Aligned Word Vector Spaces and Document Vectors

A thesis submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Shashwat Chandra

to the
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
July, 2015

CERTIFICATE

It is certified that the work contained in the thesis titled **Aligned Word Vector Spaces and Document Vectors**, by **Shashwat Chandra**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof. Amitabha Mukerjee

Department of Computer Science & Engineering

IIT Kanpur

July, 2015

ABSTRACT

Name of student: **Shashwat Chandra** Roll no: **13111059**

Degree for which submitted: **Master of Technology**

Department: **Computer Science & Engineering**

Thesis title: **Aligned Word Vector Spaces and Document Vectors**

Name of Thesis Supervisor: **Prof. Amitabha Mukerjee**

Month and year of thesis submission: **July, 2015**

Many machine learning algorithms require fixed-length vectors as input. There have been many approaches to generate fixed-length vectors representing text documents and words in a vocabulary. However most of these techniques are restricted by the vocabulary of the corpus, the corpus size, and the training time. We present a novel approach to align different mappings from a vocabulary space to a d-dimensional feature space. We discuss and present two approaches, comparing local alignment and global alignment of the feature spaces. We show that the smoothing inherent in such a process fixes some of the problems mentioned above, and gives better results than the original word vectors. This alignment approach has applications in aligning parallel corpora, and working with massive corpora.

We also present a highly customizable unsupervised algorithm that extends the existing Global Vectors algorithm to learn feature vectors for variable-length texts, like sentences, paragraphs, and documents.

To my brother and my parents

Acknowledgements

I would like to start off by thanking my thesis supervisor, Dr. Amitabha Mukerjee for his constant support and encouragement. Without his aid and guidance, this work would not have been possible. I would like to thank the faculty, staff, and students of the Department of Computer Science and Engineering at IIT Kanpur for helping me with any problems I faced. I would especially like to thank my friends at the Brain and Manifolds Group, especially M. S. Ram, Ashudeep, Pranjal, Anirban, and Debojyoti, for their invaluable help.

I would especially like to thank my friends for letting me bounce ideas off them, and for all their help during my stay on campus. I would specifically like to thank Swapnil Mahajan, Ayan Ghatak, Rishabh Vaid, Anurag Sahay, Rajdeep Das, Ritika Verma, Richa Sharma, Nimisha Agarwal, Shail Dabas, Satyajit Bhadange, Chirayu Modi, Mohit Bhadade, Sanjana Gopal, Divya Goel, Shashank Bhandari, Gundeep Kaur and Iffat Siddiqui. I would like to thank the entire CS group, and the SPO group, specifically.

I thank the Flying Spaghetti Monster for his noodly guidance. Lastly, this would not have been possible without the constant support and guidance of my entire family, to whom I am deeply indebted.

Contents

List of Tables **xiii**

List of Algorithms **xv**

List of Figures **xvii**

1 Introduction **1**

1.1 Word Vectors versus Traditional Approaches 1

1.2 Phrase, Sentence, and Document Vectors 3

1.3 Multi-Word Vectors 4

1.4 Contributions 5

2 Alignment of Vector Spaces **7**

2.1 The Problem of Vector Alignment 9

2.1.1 Congruent Mapping 10

2.1.2 Globally Linear Equivalence 11

2.1.3 Local Equivalence 12

2.2 Globally Linear Approach 13

2.2.1 Results of the Globally Linear Approach 15

2.3 Local Linear Neighbourhood Approach 17

2.3.1 Results of Linear Neighbourhood Approach 19

2.3.2 Word Analogy Dataset 20

2.3.3 Testing on the Word-Analogy dataset 22

3 Paragraph Vectors for GloVe **27**

3.1 Introduction 27

3.1.1 Paragraph Vectors 27

3.1.2 Word Vectors using GloVe 28

3.2 The GloVe algorithm for word vectors 28

3.3 Extending to Paragraph Vectors 29

3.4 Equations 30

3.5 Baseline comparison algorithms 32

3.5.1 Bag of Words 32

3.5.2 Paragraph Vectors - Distributed Memory Model 32

3.5.3 Word Vector Averaging 33

3.5.4 Word Vector Weighted Averaging 33

3.5.5 Clustering based on Chinese Restaurant Process 33

3.6 Datasets and Corpora 35

3.6.1 English Wikipedia Corpus 35

3.6.2 Hindi Wikipedia Corpus 36

3.6.3 Wikipedia Paragraph Dataset 36

3.6.4 SEMEVAL 2014 - Task 3 36

3.7 Testing and Results 37

3.7.1 Wikipedia Datasets 37

3.7.2 SemEval Dataset 40

4 Conclusions **41**

4.1 Summary 41

4.2 Scope for further work 42

References **43**

List of Tables

2.1 Details about corpora used in the Matrix Alignment Tests 8
2.2 Results of the Globally Linear Approach. The aligned points are the most frequent aligned points in $\hat{V}$. Testing is done on a set of words not used for training. 17
2.3 Nearest Neighbours of vectors generated by the Globally Linear Approach. Here the rank of the words is based on their frequency in the combined corpora. These are the same words used by Collobert and Weston, 2008b. 18
2.4 Results of the Linear Neighbourhood Approach 20
2.5 Nearest Neighbours of vectors generated by the Linear Neighbourhood Approach. Here the rank of the words is based on their frequency in the combined corpora. These are the same words used by Collobert and Weston, 2008b. 21
2.6 Accuracies on Word-Analogy task. 23
2.7 Eigenvalues of Principal Components for some example word-vectors in the word-analogy task. The Predicted vectors are calculated using the Linear Neighbourhood Approach 24

3.1 Statistics of the Wikipedia Paragraph datasets 36
3.2 Comparison on different function combinations for the Hindi Wikipedia task in the GloVe Paragraph Vectors 38
3.3 Comparison of different approaches on Hindi Wikipedia task 39
3.4 Comparison of different approaches on English Wikipedia task 39

3.5 Comparison of different approaches on SemEval 2014 subtask 40

List of Algorithms

2.1 Globally Linear Approach 14
2.2 Linear Neighbourhood Approach 19
3.1 Clustering - Variant 1 34
3.2 Clustering - Variant 2 35

List of Figures

2.1 The system is tested on two corpora, using two different techniques: (1) GloVe and (2) Word2Vec. Thus, embedding W_{12} is one created using Corpus 1 and Technique 2. The thick lines show the mappings (from one embedding to another) that we have tested our approaches on. 9

2.2 Vocabulary Alignment mappings. Note that the (2.2c) manifold is a transformation of (2.2b) 13

2.3 We align the m-dimensional submanifold using v_t^i and v_s^i and use these alignments to calculate the mapping of w_s in t-space 18

2.4 We present vectors for some examples in the word-analogy syntactic subtask. Well-calculated vectors will form a parallelogram, which can accurately be reduced to 2-dimensions. 25

2.5 We present vectors for some examples in the word-analogy semantic subtask. Well-calculated vectors will form a parallelogram, which can accurately be reduced to 2-dimensions. 26

3.1 Modifying the GloVe co-occurrence matrix 30

Chapter 1

Introduction

As the amount of unstructured text data increases in this world, due to the popularity of social networking, blogging, and news articles, the need to be able to intelligently parse through social media and news data, and collect relevant items of information greatly increases. There have been many approaches to make the task of a researcher simpler by formalizing a method of comparison of documents.

In the last few years, there has been a rapid growth of approaches that deal with language by creating vectors that represent words in the vocabulary, and also of vectors that attempt to represent the article has a whole. Vectors representing words or documents are required to be fixed-length so as to allow users to compare different vectors. A comparison of these vectors can provide information about semantic and syntactic similarity.

1.1 Word Vectors versus Traditional Approaches

Word Vectors are a remarkably successful approach to word similarity, since they permit a continuum model of similarity. This is in stark contrast with traditional approaches in Natural Language Processing, which uses discrete partitioning based on Part-of-Speech, and other linguistic properties. This treats all classes (ex. Noun) similarly, but, in an example such as

1. Our company is training workers

2. Our problem is training workers

the phrase 'training workers' may be a Verb Phrase, in sentence 1, but a Noun Phrase in sentence 2. The dominant parsers deal with this problem by specializing the phrase category by listing the head tag with it. So, sentence 1 would have VP(*training*) as its tag, while sentence 2 would have NP(*workers*). In such lexical PCFGs, like Collins, 1997, the set of phrasal tags soon approaches the size of the lexicon.

In linguistic theory also, the number of word classes have grown, for example, subcategorization lexicons detailing specific subcategories for different heads (Korhonen, Krymolowski, and Briscoe, 2006). However, this approach, even with hand-coding of tags, has lots of exceptions: For example, the Penn Treebank has several hundred alternate tags (Atwell et al., 2000). Even after analysing the language, linguists can use a diverse number of PoS tags from 45 to 100, for less inflected languages like English, to 1155 for the highly inflected old Greek of the New Testament (DeRose, 1988).

