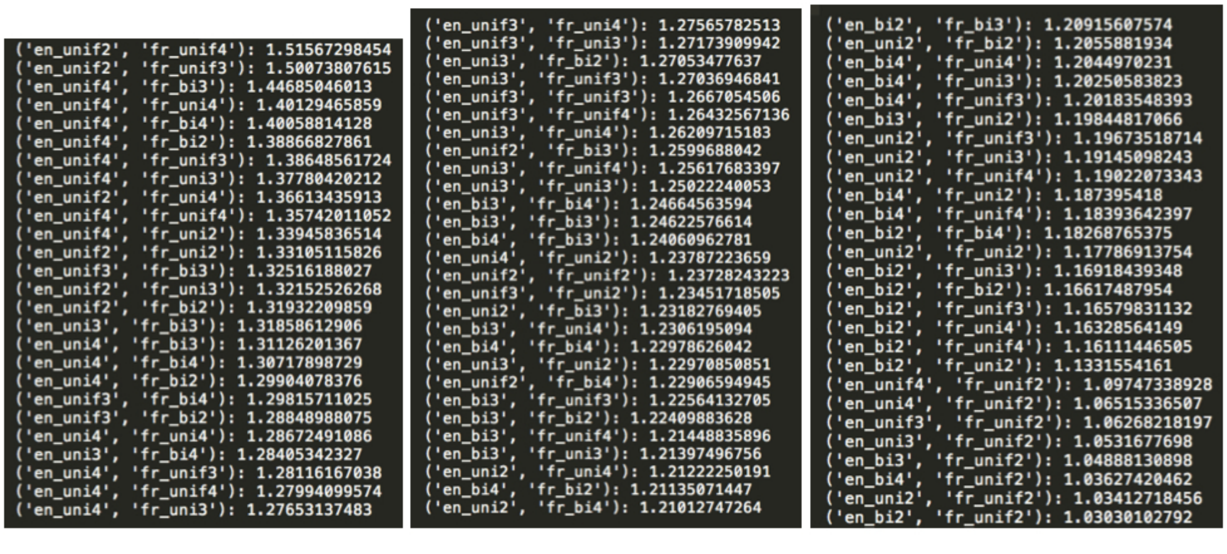


```
('en_bi2', 'fr_bi3'): 2.06884166214
('en_uni2', 'fr_bi3'): 2.08184111437
('en_bi4', 'fr_bi3'): 2.10377312968
('en_unif3', 'fr_bi3'): 2.11927019958
('en_bi3', 'fr_bi3'): 2.12004243375
('en_uni3', 'fr_bi3'): 2.12372939433
('en_uni4', 'fr_bi3'): 2.15837723696
('en_bi2', 'fr_bi4'): 2.183294789
('en_bi4', 'fr_bi4'): 2.19932705274
('en_uni2', 'fr_bi4'): 2.20057332131
('en_bi3', 'fr_bi4'): 2.21065038787
('en_unif4', 'fr_bi3'): 2.22028809522
('en_uni3', 'fr_bi4'): 2.2260917807
('en_unif3', 'fr_bi4'): 2.22981699815
('en_uni4', 'fr_bi4'): 2.2482240071
('en_bi2', 'fr_bi2'): 2.27135655594
('en_uni2', 'fr_bi2'): 2.28583698453
('en_uni2', 'fr_uni4'): 2.29450507319
('en_bi4', 'fr_bi2'): 2.29561906946
('en_uni3', 'fr_bi2'): 2.30472297319
('en_bi2', 'fr_uni4'): 2.30570890583
('en_bi4', 'fr_uni4'): 2.30844779418
('en_unif3', 'fr_bi2'): 2.30942268264
('en_bi3', 'fr_bi2'): 2.31146134609
('en_bi3', 'fr_uni4'): 2.31421017632
('en_uni4', 'fr_bi2'): 2.31937540131
```

```
('en_unif3', 'fr_uni4'): 2.32594945907
('en_unif4', 'fr_bi4'): 2.32770195997
('en_uni3', 'fr_uni4'): 2.33546173564
('en_uni4', 'fr_uni4'): 2.34458526401
('en_bi2', 'fr_uni3'): 2.35753494352
('en_uni2', 'fr_uni3'): 2.37509403202
('en_bi2', 'fr_unif3'): 2.3781435764
('en_bi4', 'fr_uni3'): 2.37978813246
('en_uni2', 'fr_unif3'): 2.38044913195
('en_bi3', 'fr_unif3'): 2.38112175398
('en_bi3', 'fr_uni3'): 2.38145858479
('en_bi4', 'fr_unif3'): 2.38848879101
('en_uni3', 'fr_uni3'): 2.39154092281
('en_unif3', 'fr_uni3'): 2.39252923781
('en_unif3', 'fr_unif3'): 2.39741470733
('en_uni3', 'fr_unif3'): 2.39879381665
('en_uni4', 'fr_uni3'): 2.39975777534
('en_unif4', 'fr_uni4'): 2.40202946437
('en_uni4', 'fr_unif3'): 2.40506956063
('en_unif4', 'fr_bi2'): 2.41383180592
('en_uni2', 'fr_unif4'): 2.4157490474
('en_bi2', 'fr_unif4'): 2.4172427318
('en_bi3', 'fr_unif4'): 2.42302775773
('en_bi4', 'fr_unif4'): 2.43132124882
('en_uni3', 'fr_unif4'): 2.4363309189
('en_uni4', 'fr_unif4'): 2.44302799724
('en_unif3', 'fr_unif4'): 2.44640423084
```

```
('en_bi2', 'fr_uni2'): 2.46578174359
('en_unif4', 'fr_unif3'): 2.4738216758
('en_unif4', 'fr_uni3'): 2.47563444608
('en_uni2', 'fr_uni2'): 2.48454970285
('en_bi3', 'fr_uni2'): 2.49110452312
('en_bi4', 'fr_uni2'): 2.49613729499
('en_unif3', 'fr_uni2'): 2.52573815419
('en_uni4', 'fr_uni2'): 2.52812930343
('en_uni3', 'fr_uni2'): 2.53158806482
('en_unif4', 'fr_unif4'): 2.54252475112
('en_unif4', 'fr_uni2'): 2.59726200511
('en_unif2', 'fr_unif4'): 3.0589574341
('en_unif2', 'fr_unif3'): 3.07383373601
('en_unif2', 'fr_uni4'): 3.30693495967
('en_unif2', 'fr_bi2'): 3.43094580586
('en_unif2', 'fr_uni3'): 3.46419593683
('en_unif2', 'fr_uni2'): 3.50469610157
('en_unif2', 'fr_bi3'): 3.53708919179
('en_unif2', 'fr_bi4'): 3.63301427246
('en_bi3', 'fr_unif2'): 5.10309634521
('en_bi4', 'fr_unif2'): 5.10322117434
('en_unif3', 'fr_unif2'): 5.10431436864
('en_bi2', 'fr_unif2'): 5.10446174858
('en_uni2', 'fr_unif2'): 5.10843684191
('en_uni3', 'fr_unif2'): 5.10884705253
('en_uni4', 'fr_unif2'): 5.11041567431
('en_unif4', 'fr_unif2'): 5.1156896293
('en_unif2', 'fr_unif2'): 5.18507121576
```