A similar difficulty holds for semantics. Propositional interpretations of language are discrete, and have suffered from difficulties similar to that of ontologies, where creating a rule in one part of the ontology often created contradictions in a very different part of the ontology, a problem known as brittleness (Gallant, 1988). Category boundaries are never crisp, and overlapping senses create problems for discrete partitions.

A continuous mapping of words to a vector space allows similar senses and syntactic usage patterns to share nearby regions of the vector space, which is precisely the aim of word-vector models like Schütze, 1993; Morin and Bengio, 2005; Deerwester, Dumais, Landauer, Furnas, and Harshman, 1990.

There have been many approaches of constructing word vectors. One of the first attempts at generating word vectors is Rumelhart, Hinton, and Williams, 1988, which used neural networks. Over time, the representations have gradually become more expressive of the semantics of the language. Recent approaches to generate word vectors include Mikolov, Chen, Corrado, and Dean, 2013; Pennington, Socher, and Manning, 2014; Collobert and Weston, 2008a; Turian, Ratinov, and Bengio, 2010.

Given a measure of semantic relatedness for words and articles, we can often use this for various practical applications such as Sentiment Analysis (Maas et al., 2011), Word Sense Disambiguation (Schütze, 1998), POS Tagging (Collobert and Weston, 2008b), and Document Clustering (Le and Mikolov, 2014).

1.2 Phrase, Sentence, and Document Vectors

Attempts to model language in the form of such fixed-length vectors falls in two broad categories: language dependent models, and language independent models. Language dependent models are highly constrained by the resources required in the form of language experts, and by the availability of cleaned, parsed, and tagged corpora; especially for relatively less-studied languages like Hindi.

Language independent models require large corpora for training, since their algorithms depend entirely on the frequency of occurrences of words in the corpora, and the quality of the sentences they occur in.

Since these vectors are independently learned based on each word's context in the corpus, the vectors can be noisy, and highly corpus-dependent. Additionally, considering the training speed of current algorithms, they do not make more than one pass over the corpus. Fast algorithms certainly give less scope for correlating the vectors, but, even so, the corpus length and computer memory certainly put a limit on the size of the vocabulary that can be trained at one time.

In this work, we propose a novel approach to merge fixed-length word vectors generated using different models. This allows us to merge the vectors generated using different, training settings, resulting in a smoothing of the aligned vectors. We demonstrate that local and global alignment techniques can allow us to map word vectors generated using one technique to word vectors generated using another technique. We test the created word vectors on word analogy tasks, and report better accuracies, due to the smoothing. In fact, the advantages of such smoothing has been reported by Pennington et al., 2014 in their approach towards context word-vectors. Our approach is a more general example of the same.

In addition to improving word-vector accuracies, this model can be used for parallel corpora mapping, and predicting bilingual relations. If we have an alignment for a few words, we can use this approach to extend that alignment to the entire vocabularies, irrespective of their language.

1.3 Multi-Word Vectors

Recent language independent models have focused on using word-vectors to learn mappings for multi-word phrases. Such mappings can be done to any vector space, but, ideally, we would like to map the sentence, phrase or document onto the same dimension as the word-vector space.

Such a problem has some theoretical problems to begin with. The distribution of words in the word vector space are already determined before one attempts to project phrases onto it, without disturbing the already existing word vectors. Given a vocabulary size $|V|$, the number of phrases of length k is $O(|V|^k)$. Thus the semantic space of larger phrases and paragraphs is far more fine-grained than the word space. Hence, one question that arises is: Are the constraints of the existing distribution rich enough to permit an useful representation of paragraph vectors?

Another difficulty arises due to the attempt at imposing this new mass of embeddings onto a space that is already mapped. It is well known that in situations involving non-linear maps, out-of-sample embedding is a difficult problem, and here one is attempting to embed not just a few more data points, but an exponentially larger class.

There are advantages to such a map, however, This will give us the ability to generate summaries and make comparisons between document senses and word senses. This is especially relevant for phrases (for example, a Noun Phrases may be close to a Noun), but it has some meaning for single or multiple sentences too.

Bengio, Ducharme, Vincent, and Janvin, 2003 expresses the possibility of using a joint probability function to use these word vectors to generate sentence vectors. Other approaches to generate document vectors include Z. S. Harris, 1954; Le and Mikolov, 2014. Latent Semantic Analysis (Dumais, Furnas, Landauer, Deerwester, and Harshman, 1988) also attempted to extract topics in documents and generate document vectors.

The Le and Mikolov, 2014 approach generates paragraph vectors by generalizing the approach used by Mikolov, Chen, et al., 2013. Similarly, we will be presenting an algorithm that generalizes the current state-of-the-art word-vector model, Pennington et al., 2014, to sentences and documents in a language independent manner. We modify the Pennington et al., 2014 model, and present a highly customizable approach using the equations governing n-gram probabilities.

1.4 Contributions

This thesis presents a number of contributions to existing work in word and document vector generation.

1. We present algorithms for combining vector-space mappings from multiple algorithms into one single map. We aim at getting advantages from these algorithms into one vector space. An important result of this is the smoothing of vectors generated using traditional algorithms, leading to an increase in the accuracy of the predicted vectors.

2. We present a language independent model for generation of document vectors, built on word-vector models. We test this model on a dataset created using English and Hindi Wikipedia. We also compare our results with meaningful word-averaging models. We use a collection of phrase-vector generation algorithms, and present the results for each.

Chapter 2

Alignment of Vector Spaces

Vector space representations of words are useful for a variety of applications ranging from named entity recognition to textual entailment. Most modern representations like Mikolov, Chen, et al., 2013; Pennington et al., 2014 encode word similarity by the distance between the words in the vector space. Unfortunately, the vocabulary mapped by these approaches is necessarily limited by the size of the corpus. Additionally, training time is dependant on the corpus size and/or the vocabulary size, leading to further difficulties in training over a large corpus.

In this chapter, we will discuss the alignment of vocabulary mappings to vector spaces generated using various approaches. Specifically, we will deal with mappings generated using the two algorithms, GloVe (Pennington et al., 2014), and Word2Vec - Skip Gram (Mikolov, Chen, et al., 2013) that map two different corpora C_1 and C_2 (both subsets of Wikipedia) to a manifold embedded in a vector space $\mathbb{R}^d$ (the results here are for $d = 200$). We will discuss approaches to merge these mappings into a single mapping.

Word vectors are learned from a corpus by considering the context in which each word appears. By maximising the contextual relevance of all the words together, one can solve, for a set of word vectors. Each technique does have its own biases, however, and even small perturbations in the parameters of the same technique can alter the values of these vectors. In fact, since the vectors are initialized randomly at the start, running the same code, on the same corpus, will give completely different vectors. The intuition that these word vector embeddings may have some underlying similarity is rooted in the fact that the contexts in which they appear in differing corpora may have some resemblance.

As a result, we can still make reasonable assumptions to try to estimate such a mapping between techniques. Given that the two techniques (Pennington et al., 2014 and Mikolov, Chen, et al., 2013) solve, with some degree of accuracy, the problem they are tasked to solve, we can assume that, for two words, w^i and w^j, the cosine distance between their vectors v^i and v^j is inversely related with their semantic similarity. Hence, in a local subspace, we expect the points and their neighbourhood to be the same, governed by the same semantic similarity rules. This indicates that one may expect at least some local alignment between neighbourhoods of the same word. If the entire mass of word vectors can, similarly, be related, we may expect some sort of a global mapping too.

Additionally, as Section 3.1 of Pennington et al., 2014 states, the algorithm followed by many approaches (ex. GloVe and Word2Vec) aims at minimizing the same kind of cost function. Their methods of minimizing the cost function are different, but their end goal is expected to be the same. Thus, we can expect there to be some sort of global alignment which may map one solution of the cost function to the other.

Corpus	Vocabulary Size	Vector Space Dimensionality	Notes
C_1	100,000	200	This corpus has been created by randomly sampling 75% of the English Wikipedia dataset. (3.2M articles, 1.2B words)
C_2	100,000	200	This corpus has been created by randomly sampling 75% of the English Wikipedia dataset. (3.2M articles, 1.2B words)

Table 2.1: Details about corpora used in the Matrix Alignment Tests

In this work, we test these ideas using the English Wikipedia corpus (3.6.1). We trained GloVe vectors on a subset of the corpus (C_1) and separately on another subset of the corpus (C_2). We looked for words w^i present in both mappings, V_s and V_t. We then calculated the transformation T and looked at the distance between v_t and $T(v_s)$.

We have tested these approaches on different combinations of corpus and technique. A summary of our approaches is in Figure 2.1. Here, the results of a W_{12} to W_{21} mapping imply the Word2Vec algorithm was run on corpus C_1 to get the s-space; the GloVe algorithm was run on corpus C_2 to get the t-space; and the results of the mapping from s to t are displayed.

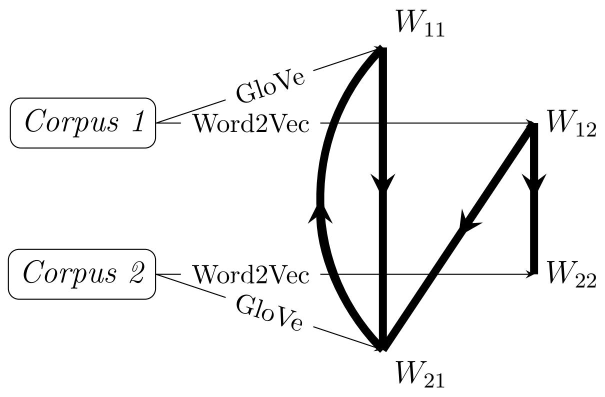


Figure 2.1: The system is tested on two corpora, using two different techniques: (1) GloVe and (2) Word2Vec. Thus, embedding W_{12} is one created using Corpus 1 and Technique 2. The thick lines show the mappings (from one embedding to another) that we have tested our approaches on.

2.1 The Problem of Vector Alignment

Given a corpus C_s, with a vocabulary of V_s, we can use a technique to generate a vector belonging to $\mathbb{R}^d$ for every word in V_s. Let us represent these vectors as a $|V_s| \times d$ matrix, W_s.

Similarly, we have a corpus C_t with vocabulary V_t giving us a matrix W_t. Let $\hat{V} = V_s \cap V_t$; and let $\hat{W}_s$ and $\hat{W}_t$ be the maps using C_s and C_t for words in $\hat{V}$. Our task is to find a function T, that maps $\hat{W}_s$ to $\hat{W}_t$ for words $w \in \hat{V}$.

$$W_t \sim T(W_s) \qquad \text{for words } w \in \hat{V} = V_s \cap V_t \tag{2.1}$$

We can then use this transformation T to approximate the mapping of all words in $V_s \setminus V_t$ within the manifold created for C_t. Hence, for a word w_s, we can use its vector v_s to calculate an approximation

$$\tilde{v}_t = T(v_s) \qquad \forall w_s \in V_s \setminus V_t \tag{2.2}$$

To calculate the alignment in equation (2.1), we need to look at how we can calculate the mapping $T()$. If this mapping is linear, we may represent the operation as $W_t = T \cdot W_s$.

Notation

Let the two techniques for generating word vectors be s (the technique generating the source space) and t (the technique generating the target space).

s acts on a corpus C_s (with vocabulary V_s). t acts on a corpus C_t (with vocabulary V_t).

Words are represented by w. A certain word i in vocabulary V_s will be represented as $w_s^i \in V_s$. If a word exists in $\hat{V} = V_s \cap V_t$, the subscript is dropped ($w^i \in \hat{V}$)

Word vectors are represented as $v \in \mathbb{R}^d$. A vector for word w^i created using technique s will be represented by v_s^i and a vector for word w^i created using technique t is represented by v_t^i.

The matrix of vectors created using a technique s (on corpus C_s) is represented by W_s which is a matrix of size $|V_s| \times d$. Each row i in W_s corresponds to word w^i and is equivalent to the vector v_s^i. $\hat{W}_s$ represents the matrix of vectors within $\hat{V}$.

The s-space is the subspace within the $\mathbb{R}^d$ space where the vectors v_s lie.

We next consider several levels of alignment that may be valid for the mapping we generate.

2.1.1 Congruent Mapping

If we look at the mapping function $T()$, transforming the entire s-space into the t-space, we make certain assumptions on this function:

1. The transformation should preserve all relative distances:

   Given a distance metric `dist`, and vectors v^i for words w^i in a vocabulary, $\exists \alpha \in \mathbb{R}$ (a scale factor) such that

$$\forall w^i, w^j \in \hat{V}, \quad \frac{\text{dist}(v_s^i, v_s^j)}{\text{dist}(v_t^i, v_t^j)} = \alpha \tag{2.3}$$

2. If $T()$ is a linear mapping, this can be represented by the following equation (for small $\epsilon \geq 0$) where T is restricted to rigid transformations (excluding scale, $det(T) = \pm 1$).

$$||T \cdot \hat{W}_s - \hat{W}_t||_F < \epsilon \tag{2.4}$$

This is a very strong equivalence, and we initially did not expect this equivalence to hold in our case.

2.1.2 Globally Linear Equivalence

A mapping from W_s to W_t is a global equivalence if there exists a universal function $T()$ such that

$$T(W_s) = W_t \tag{2.5}$$

Here, $T()$ has the same form all over the V_s space, and can be any general function satisfying the mapping

$$||T(\hat{W}_s) - \hat{W}_t||_F < \epsilon \tag{2.4}$$

If we assume that T is a linear transformation, we can write $T(v_s^i) = M \cdot v_s^i + b$ for unknown M and b, where M is an affine transformation and b is the translation vector. This, like Congruent Mapping, is a global equivalence, but a weaker one. We did not expect this to work, however, our results surprised us.

2.1.3 Local Equivalence

Rather than looking at an approach that aims for global equivalence, we can also try to achieve piecewise alignment for local regions. For example, one could have differing maps for differing neighbourhoods in V_s and try to preserve some metric over that locality in our transformation.

A piecewise linear model can be constructed based on the local neighbourhoods. We look at the k-nearest neighbours of a word w_s^i from vocabulary V_s. Let N_i^k denote the neighbourhood of v_s^i that contains only its k nearest points, as measured by cosine distance. We reconstruct v_s^i by a linear function of its neighbours, v^j

$$\hat{v_s^i} = \sum_{j=1}^{n} a_{ij} v^j \tag{2.6}$$

where a_{ij} is a scalar weight for the neighbour v^j with unit sum $\sum_j w_{ij} = 1$, for translational invariance. If $v^j \notin N_i^k$, then $a_{ij} = 0$.

This representation can be thought of as modelling the tangent space at v_s^i. The entire space is then modelled as a collection of such neighbourhood tangent spaces. (2.2).

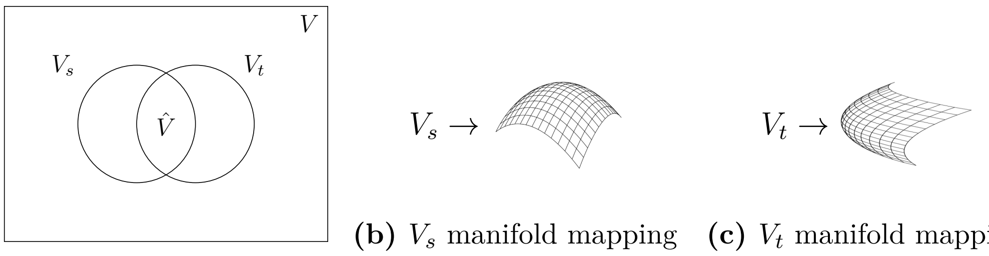
When we look at the k-nearest neighbours of a word w_s^i from vocabulary V_s, this neighbourhood can be represented in matrix form as an m ($m < k$) dimensional submanifold in the following manner

$$\hat{X}_s \approx A \cdot X_s \tag{2.7}$$

where $A = (a_{ij})$ is a sparse $n \times n$ matrix of weights, and $X_s = (v_s^1, v_s^2, \ldots, v_s^n)^T$, only k of which are non-zero.

We can align the linear system of equations in the t-space instead of the s-space to get

$$\hat{X}_t \approx A \cdot X_t \tag{2.8}$$



(a) Vocabulary Space **(b)** V_s manifold mapping **(c)** V_t manifold mapping

Figure 2.2: Vocabulary Alignment mappings. Note that the (2.2c) manifold is a transformation of (2.2b)

This approach is equivalent to a weighted averaging over the neighbourhood of the word vector. A similar simple average technique would give

$$\hat{X}_t \approx A' \cdot X_t \tag{2.9}$$

where A' is a sparse $n \times n$ matrix, such that $A'_{ij} = 1$ if $w_j \in N_i^k$ and $A'_{ij} = 0$ otherwise.

2.2 Globally Linear Approach

In this algorithm we aim to calculate the transformation matrix minimizing the error $||\hat{W}_t - M \cdot \hat{W}_s||$ assuming a global equivalence. We need to calculate M, given W_s and W_t in (2.4).