Figure 4: *Each of the Semantic Vector Set Pairs and Associated Procrustes' Distance Score*



```
('en_unif2', 'fr_unif4'): 1.51567298454
('en_unif2', 'fr_unif3'): 1.50073807615
('en_unif4', 'fr_bi3'): 1.44685046013
('en_unif4', 'fr_uni4'): 1.40129465859
('en_unif4', 'fr_bi4'): 1.40058814128
('en_unif4', 'fr_bi2'): 1.38866827861
('en_unif4', 'fr_unif3'): 1.38648561724
('en_unif4', 'fr_uni3'): 1.37780420212
('en_unif2', 'fr_uni4'): 1.36613435913
('en_unif4', 'fr_unif4'): 1.35742011052
('en_unif4', 'fr_uni2'): 1.33945836514
('en_unif2', 'fr_uni2'): 1.33105115826
('en_unif3', 'fr_bi3'): 1.32516188027
('en_unif2', 'fr_uni3'): 1.32152526268
('en_unif2', 'fr_bi2'): 1.31932209859
('en_uni3', 'fr_bi3'): 1.31858612906
('en_uni4', 'fr_bi3'): 1.31126201367
('en_uni4', 'fr_bi4'): 1.30717898729
('en_uni4', 'fr_bi2'): 1.29904078376
('en_unif3', 'fr_bi4'): 1.29815711025
('en_unif3', 'fr_bi2'): 1.28848988075
('en_uni4', 'fr_uni4'): 1.28672491086
('en_uni3', 'fr_bi4'): 1.28405342327
('en_uni4', 'fr_unif3'): 1.28116167038
('en_uni4', 'fr_unif4'): 1.27994099574
('en_uni4', 'fr_uni3'): 1.27653137483
```

```
('en_unif3', 'fr_uni4'): 1.27565782513
('en_unif3', 'fr_uni3'): 1.27173909942
('en_uni3', 'fr_bi2'): 1.27053477637
('en_uni3', 'fr_unif3'): 1.27036946841
('en_unif3', 'fr_unif3'): 1.2667054506
('en_unif3', 'fr_unif4'): 1.26432567136
('en_uni3', 'fr_uni4'): 1.26209715183
('en_unif2', 'fr_bi3'): 1.2599688042
('en_uni3', 'fr_unif4'): 1.25617683397
('en_uni3', 'fr_uni3'): 1.25022240053
('en_bi3', 'fr_bi4'): 1.24664563594
('en_bi3', 'fr_bi3'): 1.24622576614
('en_bi4', 'fr_bi3'): 1.24060962781
('en_uni4', 'fr_uni2'): 1.23787223659
('en_unif2', 'fr_unif2'): 1.23728243223
('en_unif3', 'fr_uni2'): 1.23451718505
('en_uni2', 'fr_bi3'): 1.23182769405
('en_bi3', 'fr_uni4'): 1.2306195094
('en_bi4', 'fr_bi4'): 1.22978626042
('en_uni3', 'fr_uni2'): 1.22970850851
('en_unif2', 'fr_bi4'): 1.22906594945
('en_bi3', 'fr_unif3'): 1.22564132705
('en_bi3', 'fr_bi2'): 1.22409883628
('en_bi3', 'fr_unif4'): 1.21448835896
('en_bi3', 'fr_uni3'): 1.21397496756
('en_uni2', 'fr_uni4'): 1.21222250191
('en_bi4', 'fr_bi2'): 1.21135071447
('en_uni2', 'fr_bi4'): 1.21012747264
```

```
('en_bi2', 'fr_bi3'): 1.20915607574
('en_uni2', 'fr_bi2'): 1.2055881934
('en_bi4', 'fr_uni4'): 1.2044970231
('en_bi4', 'fr_uni3'): 1.20250583823
('en_bi4', 'fr_unif3'): 1.20183548393
('en_bi3', 'fr_uni2'): 1.19844817066
('en_uni2', 'fr_unif3'): 1.19673518714
('en_uni2', 'fr_uni3'): 1.19145098243
('en_uni2', 'fr_unif4'): 1.19022073343
('en_bi4', 'fr_uni2'): 1.187395418
('en_bi4', 'fr_unif4'): 1.18393642397
('en_bi2', 'fr_bi4'): 1.18268765375
('en_uni2', 'fr_uni2'): 1.17786913754
('en_bi2', 'fr_uni3'): 1.16918439348
('en_bi2', 'fr_bi2'): 1.16617487954
('en_bi2', 'fr_unif3'): 1.16579831132
('en_bi2', 'fr_uni4'): 1.16328564149
('en_bi2', 'fr_unif4'): 1.16111446505
('en_bi2', 'fr_uni2'): 1.1331554161
('en_unif4', 'fr_unif2'): 1.09747338928
('en_uni4', 'fr_unif2'): 1.06515336507
('en_unif3', 'fr_unif2'): 1.06268218197
('en_uni3', 'fr_unif2'): 1.0531677698
('en_bi3', 'fr_unif2'): 1.04888130898
('en_bi4', 'fr_unif2'): 1.03627420462
('en_uni2', 'fr_unif2'): 1.03412718456
('en_bi2', 'fr_unif2'): 1.03030102792
```

Figure 5: *Each of the Semantic Vector Set Pairs and Associated Procrustes' Ratio Score*