Given exactly $d + 1$ aligned points, we can find an exact solution to the following equation. Here, M is a $d \times d$ matrix representing the linear map of the vectors in W_s, and b is a $d \times 1$ vector representing the translation of the vectors.

$$W_t = M \cdot W_s + b \tag{2.10}$$

We can augment the M matrix to get a linear map using a single matrix multiplication.

$$\begin{bmatrix} W_t \\ 1 \dots 1 \end{bmatrix} = \left[\begin{array}{c|c} M & b \\ 0 \dots 0 & 1 \end{array}\right] \cdot \begin{bmatrix} W_s \\ 1 \dots 1 \end{bmatrix} \tag{2.11}$$

Algorithm 2.1 Globally Linear Approach

```
Let $\tilde{V} \subset \hat{V}$ s.t. $|\tilde{V}| = N$
Let $ctr \leftarrow 0$
Let $W_s \leftarrow Matrix[d+1, N]$
Let $W_t \leftarrow Matrix[d+1, N]$
**for** $w^i \in \tilde{V}$ $(i = 1, \ldots, N)$ **do**
    $v_s^i \leftarrow s(w)$
    $v_t^i \leftarrow t(w)$
    $W_s[1 \ldots d, i] \leftarrow v_s^i$
    $W_t[1 \ldots d, i] \leftarrow v_t^i$
**end for**
$W_s[d+1, \ldots] \leftarrow Vector[1, \ldots, 1]$
$W_t[d+1, \ldots] \leftarrow Vector[1, \ldots, 1]$

Let $\tilde{M} \leftarrow W_t \cdot W_s^+$
where $W_s^+$ is the pseudo-inverse of $W_s$.
**function** M'(w)
    **if** $w \in V_t$ **then**
        **return** $v_t$
    **else if** $w \in V_s$ **then**
        **return** $M \cdot v_s$
    **end if**
**end function**
```

This can be solved by calculating the matrix inverse:

$$\left[\begin{array}{c|c} M & b \\ 0 \ldots 0 & 1 \end{array}\right] = \left[\begin{array}{c} W_t \\ 1 \ldots 1 \end{array}\right] \cdot \left[\begin{array}{c} W_s \\ 1 \ldots 1 \end{array}\right]^{-1} \tag{2.12}$$

Given more than $d+1$ aligned points, we can use least-squares regression to solve this system of equations and minimize the Frobenius norm:

$$\begin{gathered} \text{Find } M \text{ to minimize} \\ ||M \cdot W_s - W_t||_F \end{gathered} \tag{2.4}$$

In a similar approach to the exact solution, this can be solved using the pseudo-inverse of the W_s matrix.

$$\left[\begin{array}{c|c} M & b \\ 0\dots0 & 1 \end{array}\right] = \left[\begin{array}{c} W_t \\ 1\dots1 \end{array}\right] \cdot \left[\begin{array}{c} W_s \\ 1\dots1 \end{array}\right]^+ \tag{2.13}$$

where the pseudo-inverse of a matrix A is defined as

$$A^+ = A^T(AA^T)^{-1} \tag{2.14}$$

2.2.1 Results of the Globally Linear Approach

We tested the Globally Linear Approach in the following manner: We varied the number of input vector alignments used for training. In every case, we took 201 or more alignments, since at least 201 are required.

We used Algorithm 2.1 to calculate the transformation matrix M. We then used M to predict vector locations for unseen words (words not used for alignment).

$$\tilde{v_t} = M \cdot v_s \tag{2.15}$$

We then calculated the euclidean and cosine distance between the predicted vectors $\tilde{v_t}$ and the original vectors v_t.

Euclidean distance is calculated by taking the simple L^2 norm

$$d_e(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2.16}$$

while cosine distance is calculated in the following manner:

$$d_c(p,q) = 1 - \frac{p \cdot q}{||p||||q||} \tag{2.17}$$

Before we present the results, to give an estimate of what these results mean, we would like to mention the following statistics:

- **Alignments mapping to GloVe (to M_{11} or M_{22}):** The average euclidean distance between pairs of point is 8.0, the minimum distance is 1.2, and the maximum distance is 18.4. The median **nearest neighbour** euclidean distance from every word is 4.7.

The average cosine distance between pairs of points is 0.98. The median nearest neighbour cosine distance from each word is 0.36.

- **Alignments mapping to Word2Vec (to M_{12} or M_{21}):** The average euclidean distance between pairs of point is 34.9, the minimum distance is 3.7, and the maximum distance is 103.4. The median **nearest neighbour** euclidean distance from every word is 17.9.

  The average cosine distance between pairs of points is 0.97. The median **nearest neighbour** cosine distance from each word is 0.29.

- The cosine distance for two perfectly matched vectors would be 0.0.

The results of the Globally Linear Approach, with varying number of input equations can be found in Table 2.2.

These results show remarkable improvement once additional word vector mappings are taken into consideration. In addition the euclidean and cosine distances are, in general, a decreasing function of the number of mappings, although there are exceptions. This may be because as we increase the number of words (ex. from 1001 to 10,001), the number of occurrences of these words decreases (At rank 4000, the frequency is 40176 occurrences; and 12081 occurrences for rank 10000; while at rank 1001, the frequency is 162020 occurrences), causing the mapping to be slightly inaccurate. This shows that an affine transformation does indeed exist that maps one technique to another (in all cases). This is a very interesting result, especially for the mapping from W_{12} to W_{21}, since these results are for vectors created using different techniques. In addition, the results for the mapping from W_{11} to W_{21}, and the mapping from W_{21} to W_{11} seem to show near-identical results. This may be because of a forward linear mapping exists, a reverse linear mapping will exist too.

We would expect the transformation matrices to be inverses of each other. However, on testing, that does not seem to be the case (for M as the transformation

Number of Aligned Points	Euclidean Distance	Cosine Distance
201	25.39	0.698
401	1.43	0.026
601	1.13	0.017
1001	0.99	0.013
4001	1.19	0.023
10001	1.16	0.022

(a) W_{11} to W_{21} mapping

Number of Aligned Points	Euclidean Distance	Cosine Distance
201	58.47	0.86
401	4.53	0.20
601	3.46	0.141
1001	2.95	0.110
4001	3.83	0.24
10001	3.35	0.19

(b) W_{12} to W_{21} mapping

Number of Aligned Points	Euclidean Distance	Cosine Distance
201	89.03	0.702
401	10.64	0.090
601	7.95	0.054
1001	6.65	0.038
4001	6.81	0.058
10001	6.41	0.052

(c) W_{12} to W_{22} mapping

Number of Aligned Points	Euclidean Distance	Cosine Distance
201	27.19	0.723
401	1.45	0.026
601	1.14	0.017
1001	0.99	0.013
4001	1.21	0.023
10001	1.17	0.022

(d) W_{21} to W_{11} mapping

Table 2.2: Results of the Globally Linear Approach. The aligned points are the most frequent aligned points in $\hat{V}$. Testing is done on a set of words not used for training.

matrix from W_{11} to W_{21} and M' from W_{21} to W_{11}):

$$||M||_F = 13.92 \text{ and } ||M'||_F = 14.01 \text{ while } ||M \cdot M' - I||_F = 3.42 \neq 0 \tag{2.18}$$

2.3 Local Linear Neighbourhood Approach

The algorithm followed in this approach is given in Algorithm 2.2.

Thus, we have a function based on piecewise linear maps A that returns an approximation $\tilde{v}_t^i$ for every word $w^i \in V_s \setminus V_t$.

This algorithm attempts to use the m-dimensional submanifold (representing the locality of our target word) of the vocabulary manifold in $\mathbb{R}^d$ space. We use

France *Rank: 405*		Jesus *Rank: 2363*	
Original	**Predicted**	**Original**	**Predicted**
Spain	Italy	Christ	Christ
Italy	Spain	God	God
Belgium	Belgium	apostles	apostles
Germany	Germany	Satan	Satan
Portugal	Portugal	resurrection	incarnation

reddish *Rank: 16587*		scratched *Rank: 40852*	
Original	**Predicted**	**Original**	**Predicted**
yellowish	yellowish	scraped	scraped
greenish	greenish	cracked	rubbed
brownish	brownish	rubbed	flushed
reddish-brown	reddish-brown	ripped	ripped
bluish	whitish	thrown	shoved

Table 2.3: Nearest Neighbours of vectors generated by the Globally Linear Approach. Here the rank of the words is based on their frequency in the combined corpora. These are the same words used by Collobert and Weston, 2008b.

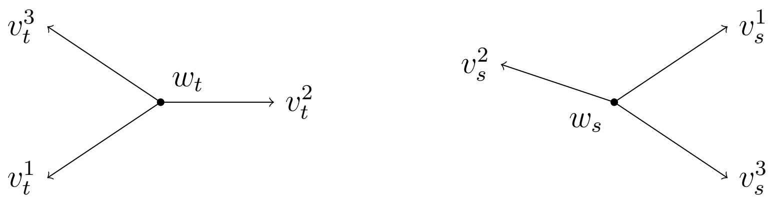


Figure 2.3: We align the m-dimensional submanifold using v_t^i and v_s^i and use these alignments to calculate the mapping of w_s in t-space

the nearest neighbour vectors in the s-space and project each vector to the t-space. Hence, in the local-space around the target word, we have succeeded in aligning the submanifolds using the tangent plane at v_s^i.

The reasoning behind this approach is as follows: To align the submanifolds preserving local equivalence, we require the nearest neighbours of a word in s-space to be preserved in t-space. The simplest way to ensure that the s-space neighbours are as close to their counterparts in t-space is by placing the predicted vector $\tilde{v}$ at the vector average of the neighbours.

Algorithm 2.2 can be used to merge the vectors generated on different corpora.

Algorithm 2.2 Linear Neighbourhood Approach

```
function M'(w)
    if $w^i \in V_t$ then
        return $v_t^i$
    else if $w^i \in V_s$ then
        $v_s \leftarrow v_s^i$
        Find set $N_i^k$ of the $k$ nearest neighbours of $v_s^i$ that are in $\hat{V}$
        i.e.: Find $N_i^k \subset \hat{V}$ s.t. $|N_i^k| = k$ and
          $||i - v_s|| > \max ||j - v_s|| \quad \forall i \in \hat{V} \setminus N_i^k, j \in N_i^k$
        Let $\tilde{v}_t^i \leftarrow \left(\sum_{w^j \in N_i^k} v_t^j\right) / |N_i^k|$
        return $\tilde{v}_t^i$
    end if
end function
```

This can greatly speed up vector-creation, and can prevent memory issues while working on massive corpora. In addition, it can merge vector mappings performing better in different areas, to further improve the semantic information en-captured by the vectors.

A secondary, but in practice significant effect of this approach is smoothing. This approach bases its generated vectors on a larger set of training sentences (comprising of both C_s and C_t corpora). It also reduces the noise in vectors with relatively few occurrences.

2.3.1 Results of Linear Neighbourhood Approach

We tested the Linear Neighbourhood method in the same way as mentioned in Section 2.2.1.

The results of the Linear Neighbourhood Approach, with varying k (number of nearest neighbours) can be found in Table 2.4.

Thus, we can see that, in all cases, our approach gives predictions that are as close or closer than the median nearest neighbours. Our predicted vectors are within the median minimum euclidean distance of the nearest neighbours. They also have a very small cosine distance from the original vectors.

We can also observe that the cosine distance is decreasing as the number of nearest neighbours increases. This is to be expected because the larger the number of nearest

k	Euclidean Distance	Cosine Distance
3	3.49	0.158
6	3.44	0.145
9	3.48	0.144
12	3.52	0.145
15	3.55	0.146

(a) W_{11} to W_{21} mapping

k	Euclidean Distance	Cosine Distance
3	4.60	0.338
6	4.42	0.306
9	4.36	0.294
12	4.34	0.289
15	4.35	0.285

(b) W_{12} to W_{21} mapping

k	Euclidean Distance	Cosine Distance
3	10.51	0.226
6	10.19	0.207
9	10.18	0.203
12	10.27	0.206
15	10.34	0.207

(c) W_{12} to W_{22} mapping

k	Euclidean Distance	Cosine Distance
3	4.10	0.339
6	3.96	0.302
9	3.95	0.290
12	3.97	0.284
15	4.00	0.285

(d) W_{21} to W_{11} mapping

Table 2.4: Results of the Linear Neighbourhood Approach

neighbours, the more accurately we can align the neighbourhood of the vectors. We notice that the technique of creation of these vectors does not matter, since aligning the local submanifolds of word vectors created using different techniques still leads to good results.

However, comparatively, we see that aligning two sets of vectors generated using the same technique perform better than aligning two sets of vectors generated using different techniques. This indicates that there may be a fundamental difference in the underlying theory behind the Mikolov, Chen, et al., 2013 and Pennington et al., 2014 approaches (contrary to Section 3.1 in Pennington et al., 2014).

We can further exemplify these results by looking at the nearest neighbours of the original vector v_t and the predicted vector $\tilde{v}_t = M \cdot v_s$. In table 2.5 we present some examples of such nearest neighbours obtained using this approach.

2.3.2 Word Analogy Dataset

This dataset (Mikolov, Chen, et al., 2013) has been widely used (Mikolov, Chen, et al., 2013; Pennington et al., 2014) for testing the capability of Word Vector Spaces.

France *Rank: 405*		Jesus *Rank: 2363*	
Original	**Predicted**	**Original**	**Predicted**
Spain	Spain	Christ	Christ
Italy	Italy	God	Baptism
Belgium	Belgium	apostles	resurrection
Germany	Portugal	Satan	God
Portugal	Morocco	resurrection	apostles

reddish *Rank: 16587*		scratched *Rank: 40852*	
Original	**Predicted**	**Original**	**Predicted**
yellowish	brownish	scraped	rubbed
greenish	reddish-brown	cracked	scraped
brownish	yellowish	rubbed	crumpled
reddish-brown	pinkish	ripped	peeled
bluish	greyish	thrown	pinched

Table 2.5: Nearest Neighbours of vectors generated by the Linear Neighbourhood Approach. Here the rank of the words is based on their frequency in the combined corpora. These are the same words used by Collobert and Weston, 2008b.

It contains word analogies from various categories. A sample of the categories and the type of questions in the categories is shown below

Semantic Questions

1. **Capitals of Common Countries** – (*Athens : Greece* as *Baghdad : Iraq*)

2. **Currency** – (*Algeria : dinar* as *Europe : euro*)

3. **Family** – (*boy : girl* as *dad : mom*)

Syntactic Questions

1. **Adjective to Adverb** – (*free : freely* as *most : mostly*)

2. **Opposite** – (*sure : unsure* as *logical : illogical*)

3. **Present Participle** – (*fly : flying* as *run : running*)

4. **Past Tense** – (*flying : flew* as *dancing : danced*)

5. **Plural - Nouns** – (*banana : bananas* as *hand : hands*)

6. **Plural - Verbs** – (*eat : eats* as *implement : implements*)

In each task, given three of the four words, the algorithm needs to predict the fourth word from the entire vocabulary available to it. The model needs to correctly identify the relationship between the first two words, and extend it to the third word, to predict the fourth. The prediction can be any word in the vocabulary. We use a vocabulary of size 100,000, giving a random selection approach a 0.001% chance of success in one task.

2.3.3 Testing on the Word-Analogy dataset

We tested the alignment accuracy using the word-analogy dataset (Section 2.3.2). This dataset contains 18,200 questions, each of the form $a : b$ as $c : d$ where we need to predict d given a, b, and c. Only exact correspondence of d with the target word is counted as a match. These tasks are divided into two broad categories: semantic questions and syntactic questions.

Given a word-analogy task like *Berlin* : *Germany* as *Beijing* : _______, we can use word-vectors to predict the blank (*China* in this case) in the following manner:

We find word w with vector v_w closest to $v_{Germany} - v_{Berlin} + v_{Beijing}$ by cosine similarity.

To test the accuracy of the above approaches, we mapped all the vectors in the analogy task using the approaches mentioned above, and used their combination to find the nearest neighbour word to the calculated vector. We present the accuracy of the returned word matching the actual blank.

We report the accuracies of the two subtasks: syntactic (eg. *dance* : *dancing* as *fly* : *flying*) and semantic (eg. *Berlin* : *Germany* as *Beijing* : *China*) independently in Table 2.6.

Model	Syntactic Subtask	Semantic Subtask
Original s Word Vectors	39.0%	47.3%
Original t Word Vectors	38.6%	47.0%
6 - Nearest Neighbour	38.3%	58.8%

(a) W_{11} to W_{21} mapping

Model	Syntactic Subtask	Semantic Subtask
Original s Word Vectors	42.0%	51.1%
Original t Word Vectors	38.6%	47.0%
6 - Nearest Neighbour	33.0%	56.6%

(b) W_{12} to W_{21} mapping

Model	Syntactic Subtask	Semantic Subtask
Original s Word Vectors	42.0%	51.1%
Original t Word Vectors	41.7%	48.4%
6 - Nearest Neighbour	37.0%	57.0%

(c) W_{12} to W_{22} mapping

Model	Syntactic Subtask	Semantic Subtask
Original s Word Vectors	38.6%	47.0%
Original t Word Vectors	39.0%	47.3%
6 - Nearest Neighbour	38.4%	55.6%

(d) W_{21} to W_{11} mapping

Table 2.6: Accuracies on Word-Analogy task.

We note that the results for the original vectors are worse than reported in the literature. This is due to a much smaller training corpus, since we used a subset of the Wikipedia corpus. Additionally, note that, surprisingly, the vectors generated after mapping to the t-space seem to perform better than the original vectors in semantic subtasks. This is an interesting result.

To further verify this, we took a few examples and performed an additional evaluation test:

Testing Dimensionality of Vectors in Analogy Task

If we have a word-analogy task $a : b$ as $c : d$, we expect the vectors v_a, v_b, v_c, and v_d to lie in a plane. This follows directly from our assumption that $v_a - v_b \approx v_c - v_d$.

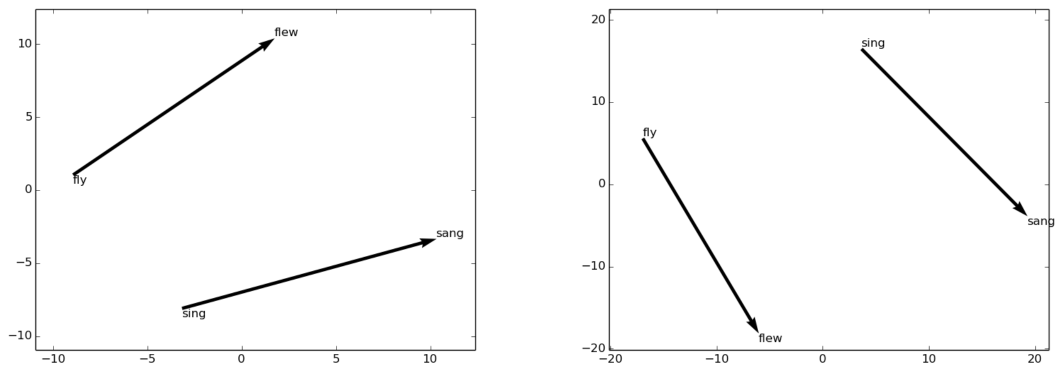
Thus, performing Principal Component Analysis to identify the three principal components. The eigenvalue of the third component will give us information on how accurate our technique is. Ideally, it should be as close to zero as possible. Some examples of this approach are in Table 2.7.