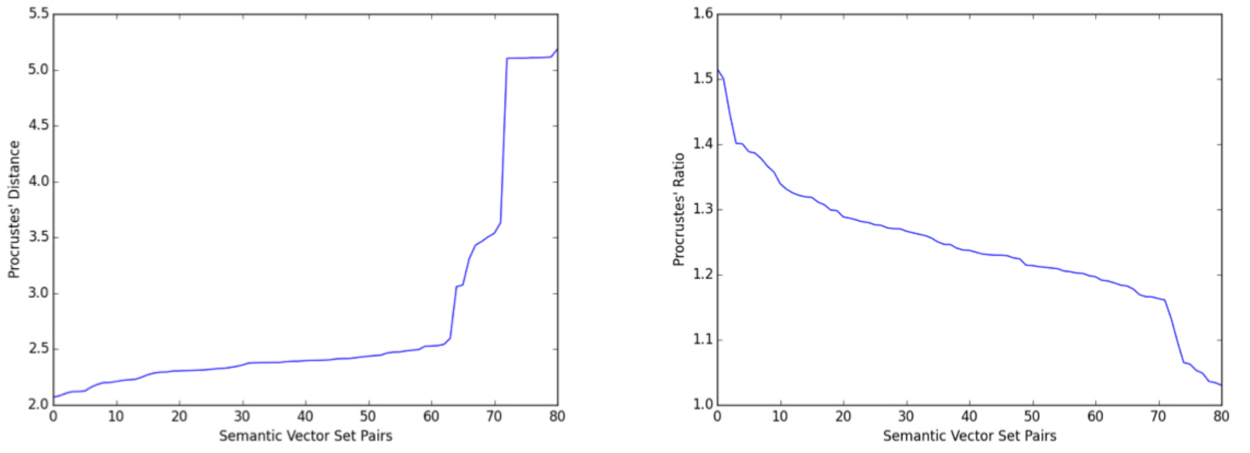


Figure 6: *Semantic Vector Pairs Plotted Best to Worst for Respective Metrics*

the first 50 or so vector set pairs, the score does not change that much and there is considerable overlap. However, the tail ends which perform quite badly do not seem to match up very much. Curiously, the pair ((English, uniform, $r = 2$); (French, uniform, $r = 2$)) has the best score for ALSD and the worst score for Procrustes' distance. Therefore, though the metrics were intended to measure a similar notion of word vector invariance across language, ALSD and Procrustes' distance appear to measure opposite traits of the word vectors! The strange part is that the distributions of values appear so similar: In fact, it is almost as though the bad pairs from ALSD have become the good pairs of Procrustes' distance, and vice versa.

In contrast to the distance scores, the pairs containing (English, uniform, $r = 2, 4$) typically do particularly well with the Procustes' ratio score. This behavior is matched in ALSD for $r = 2$, but not at all for $r = 4$ (recall that (English, uniform, $r = 4$) performed terribly on the ALSD metric).

Looking over Figure 4 and Figure 5, we see that the top semantic vector set language pairs are dominated by model comparisons that have at least one non-uniform distribution for the sliding window; particularly, the French unimodal and bimodal models tend to be considerably better than uniform.

In Figure 7, we plot both Procrustes' scores where the x-axis is ordered from 0 to 80 by increasingly worse semantic vector set pairs according to the Procrustes' ratio. In this plot, Procrutes' ratio is necessarily monotonically decreasing (since lower ratios imply worse vector set pairs) while the behavior of the Procrustes' distance need not be monotonically increasing (i.e., getting worse). Essentially, this graph demonstrates that the Procrustes' distance is an inherently different metric from the Procrustes' ratio. There appears to be no noteworthy monotonicity in the Procrustes' distance when pairs are plotted in order with respect to the Procrustes' ratio metric. While the dropoff in performance at the end (a decrease for Procrustes' ratio and an increase for Procrustes' distance) remains in the two plots - the decrease in score is nevertheless much more dramatic for the Procrustes' distance. Overall, it is fair to say that the ratio captures a different aspect of the word vector relationship.

One interesting data point is pair 49, or (English, uniform, 2) and (French, uniform 2). This pair is unique in the drastic difference in performance of the distance and ratio metrics. Recall again that this pair performed the best under the ALSD metric. While this semantic vector set pair performs averagely in terms of ratio, it is the worst in terms of Procrustes' distance, which suggests that the initial distance between the English and French vectors was particularly bad for the uniform case with $r = 2$. Since the ALSD score is good for this pair but the Frobenius distance is bad, perhaps some nonlinear transformation dilates the (French, uniform, $r = 2$) vectors so that distance is large while angles between correponding vector pairs across language are small.

Another pattern we scrutinize in more detail is the decay in quality of both the Procrustes' ratio and Procrustes' distance at the 73^{rd} semantic vector set language pair. Referring to Figure 4 and Figure 5, we see that these pairs are dominated by uniform distributions compared to distributions with non-identical parameters (i.e., either different r or different distribution). In particular, (French, uniform, $r = 2, 3, 4$) dominates the bottom end of the pairs for both Procrustes' metrics.

5.1.3 t-SNE Plots

Now we provide visual representations using t-SNE for all 18 semantic vector spaces.

Figure 8, Figure 10, Figure 12 are the English semantic vector spaces while Figure 9, Figure 11, Figure 13 are the French semantic vector spaces. Note that r increases left to right in each of these figures. Most of the representations tend to have a cluster at the center of varying size while the rest of the vectors are distributed in an increasingly sparse oval around the dense center. Two distributions violate this trend; namely, the two uniform distributions (particularly, (English, uniform, $r = 3$), (English, uniform, $r = 4$), and (French, uniform, $r = 4$)). These distributions perform very badly with the ALSD metric, but pretty well with the Procrustes' distance metric and Procrustes' ratio metric. It might be interesting future work to apply these methods on a bigger dataset to see how the picture changes – some of the uniformity is probably due to noise due to low frequencies of some words (especially given the relatively uniformly radial distribution of the data). Another bit of fruitful future work may be the categorization of the cluster

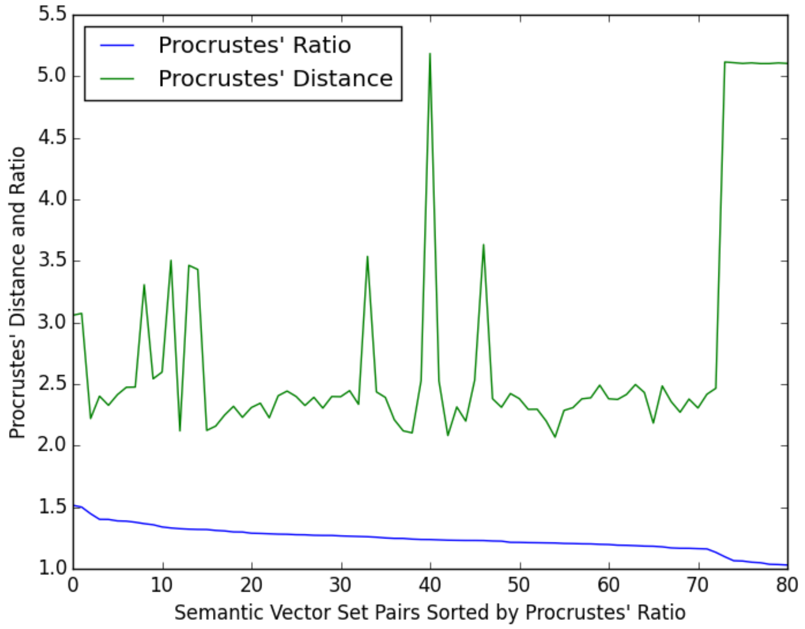


Figure 7: *Overlayed Procrustes' Distance and Procrustes' Ratio*

of low-norm vectors in the center. Based on examining the actual vector values, we suspect these smaller vectors at the center may be the vectors representing low-frequency words. Then, it would appear that the uniform distributions without this central clump may in fact have found significant representations for the low-frequency words. However, because there is not enough information in the corpus for low-frequency words, these representations might be in fact incorrect, explaining the bad ALSD scores, which leads us to posit that for the uniform distribution, higher r values require a larger corpus, which is in line with common wisdom regarding n-grams: namely, larger context windows lead to more possible combinations of words, and thus require more data to perform well. Intriguingly, this problem does not appear to arise for the unimodal and bimodal distributions. The general shape and structure is roughly retained as r increases, with the possible exception of (English, bimodal, $r = 4$), which has a significantly smaller clump at the center. Indeed, the bimodal and uniform distributions with larger r makes their appearance in the top 20 word vector set pairs multiple times, and perform similarly well for the Procrustes' distance and ratio metrics. Perhaps these distributional models have the effect of being trainable for larger contexts than a regular bag-of-words model would, due to discounting either the tail ends of the context window (unimodal) or the inner parts of the context window (bimodal).