Example	Eigenvalues (Original Vectors)	Eigenvalues (Predicted Vectors)
Bangkok : Thailand Cairo : Egypt	[0.58, 0.35, 0.072]	[0.63, 0.33, 0.044]
Algeria : dinar Mexico : peso	[0.51, 0.41, 0.076]	[0.80, 0.19, 0.008]
father : mother groom : bride	[0.77, 0.14, 0.091]	[0.89, 0.09, 0.019]
known : unknown honest : dishonest	[0.63, 0.29, 0.089]	[0.62, 0.29, 0.088]

Table 2.7: Eigenvalues of Principal Components for some example word-vectors in the word-analogy task. The Predicted vectors are calculated using the Linear Neighbourhood Approach

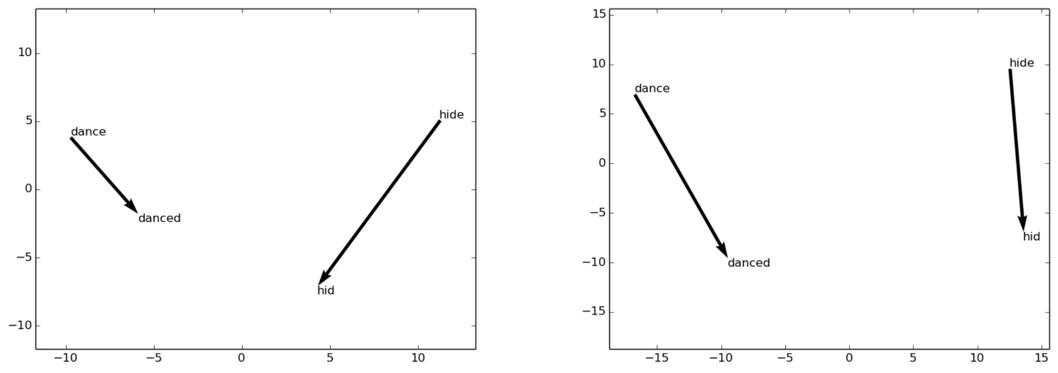
Our intuition says that these newly predicted vectors perform better than the original vectors because our Linear Neighbourhood approach acts as a smoothing function, which is more useful for semantics than syntactics, making certain areas of the vocabulary not captured well by the corpus perform better, and leaving the other areas unaffected. We expect the original vector performance to gradually get closer to the Linear Neighbourhood Approach results as the corpus size increases. Thus further validates our aim of achieving comparable accuracy using smaller corpora and different techniques.

In the results of this chapter, we notice that the Global Alignment approach gives better results than the Local Neighbourhood approach, especially for the *Corpus 2* to *Corpus 1* mapping. The Local Neighbourhood approach gives a good smoothing effect, and has interesting results in the Word Analogy task. We can infer that a global alignment probably exists, although it is not entirely accurate in every local



Predicted Vectors Original Vectors

(a) Vectors for the relation *fly : flew = sing : sang*.

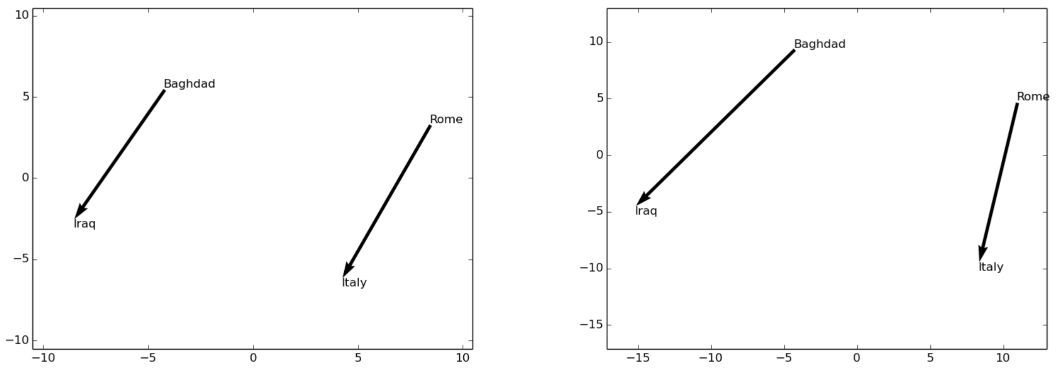


Predicted Vectors Original Vectors

(b) Vectors for the relation *dance : danced = hide : hid*.

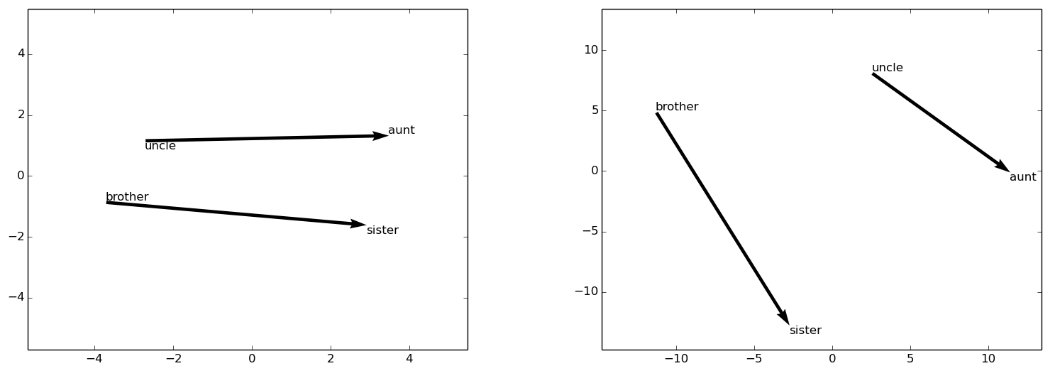
Figure 2.4: We present vectors for some examples in the word-analogy syntactic subtask. Well-calculated vectors will form a parallelogram, which can accurately be reduced to 2-dimensions.

neighbourhood.



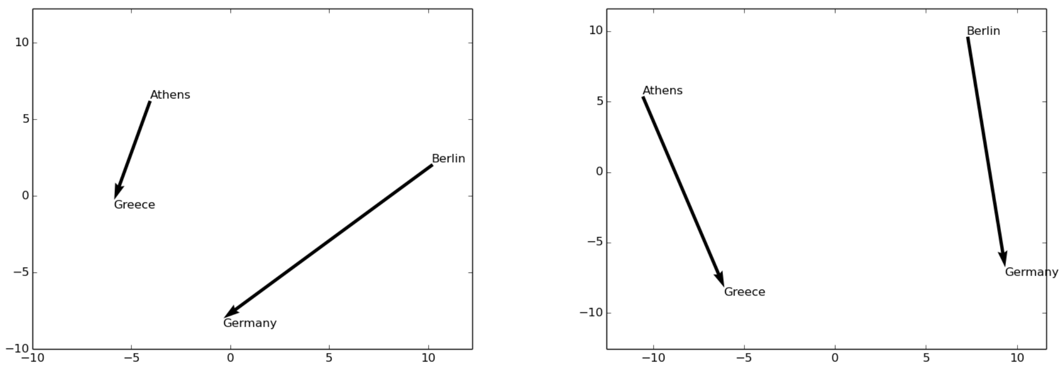
Predicted Vectors Original Vectors

(a) Vectors for the relation *Baghdad : Iraq = Rome : Italy*. We can see that our predicted vectors form a more accurate parallelogram



Predicted Vectors Original Vectors

(b) Vectors for the relation *brother : sister = uncle : aunt*.



Predicted Vectors Original Vectors

(c) Vectors for the relation *Athens : Greece = Berlin : Germany*. This particular example doesn't perform as well as the original vectors.

Figure 2.5: We present vectors for some examples in the word-analogy semantic subtask. Well-calculated vectors will form a parallelogram, which can accurately be reduced to 2-dimensions.

Chapter 3

Paragraph Vectors for GloVe

3.1 Introduction

3.1.1 Paragraph Vectors

There have been many approaches that attempt to model sentences and documents in the form of fixed-length vectors. Such vectors are very useful in machine learning algorithms like K-means (for document clustering), and Support Vector Machines (for sentiment analysis). We can also use these vectors to calculate relevance rankings of documents, and to compare document similarity (Turney, Pantel, et al., 2010).