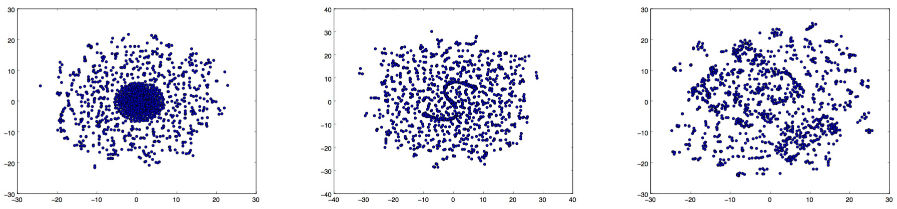


Figure 8: *t-SNE Projection for (English, uniform, $r = 2, 3, 4$)*

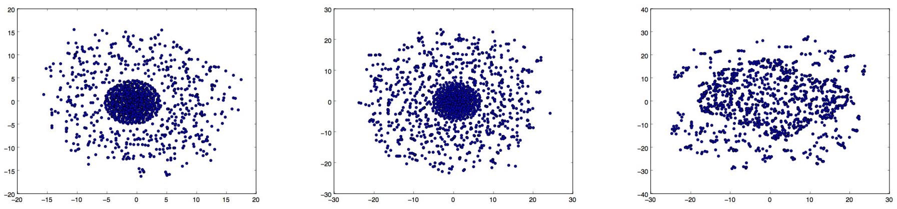


Figure 9: *t-SNE Projection for (French, uniform, $r = 2, 3, 4$)*

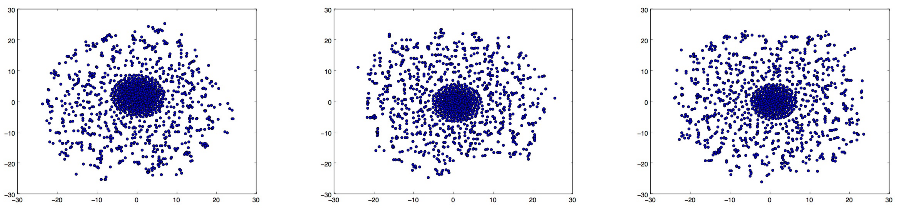


Figure 10: *t-SNE Projection for (English, unimodal, $r = 2, 3, 4$)*

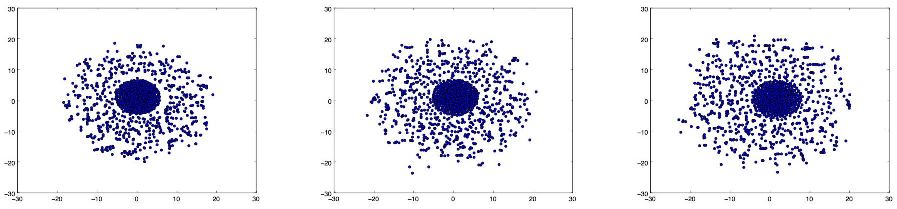


Figure 11: *t-SNE Projection for (French, unimodal, $r = 2, 3, 4$)*

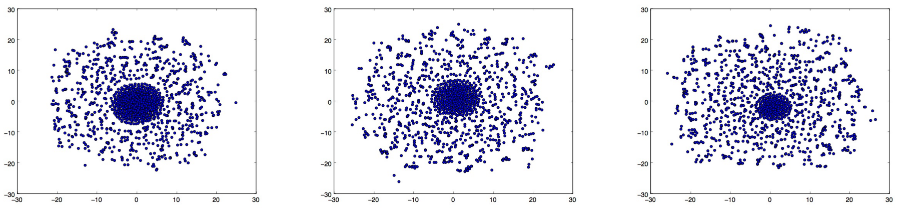


Figure 12: *t-SNE Projection for (English, bimodal, $r = 2, 3, 4$)*

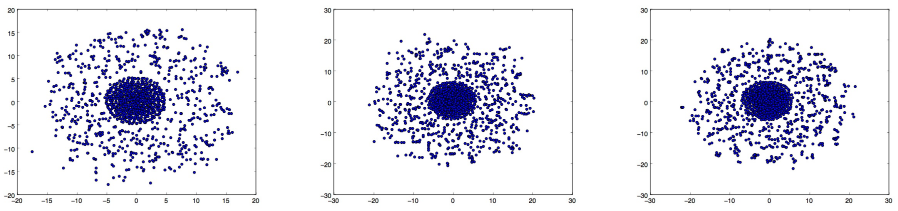


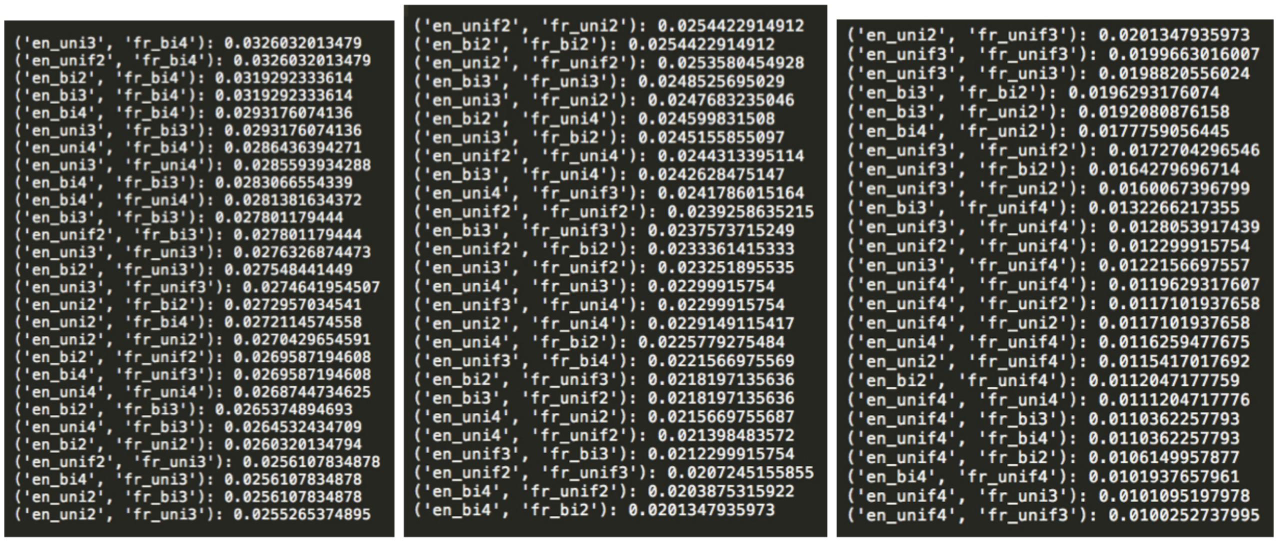
Figure 13: *t-SNE Projection for (French, bimodal, $r = 2, 3, 4$)*