The simplest and perhaps the most common approach for representing paragraphs as vectors is the bag-of-words approach (Z. S. Harris, 1954). This approach has been modified using the Term Frequency - Inverse Document Frequency approach, to get better results, but such approaches greatly simplify the paragraphs they represent. In addition, the dimensionality of any such approach is very large.

One of the more recent works related to word vectors is the paper, Le and Mikolov, 2014. The *Paragraph Vector* approach fixes many of the problems faced by the earlier models like Mitchell and Lapata, 2010 and Collobert and Weston, 2008b. It can represent paragraphs of variable length as a fixed-length vector, and is purely unsupervised. Furthermore, it uses local context windows to train a paragraph vector.

The Paragraph Vector approach modifies the existing word vector training algorithm, Word2Vec, discussed in Mikolov, Chen, et al., 2013, and extends it naturally to train paragraph representations.

3.1.2 Word Vectors using GloVe

The GloVe algorithm, Pennington et al., 2014, deals with a new technique for learning vector space representations for words. This builds on previous works to use both standard sets of techniques: global matrix factorization and local context windows. This algorithm uses the word co-occurrence matrix, trained using least-squares regression, to generate word vectors which are able to represent meaning.

The semantic strength of the GloVe algorithm hinges on the ratio of co-occurrence probabilities. Given related target words like *ice* and *steam*, and a context word k, it is the ratio of probabilities, $P(k|ice)/P(k|steam)$ that can detect the relation between k and either of the target words. For example, if $k = banana$, we will not see the ratio change appreciably from 1. However, for a word like $k = solid$, we will see the ratio change from 1 (in this case $P(solid|ice)/P(solid|steam) \gg 1$). This allows us to look for vector differences correlated with probability ratios:

$$F(w_i - w_j, \tilde{w}_k) = \frac{P(k|i)}{P(k|j)} = \frac{P_{ik}}{P_{jk}} \tag{3.1}$$

This allows the vector differences between word vectors in the mapped $\mathbb{R}^d$-space to represent the similarity between these words.

3.2 The GloVe algorithm for word vectors

The current GloVe implementation (Pennington et al., 2014) uses the following approach to generate word vectors given a corpus.

Let us denote the matrix of word-word co-occurrence counts with $\tilde{C}$, whose entries $\tilde{C}_{ij}$ represent the number of times the word j occurs in the context of the word i. We also define $\tilde{C}_i = \sum_k \tilde{C}_{ik}$ as the number of times any word appears in the context of word i. Now, clearly,

$$P_{ij} = P(j|i) = \frac{\tilde{C}_{ij}}{\tilde{C}_i} \tag{3.2}$$

will represent the probability that word j occurs in the context of the word i.

Let $w_i \in \mathbb{R}^d$ be word vectors, and $\tilde{w}_k \in \mathbb{R}^d$ be separate context word vectors defined as per 3.1.2 and Pennington et al., 2014.

Pennington et al., 2014 goes ahead to infer the following cost function:

$$J = \sum_{i,j=1}^{V} f\left(\tilde{C}_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log \tilde{C}_{ij}\right) \tag{3.3}$$

where b_i and $\tilde{b}_j$ are bias terms depending on i and j respectively, and f is a weighing function to prevent weighing all co-occurrence equally. Performing least-squares regression using (3.3) as the cost function, we can calculate the word vectors w_i and the reference word vectors $\tilde{w}_j$.

3.3 Extending to Paragraph Vectors

Given a set of documents/paragraphs S, we want to learn paragraph vectors such that each paragraph is mapped to the same $\mathbb{R}^d$ space as the words of vocabulary V used in the corpus.

We propose a modification the the matrix $\tilde{C}$ (as defined above) to represent paragraph 'co-occurrence' counts as well as the word-co-occurrence counts.

In order to do so, we will modify the original $\tilde{C}_{|V|\times|V|}$ matrix by creating a new matrix, $C_{(|V|+|S|)\times(|V|+|S|)}$ with entries C_{ij} corresponding to word-word co-occurrence counts, C_{iA} and C_{Ai} corresponding to word-paragraph co-occurrence counts, and C_{AB} corresponding to paragraph-paragraph co-occurrence counts, for $A, B \in S$ and $i, j \in V$.

This will give us the following cost-function

$$J = \sum_{i,j=1}^{V \cup S} f\left(C_{ij}\right) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log C_{ij}\right) \tag{3.4}$$

We leave $\tilde{C}_{ij}$ untouched. i.e. $C_{ij} = \tilde{C}_{ij}$. Now, we need to populate the remaining values of the matrix, using the values that we have.

In general, given a paragraph $A = a_1 a_2 \dots a_n$, $(\forall a_i \in V)$ we wish to write an expression for C_{iA} such that

$$\begin{aligned} C_{iA} = F_{iA}(C_1, \dots, C_{|V|}, \\ C_{12}, \dots, C_{|V|-1,|V|}) \end{aligned} \tag{3.5}$$

Similarly, given paragraphs $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_m$, $(\forall a_i, b_i \in V)$ we wish to write an expression for C_{AB} such that

$$\begin{aligned} C_{AB} = F_{AB}(C_1, \dots, C_{|V|}, \\ C_{12}, \dots, C_{|V|-1,|V|}) \end{aligned} \tag{3.6}$$

Note that this means we will not be able to accurately deal with paragraphs that contain Out Of Vocabulary (OOV) words, and there is no existing technique that is able to do so at the moment.

$$\begin{bmatrix} \quad \tilde{C}_{ij} \quad \end{bmatrix} \Rightarrow \left[\begin{array}{c|c} \tilde{C}_{ij} & C_{iA} \\ \hline C_{Aj} & C_{AB} \end{array}\right]$$

Figure 3.1: Modifying the GloVe co-occurrence matrix

3.4 Equations

The choice of F_{iA} and F_{AB} depend on us. Any equation that can approximate the co-occurrence counts will be satisfactory. Hence, we have quite a bit of freedom in this choice. We have tried the following equations for F_{iA}:

$$C_{iA} = 0 \tag{3.7}$$

$$C_{iA} = C_i \times \prod_{j=1}^{n} C_{a_j} \tag{3.8}$$

$$C_{iA} = C_i \times \prod_{j=1}^{n} C_{ia_j} \tag{3.9}$$

$$C_{iA} = C_i \times C_{ia_1} \times \prod_{j=1}^{n-1} \frac{C_{a_j a_{j+1}}}{C_{a_j}} \tag{3.10}$$

$$c_{iA} = C_i \times \sum_{j=1}^{n} \left(C_{ia_j} \times \left(1 - \frac{C_{a_j}}{\sum_k^V C_k} \right) \right) \times \frac{1}{n} \tag{3.11}$$

$$C_{iA} = C_i \times \frac{\sum_{j=1}^{n} C_{ia_j}}{n} \tag{3.12}$$

and we have tried the following equations for F_{AB}:

$$C_{AB} = 0 \tag{3.13}$$

$$C_{AB} = \sum_{w}^{A \cap B} C_w \tag{3.14}$$

$$C_{AB} = |\{a_1, a_2, \ldots, a_n\} \cap \{b_1, b_2, \ldots, b_m\}| \times \frac{\sum_k^V C_k}{|V|} \tag{3.15}$$

The reasoning for these equations is as follows:

From (3.2) we know that

$$P_{ij} = P(j|i) = \frac{C_{ij}}{C_i} \tag{3.16}$$

Extending this equation to paragraphs, the probability of paragraph A occurring in the context of the word i will be

$$P_{iA} = P(A|i) = \frac{C_{iA}}{C_i} \tag{3.17}$$

Since we wish to derive an expression for C_{iA},

$$C_{iA} = C_i \times P(A|i) \tag{3.18}$$

Now, we can decide on an expression representative of $P(A|i)$, the probability of a paragraph A occurring within the context of a word i, using the information available to us in (3.5).

By the chain rule, we can represent $P(A|i) = P(a_1 a_2 \dots a_n|i)$ as

$$\begin{aligned} P(A|i) &= P(a_1 a_2 \dots a_n|i) \\ &\approx P(a_1|i) \cdot P(a_2|a_1, i) \cdot ... \cdot P(a_n|a_{n-1}, \dots, a_2, a_1, i) \end{aligned} \tag{3.19}$$

Here, we can make reasonable unigram ($P(i|j, k, \dots) \approx P(i)$) and bigram ($P(i|j, k, \dots) \approx P(i|j)$) assumptions to get the above equations.

3.5 Baseline comparison algorithms

We tested our paragraph vector approach on Dataset 3.6.3. In addition, we tested the following approaches:

3.5.1 Bag of Words

We represent each paragraph as a vector with length $|V|$ with values representing the frequency of words. This approach disregards grammar and word order.

3.5.2 Paragraph Vectors - Distributed Memory Model

This approach extends the Word2Vec, Mikolov, Chen, et al., 2013, algorithm to paragraph vectors (Le and Mikolov, 2014). We will be using the Distributed Memory Model of Paragraph Vectors (PV-DM) from the article. We pre-trained the word-vectors using the entire Wikipedia Corpus 3.6.2, and used these vectors to get

d-dimensional vectors for the paragraphs.