5.1.4 *k*-Nearest Neighbors

For k-Nearest Neighbors, we chose $k = 10$ and calculated average k-recall over every semantic vector set pair, shown in Figure 14. Looking at the numbers, it seems that average recall was very low for all semantic vector set pairs. This facet of the models could be due to the size of k, which was chosen somewhat arbitrarily to be not too large but not too small. The small precision could also be a feature of the low-frequency word vectors which perhaps were not trained enough to perform decently. Nevertheless, this result demonstrates a clear weakness in the model.

For qualitative interest, we now consider some of the k-nearest neighbor sets for some interesting words: In Figure 15, we present the 10 closest neighbords for the following translation pairs, in both the English and French vector spaces using the (English, uniform, $r = 2$); (French, uniform, $r = 2$) word vector set language pair. Note that we present the nearest words in French semantic space translated to their English counterparts.

Some of the nearest neighbors are pretty good! The most questionable set would probably be the French "muggle" nearest words or the French "youknowwho" nearest words. "snape" has particularly good nearest neighbors for French. Recall that these nearest words are only produced for a small subset of the 2×1187 vectors associated with a given distribution and radius, and that we are only demonstrating what these nearest words look like for one semantic vector set language pair. The code is available to try out producing nearest neighbors for other words in other semantic spaces.

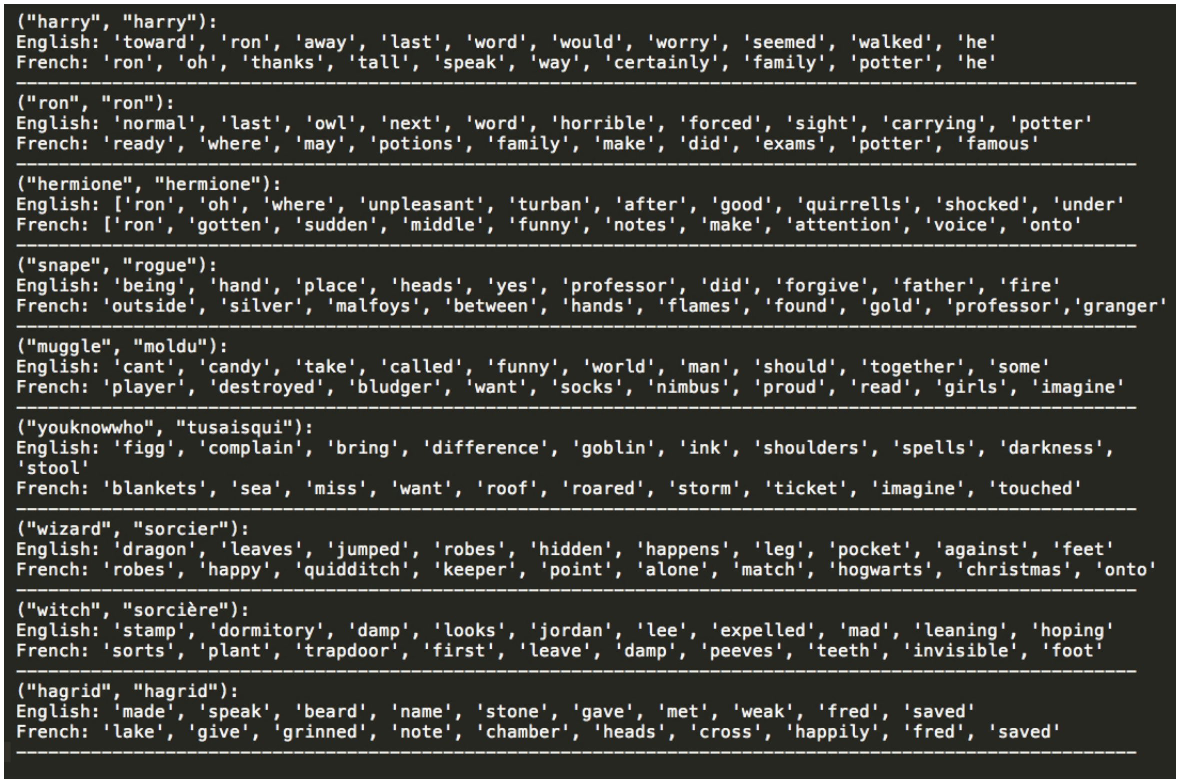


```
('en_uni3', 'fr_bi4'): 0.0326032013479
('en_unif2', 'fr_bi4'): 0.0326032013479
('en_bi2', 'fr_bi4'): 0.0319292333614
('en_bi3', 'fr_bi4'): 0.0319292333614
('en_bi4', 'fr_bi4'): 0.0293176074136
('en_uni3', 'fr_bi3'): 0.0293176074136
('en_uni4', 'fr_bi4'): 0.0286436394271
('en_uni3', 'fr_uni4'): 0.0285593934288
('en_bi4', 'fr_bi3'): 0.0283066554339
('en_bi4', 'fr_uni4'): 0.0281381634372
('en_bi3', 'fr_bi3'): 0.027801179444
('en_unif2', 'fr_bi3'): 0.027801179444
('en_uni3', 'fr_uni3'): 0.0276326874473
('en_bi2', 'fr_uni3'): 0.027548441449
('en_uni3', 'fr_unif3'): 0.0274641954507
('en_uni2', 'fr_bi2'): 0.0272957034541
('en_uni2', 'fr_bi4'): 0.0272114574558
('en_uni2', 'fr_uni2'): 0.0270429654591
('en_bi2', 'fr_unif2'): 0.0269587194608
('en_bi4', 'fr_unif3'): 0.0269587194608
('en_uni4', 'fr_uni4'): 0.0268744734625
('en_bi2', 'fr_bi3'): 0.0265374894693
('en_uni4', 'fr_bi3'): 0.0264532434709
('en_bi2', 'fr_uni2'): 0.0260320134794
('en_unif2', 'fr_uni3'): 0.0256107834878
('en_bi4', 'fr_uni3'): 0.0256107834878
('en_uni2', 'fr_bi3'): 0.0256107834878
('en_uni2', 'fr_uni3'): 0.0255265374895
```

```
('en_unif2', 'fr_uni2'): 0.0254422914912
('en_bi2', 'fr_bi2'): 0.0254422914912
('en_uni2', 'fr_unif2'): 0.0253580454928
('en_bi3', 'fr_uni3'): 0.0248525695029
('en_uni3', 'fr_uni2'): 0.0247683235046
('en_bi2', 'fr_uni4'): 0.024599831508
('en_uni3', 'fr_bi2'): 0.0245155855097
('en_unif2', 'fr_uni4'): 0.0244313395114
('en_bi3', 'fr_uni4'): 0.0242628475147
('en_uni4', 'fr_unif3'): 0.0241786015164
('en_unif2', 'fr_unif2'): 0.0239258635215
('en_bi3', 'fr_unif3'): 0.0237573715249
('en_unif2', 'fr_bi2'): 0.0233361415333
('en_uni3', 'fr_unif2'): 0.023251895535
('en_uni4', 'fr_uni3'): 0.02299915754
('en_unif3', 'fr_uni4'): 0.02299915754
('en_uni2', 'fr_uni4'): 0.0229149115417
('en_uni4', 'fr_bi2'): 0.0225779275484
('en_unif3', 'fr_bi4'): 0.0221566975569
('en_bi2', 'fr_unif3'): 0.0218197135636
('en_bi3', 'fr_unif2'): 0.0218197135636
('en_uni4', 'fr_uni2'): 0.0215669755687
('en_uni4', 'fr_unif2'): 0.021398483572
('en_unif3', 'fr_bi3'): 0.0212299915754
('en_unif2', 'fr_unif3'): 0.0207245155855
('en_bi4', 'fr_unif2'): 0.0203875315922
('en_bi4', 'fr_bi2'): 0.0201347935973
```

```
('en_uni2', 'fr_unif3'): 0.0201347935973
('en_unif3', 'fr_unif3'): 0.0199663016007
('en_unif3', 'fr_uni3'): 0.0198820556024
('en_bi3', 'fr_bi2'): 0.0196293176074
('en_bi3', 'fr_uni2'): 0.0192080876158
('en_bi4', 'fr_uni2'): 0.0177759056445
('en_unif3', 'fr_unif2'): 0.0172704296546
('en_unif3', 'fr_bi2'): 0.0164279696714
('en_unif3', 'fr_uni2'): 0.0160067396799
('en_bi3', 'fr_unif4'): 0.0132266217355
('en_unif3', 'fr_unif4'): 0.0128053917439
('en_unif2', 'fr_unif4'): 0.012299915754
('en_uni3', 'fr_unif4'): 0.0122156697557
('en_unif4', 'fr_unif4'): 0.0119629317607
('en_unif4', 'fr_unif2'): 0.0117101937658
('en_unif4', 'fr_uni2'): 0.0117101937658
('en_uni4', 'fr_unif4'): 0.0116259477675
('en_uni2', 'fr_unif4'): 0.0115417017692
('en_bi2', 'fr_unif4'): 0.0112047177759
('en_unif4', 'fr_uni4'): 0.0111204717776
('en_unif4', 'fr_bi3'): 0.0110362257793
('en_unif4', 'fr_bi4'): 0.0110362257793
('en_unif4', 'fr_bi2'): 0.0106149957877
('en_bi4', 'fr_unif4'): 0.0101937657961
('en_unif4', 'fr_uni3'): 0.0101095197978
('en_unif4', 'fr_unif3'): 0.0100252737995
```

Figure 14: *Each of the Semantic Vector Pairs and Associated k-Recall*



```
("harry", "harry"):
English: 'toward', 'ron', 'away', 'last', 'word', 'would', 'worry', 'seemed', 'walked', 'he'
French: 'ron', 'oh', 'thanks', 'tall', 'speak', 'way', 'certainly', 'family', 'potter', 'he'
------------------------------------------------------------------------------------------------
("ron", "ron"):
English: 'normal', 'last', 'owl', 'next', 'word', 'horrible', 'forced', 'sight', 'carrying', 'potter'
French: 'ready', 'where', 'may', 'potions', 'family', 'make', 'did', 'exams', 'potter', 'famous'
------------------------------------------------------------------------------------------------
("hermione", "hermione"):
English: ['ron', 'oh', 'where', 'unpleasant', 'turban', 'after', 'good', 'quirrells', 'shocked', 'under'
French: ['ron', 'gotten', 'sudden', 'middle', 'funny', 'notes', 'make', 'attention', 'voice', 'onto'
------------------------------------------------------------------------------------------------
("snape", "rogue"):
English: 'being', 'hand', 'place', 'heads', 'yes', 'professor', 'did', 'forgive', 'father', 'fire'
French: 'outside', 'silver', 'malfoys', 'between', 'hands', 'flames', 'found', 'gold', 'professor','granger'
------------------------------------------------------------------------------------------------
("muggle", "moldu"):
English: 'cant', 'candy', 'take', 'called', 'funny', 'world', 'man', 'should', 'together', 'some'
French: 'player', 'destroyed', 'bludger', 'want', 'socks', 'nimbus', 'proud', 'read', 'girls', 'imagine'
------------------------------------------------------------------------------------------------
("youknowwho", "tusaisqui"):
English: 'figg', 'complain', 'bring', 'difference', 'goblin', 'ink', 'shoulders', 'spells', 'darkness',
'stool'
French: 'blankets', 'sea', 'miss', 'want', 'roof', 'roared', 'storm', 'ticket', 'imagine', 'touched'
------------------------------------------------------------------------------------------------
("wizard", "sorcier"):
English: 'dragon', 'leaves', 'jumped', 'robes', 'hidden', 'happens', 'leg', 'pocket', 'against', 'feet'
French: 'robes', 'happy', 'quidditch', 'keeper', 'point', 'alone', 'match', 'hogwarts', 'christmas', 'onto'
------------------------------------------------------------------------------------------------
("witch", "sorcière"):
English: 'stamp', 'dormitory', 'damp', 'looks', 'jordan', 'lee', 'expelled', 'mad', 'leaning', 'hoping'
French: 'sorts', 'plant', 'trapdoor', 'first', 'leave', 'damp', 'peeves', 'teeth', 'invisible', 'foot'
------------------------------------------------------------------------------------------------
("hagrid", "hagrid"):
English: 'made', 'speak', 'beard', 'name', 'stone', 'gave', 'met', 'weak', 'fred', 'saved'
French: 'lake', 'give', 'grinned', 'note', 'chamber', 'heads', 'cross', 'happily', 'fred', 'saved'
------------------------------------------------------------------------------------------------
```

Figure 15: 10 *Nearest Neighbors for Words in* ({*English, French*}, *uniform*, $r = 2$)

6. Conclusions

We summarize the results of this paper as follows:

1. We modified a CPCA Hebbian learning network to take word order into account when learning semantic vectors for a parallel corpus across two languages: Namely, the English and French editions of *Harry Potter and the Philosopher's Stone*. We also defined and implemented several metrics for assessing word vector invariance across language, and tested them.

2. We noticed that for all metrics, the French uniform vectors were not as good as the bimodal and unimodal vectors for $r = 2, 3$, possibly suggesting that French is not as amenable to a bag-of-words model as English is (which for two of the metrics performed well with uniform distribution at $r = 2$).

3. Based on the t-SNE plots, the $r = 4$ case seems to be poor for uniform distributions, but rather better for the bimodal and unimodal distributions. The cause of this phenomenon may be due to the lesser weighting given to words in the middle (for bimodal) and words on the ends (for unimodal). The smaller amount of overall activation could potentially counteract the increased window size of the sliding context.

4. We noticed that the shape of the ALSD and Procrustes' distance distribution values were strikingly similar, though very oddly, the order of the vector set pairs was to some extent reversed.

5. The Procrustes' metrics behave differently, but have the commonality that (French, uniform, $r = 2, 3, 4$) receives low scores on both the distance and ratio metrics.

6. The seemingly simplest data point, (English, uniform, $r = 2$); (French, uniform, $r = 2$) proved to be one of the most confusing points in the set. It experienced excellent performance in the ALSD and Procrustes' ratio metrics, and the worst performance in the Procrustes' distance metric. Examining the t-SNE plots do not suggest anything out of the ordinary for these two vector sets. The strangely terrible performance on the Procrustes' distance metric suggests that the Procrustes' distance metric should potentially be modified, and that this metric is not currently capturing the language-invariant properties we desire. In fact, it does make more intuitive sense that we should examine the change in Procrustes distance as opposed to the final distance, because it is a large change that should encode the notion of the existence of a good linear map between semantic vectors across languages.

7. Future Work

This paper is but a preliminary foray into the evaluation of semantic vector spaces by comparing performance across parallel corpora. Furthermore, extensions to the Hebbian neural net to improve sentence-level semantics are also possible. We enumerate a few possibilities for future projects in this section.

7.1. The Relationship between ALSD and Procrustes' Distance

More work should be performed to elucidate the strange inverse relationship between these two defined metrics. It seems possible that Frobenius distances are not relevant at all in measuring semantics, but the oddly similar distribution and the essential inversion of the order statistic of the semantic vector set pairs suggests that there could be something more interesting to discover upon further investigation.

7.2. Varying Network Parameters

We would like to see which number for k produces the best cross-lingual representation in terms of our various metrics. In some sense, the optimal k would show what the necessary size of a subset of words is in the algorithm. In other words, the optimal k would give a sense of the average number of words in the corpus that are relevant to a given word's meaning across both languages. We could also experiment with the size of $\mathcal{H}$, test out other context window distributions $\mathcal{D}$, and vary r to larger values to see the effects of these changes (though based on the experimental evidence from this paper, it seems that larger r would only hurt the ALSD metric, at least).

7.3. Establishing Theoretical Connections Between the Network and the Cross-Lingual Objectives

It would be a fantastic result if by modifying the Hebbian network further, it were possible to establish theoretical guarantees on performance according to some cross-lingual objective function as defined in this paper. We have already laid the groundwork for potential objective functions that obey the maxim "meaning is invariant across language", but more work must be done to find or show that these objectives are directly connected to what the Hebbian network is learning. In fact, it is likely that the objectives are not connected. If this is in fact the case, then it would be ideal to find an objective that is neuroscientifically plausible as the Hebbian network is that encodes this maxim (varying from the traditional Distributional Hypothesis of Meaning).

7.4. Non-Neuroscientific Cross-Lingual-Based Objectives for Training Semantic Vectors

Semantic vectors in the literature are primarily trained according to an objective determined from distributional properties across a single language, whether theoretically proven valid or not. It would be interesting to change the definition of "semantically similar" to a definition that takes into account meaning across language (for instance, the Language Similarity Distance or the Procrustes' Distance defined in this paper) as an objective. Perhaps it is possible to define a convex objective in terms of these functions, and if not (likely), then perhaps further theoretical analyses in the vein of [Arora2015] could provide guarantees for semantic vectors with different properties. These language-invariant semantic vectors could improve machine translation tasks if applied to networks as defined in [Sutskever2014].

7.5. Testing Other Semantic Vectors

There are available word vector sets online - namely from word2vec, GLoVE, and others [Mikolov2013], [Pennington2014], [Arora2015]. We simply note that interesting future work could examine the properities we analyzed in this paper for these other word vector sets. Some work has been done analyzing their properties, but to the knowledge of the author, there have been no analyses of cross-lingual preservation performance (which is what this paper seeks to analyze for the Hebbian vectors). Notably, these vector sets have proven empirical performance so it would be interesting to see if the metrics they are trained on are also language-invariant.

7.6. Unifying the Similarity Metrics

Currently, the similarity metrics defined in Section 3 are all used independently as order statistics. It would be interesting to see if these various metrics could be unified in a sensible manner, perhaps creating a better objective for learning vectors in the process.

7.7. Treating Brain Activity as $\mathcal{L}_2$

The most sensible way to build word vectors that represent meaning most similarly to brain would be to define an objective that seeks to minimize distance between brain vector representations and word vector representations to produce a better semantic vector overall. In some sense, this approach is taken by [Fyshe2014]. More rigorous work in this area relating the objective function they use to an analagous "semantic invariance across brain and language" type property would be quite interesting to see.

8. Acknowledgements

This work was produced as a final project for NEU 330, Connectionist Models as taught at Princeton University in Spring 2015. I would like to thank Professor Ken Norman, who was a great help in discussions and in giving pointers to relevant books, papers, and resources. I would also like to thank Francisco Pereira, who gave additional helpful guidance and pointers to other resources and critiqued the initial design of the model. Finally, I would like to thank Pavlos Kollias, who read the inital proposals and gave helpful feedback.

9. Appendix I: How to Build the Code

The code for this project was written entirely in Python. If you do not have Python installed, it will be necessary to install it on your computer. The code is entirely self-contained outside of the modules it will be necessary to install, and is available on Github.

9.1. Brief Instructions on Python Installation

If you have a *NIX computer, then Python should already be installed. Access a terminal and type "python" to see what version it uses. The code in this paper depends on Python 2.7. The Pip package installer is recommended for downloading new modules. If you have a Windows computer, then there are many Python distributions available for Windows that come pre-packaged: For instance, the Anaconda distribution.

Once you have Pip, you can install most modules by simply entering "pip install <name of module goes here>" into the command line. IPython is also recommended as the Python shell to use for running code.

9.2. Module Dependencies

Here is a list of modules you may have to download the most recent version for:

1. Numpy.
2. Scikit-learn.
3. PyEnchant.
4. Matplotlib.

These should all be fairly straightforward to install if you do not already have them installed. The dependencies for each file of code are present at the top of each file.

9.3. Running the Code

Each python file (ends with ".py") has well-commented functions that explain what all the function inside do. In order to run a function, first open up the python environment by either typing "python" or "ipython" in Terminal. In order to run a function from a specific .py file, you will need to import it while you are inside the Python shell. You can import a specific function "foo" from a module "bar.py" as follows:

```
>> from bar import foo
```

We can also import everything from a module as follows:

```
>> from bar import *
```

If we are using IPython, then it often makes more sense to load the module as follows:

```
>> import bar as b
>> b.foo()
```

Note that we have to prefix the name of the module before the function now. You can also access variables that are inside the file in this manner.

The other kind of file found in the code are ".p" files, which are Pickle files. Pickle is a Python module which saves Python objects into a file so that they can be loaded later. The syntax for using this module is as follows:

```
>> import pickle
>> myfile = pickle.load(open("myfile.p", "rb"))
>> pickle.dump(myfile, open("myfile2.p", "wb"))
```

"rb" and "wb" stand for reading from and writing to a file respectively.

A brief tutorial on Numpy and some basic Python data structures may also be helpful when dealing with the code presented.

```
>> import numpy as np
>> a = np.array([1, 2, 3, 4])
>> np.shape(a) # returns the shape of the array, in this case, (4, 0)
>> M = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>> np.shape(M) # returns the shape of the matrix, in this case (3, 3)
>> c = [1, 2, 3] # a python array list
>> len(c) # returns the length of the array/tuple/dictionary
>> d = dict([('a', 1), ('b', 2), ('c', 3)]) # a dictionary mapping letters to numbers
>> d['a'] # prints out 1
>> d['b'] # prints out 2
>> s = set()
>> s.add(1)
>> s.add(2)
>> len(s) # prints 2
>> 2 in s # prints True
>> 3 in s # prints False
>> s.add(2)
>> len(s) # still is 2
```

As for the specific files, the main Python files are as follows:

```
neural_net.py: Contains the implementation of the Hebbian net architecture.

wordvec.py: Defines the various distributions and nonlinearities,
and also the cosine similarity function. The main method is a method to
build a set of word vectors for a given text distribution, and r.
Furthermore this file was used to split up the learning work
across several computers.

analysis.py: Contains a boatload of functions for evaluating the various metrics.
Heavily commented.

procrustes.py: Implements the Procrustes transformation.

build_corpus.py: Was used to preprocess the corpora.
Indicates some of the processing involved.
```

```
goog_translate.py: Used as a utility to do automatic translation in code.
Scrapes off of Google Translate.

saving.py: A utility to automatically save images plotted from matplotlib
into a folder.

progressbar.py: A nice visualization of how fast the network is taking to learn.
```

hp1_en.txt and hp1_fr.txt are the original text files for the two books. List versions of each book are stored inside hp1_text_en.p, hp1_text_fr.p respectively. The 18 semantic vector spaces for each of the parameter permutations are stored inside hp1_vecs_(lang)_(distribution)(r).p. translation_dict.p stores the English-to-French dictionary for the reduced wordsets. Other .p files are stored for security, they are not as important.

References

[Arora2015] Arora, S., Li, Y., Liang, Y., Ma, T. and Risteski, A. Random Walks on Context Spaces: Towards an Explanation of the Mysteries of Semantic Word Embeddings. (2015). At <http://arxiv.org/abs/1502.0352>.

[Bengio2003] Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. A Neural Probablistic Language Model. *Journal of Machine Learning Research*. **3,** 1137 − 1155. (2003).

[Fyshe2013] Fyshe, A., Talukdar, P., Murphy, B., and Mitchell, T. Documents and Dependencies: an Exploration of Vector Space Models for Semantic Composition. (2013). At <http://www.cs.cmu.edu/ ∼ afyshe/papers/conll2013/deps_and_docs.pdf>.

[Fyshe2014] Fyshe, A., Talukdar, P., Murphy, B., Mitchell, T. Interpretable Semantic Vectors from a Joint Model of Brain- and Text-Based Meaning. (2014). At <http://www.cs.cmu.edu/ ∼ afyshe/papers/acl2014/jnnse_acl2014.pdf>.

[Hassan2012] Hassan, S., Banea, C., Mihalcea, R. Measuring Semantic Relatedness Using Multilingual Representations. *First Joint Conference on Lexical and Computational Semantics (*SEM)*. 20 − 29. (2012). At <http://www.aclweb.org/anthology/S12-1003>.

[Just2008] Just, M. What Brain Imaging Can Tell Us About Embodied Meaning. In *Symbols and Embodiment: Debates on Meaning and Cognition*. Eds. de Vega, M., Glenberg, A., Graesser, A. 75 – 84. (2008). At <http://www.ccbi.cmu.edu/reprints/just_garachico-chapter_ccbi-reprint.pdf>.

[Ménard1998] Rowling, J. K. Translated by Jean-François Ménard. *Harry Potter à l'école des sorciers*. Éditions Gallimard. (1998).

[Mikolov2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. (2013). At <http://arxiv.org/abs/1301.3781>.

[O'Reilly2000] O'Reilly, R., Munakata, Y. *Computational Explorations in Cognitive Neuroscience*. Massachussets Institute of Technology. (2000).

[O'Reilly2012] O'Reilly, R. C., Munakata, Y., Frank, M. J., Hazy, T. E., and Contributors. *Computational Cognitive Science*. Wiki Book, 1st Edition. (2012). At <http://ccnbook.colorado.edu>.

[Pennington2014] Pennington, J., Socher, R. and Manning, C.D. GloVe: Global Vectors for Word Representation. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing*. (2014).

[Pulvermüller1999] Pulvermüller, F. Words in the Brain's Language. *Behavioral and Brain Sciences*. **22,** 253 – 336. (1999).

[Rowling1997] Rowling, J. K. *Harry Potter and the Philosopher's Stone*. London: Bloomsbury Children's. (1997).

[Seung2015] Seung, H.S. Course Notes for COS 598C, Spring 2015. At <http://seunglab.org/courses/>

[Sutskever2014] Sutskever, I., Vinyals, O., Le, Q. Sequence-to-Sequence Learning with Neural Networks. (2014). At <http://arxiv.org/abs/1409.3215>.

[Turney2010] Turney, P.D. and Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*. **37,** 141 – 188. (2010).

[van der Maaten2008] van der Maaten, L. and Hinton, G. Ed. Bengio, Y. Visualising Data using *t*-SNE. *Journal of Machine Learning Research*. **9,** 2579 – 2605. (2008).