3.5.3 Word Vector Averaging

We took the average of the word vectors of all words in the paragraph. We used vectors trained using the GloVe (Pennington et al., 2014) algorithm.

3.5.4 Word Vector Weighted Averaging

We followed the approach in Singh and Mukerjee, 2015. We took the average of the word vectors of all words in the paragraph, weighed by the frequency of the word in the initial corpus:

$$v_{sent} = \sum_{w}^{V} v_w \times \left(1 - \frac{U_w}{\sum_i U_i}\right) \tag{3.20}$$

where U_i is the unigram occurrence frequency of the word i in the corpus.

We used vectors trained using the GloVe (Pennington et al., 2014) algorithm.

3.5.5 Clustering based on Chinese Restaurant Process

We used both variants of the approach mentioned at Miao, 2014. This approach treats the paragraph as a bag of words. It then clusters the words in the paragraph so as to get all the words relevant to the paragraph in one cluster. We cluster the words using the Chinese Remainder Process (CRP) stochastic approach, since this approach does not have a fixed number of clusters, and it works well with cosine distance.

In these variants, we keep track of the cluster vector averages, and the words in each cluster. In every step, based on the hyperparameter $r = 1/(1 + n_{clusters})$, we take the next word and choose to either create a new cluster, or put the word into an existing cluster.

Variation 3.1 leads to more, smaller clusters, while 3.2 gives fewer, larger clusters.

Miao, 2014 does not give a way to select the most relevant cluster. We have attempted to devise our own distance metrics, suited to the vector list returned by these algorithms.

Algorithm 3.1 Clustering - Variant 1

```
Let $C_v \leftarrow Array[]$    (This will store the vectors of the clusters)
Let $C_c \leftarrow Array[]$    (This will store the words in the clusters)
Let $n_{clusters} \leftarrow 0$
for $w \in A$ do
    if $n_{clusters} = 0$ then
        Add $Array[w]$ to $C_c$
        Add $v_w$ to $C_v$
        $n_{clusters} \leftarrow n_{clusters} + 1$
        continue
    end if
    Find vector $C_v[i]$ within $C_v$ with maximum cosine similarity to $v_w$
    Pick a random number $p$ uniformly from $[0, 1)$
    if $p < {}^{1}/_{(1+n_{clusters})}$ then
        Add $Array[w]$ to $C_c$
        Add $v_w$ to $C_v$
        $n_{clusters} \leftarrow n_{clusters} + 1$
    else
        Add $w$ to $C_c[i]$
        $C_v[i] \leftarrow C_v[i] + v_w$
    end if
end for
return $C_v$    (This returns a list of cluster vectors)
```

Our algorithm returns a list of cluster vectors for an input paragraph. We need to be able to calculate the distance between two paragraphs. We have used two methods to compare the list of vectors l_1, l_2 created for two paragraphs and compute this distance:

1. **Method 1:** We take every combination of elements in l_1 and l_2, and return the minimum cosine distance as the predicted distance.

$$distance(l_1, l_2) = \min distance_{cosine}(v_1, v_2) \quad \forall v_1 \in l_1,\ v_2 \in l_2 \tag{3.21}$$

2. **Method 2:** We convert the list returned by the algorithms into a vector. We calculate the average unigram occourrence count for every cluster in C_c. We return the vector corresponding to the lowest average count. The distance

Algorithm 3.2 Clustering - Variant 2

```
Let $C_v \leftarrow Array[]$    (This will store the vectors of the clusters)
Let $C_c \leftarrow Array[]$    (This will store the words in the clusters)
Let $n_{clusters} \leftarrow 0$
for $w \in S$ do
    if $n_{clusters} = 0$ then
        Add $Array[w]$ to $C_c$
        Add $v_w$ to $C_v$
        $n_{clusters} \leftarrow n_{clusters} + 1$
        continue
    end if
    Find vector $C_v[i]$ within $C_v$ with maximum cosine similarity to $v_w$
       Let this cosine similarity be $sim(v, C_v)$
    Pick a random number $p$ uniformly from $[0, 1)$
    if $sim(v, C_v) > {}^1/_{(1+n_{clusters})}$ then
        Add $w$ to $C_c[i]$
        $C_v[i] \leftarrow C_v[i] + v_w$
    else if $p < {}^1/_{(1+n_{clusters})}$ then
        Add $Array[w]$ to $C_c$
        Add $v_w$ to $C_v$
        $n_{clusters} \leftarrow n_{clusters} + 1$
    else
        Add $w$ to $C_c[i]$
        $C_v[i] \leftarrow C_v[i] + v_w$
    end if
end for
return $C_v$    (This returns a list of cluster vectors)
```

between these returned vectors is calculated using the standard cosine distance function.

3.6 Datasets and Corpora

3.6.1 English Wikipedia Corpus

We have used a cleaned and tokenized form of the English Wikipedia corpus available through the Al-Rfou, Perozzi, and Skiena, 2013 paper at Al-Rfou, 2013. This gives us a very large dataset in the English Language.

3.6.2 Hindi Wikipedia Corpus

We have used a cleaned and tokenized form of the Hindi Wikipedia corpus available through Al-Rfou et al., 2013 and Al-Rfou, 2013. This gives us a very large corpus in the Hindi Language.

We have merged this corpus with the one available at CFILT, 2010. This has given us a larger corpus in the Hindi Language.

3.6.3 Wikipedia Paragraph Dataset

We have used the corpora mentioned in Sections 3.6.1 and 3.6.2 to help us create a dataset we can use to test our approach on.

We selected around 35,000 - 40,000 Wikipedia articles from the respective corpora. We then selected the top three paragraphs from each article longer than three paragraphs, and for each paragraph having a length of larger than 5 words.

Dataset	Number of Documents	Average Document Length	Average Number of Words per Document
English Wikipedia	106,497 (39693 articles)	119.75 characters	22.28
Hindi Wikipedia	119,079 (35499 articles)	100.45 characters	20.97

Table 3.1: Statistics of the Wikipedia Paragraph datasets

We used the above paragraphs to create our test. For each query, we create a triplet of paragraphs: two paragraphs are selected from the same Wikipedia article, whereas the third paragraph is randomly selected from the rest of the collection. Our goal is to identify the paragraph not belonging to the Wikipedia article.

A baseline approach should give an accuracy of 33%.

3.6.4 SEMEVAL 2014 - Task 3

SemEval 2014 contains a cross-level semantic similarity task described in David Jurgens and Navigli, 2014; Jurgens, Pilehvar, and Navigli, 2014, which contains a subtask specifically dealing with phrase to word similarity.

We are provided with a training dataset and a test dataset. Each dataset consists of 1000 triplets with one word, one phrase, and a score defined on the five-point Likert scale as follows:

- **4, Very Similar**
- **3, Somewhat Similar**
- **2, Somewhat Related but not Similar**
- **1, Slightly Related**
- **0, Unrelated**

Evaluation is done by calculating the Pearson correlation between the deduced similarity of the word and the phrase based on the above scoring system, and the gold-standard scores. Hence, a Pearson correlation of 1.0 implies the deduced similarities match perfectly with the actual similarities.

The current state-of-the-art (Jurgens et al., 2014) for this subtask is a Pearson correlation of 0.457. The approach used language-specific resources such as POS tags, WordNet, and Lemmatization (Kashyap et al., 2014).

3.7 Testing and Results

3.7.1 Wikipedia Datasets

We used the Hindi Wikipedia Dataset (3.6.3) for creating our tests. For each query, we create a triplet of paragraphs: two paragraphs are randomly selected from the same Wikipedia article, whereas the third paragraph is randomly selected from the rest of the collection. Our goal is to identify the paragraph not belonging to the Wikipedia article. We use the following approach to detect the odd-one-out paragraph: We compute the paragraph vectors for all three paragraphs, and return the paragraph furthest from the other two.

The following tests have been performed using 5000 random queries following the procedure mentioned above.

Accuracy	(3.13)	(3.14)	(3.15)
(3.7)	33%	44%	49%
(3.8)	32%	44%	49%
(3.9)	33%	44%	50%
(3.10)	33%	43%	45%
(3.11)	36%	45%	53%
(3.12)	35%	43%	52%

Table 3.2: Comparison on different function combinations for the Hindi Wikipedia task in the GloVe Paragraph Vectors

As we can see in Table 3.2, in most cases the F_{AB} function seems to make a much larger difference than the F_{iA} function. However, in some cases, F_{iA} changes the performance of F_{AB} completely. This is probably because we are comparing sentences with other sentences in this dataset.

In these tasks, the combination of (3.11) and (3.15) seems to work the best.

$$C_{iA} = C_i \times \sum_{j=1}^{n} \left(C_{ia_j} \times \left(1 - \frac{C_{a_j}}{\sum_k^V C_k} \right) \right) \times \frac{1}{n} \tag{3.11}$$

$$C_{AB} = |\{a_1, a_2, \ldots, a_n\} \cap \{b_1, b_2, \ldots, b_m\}| \times \frac{\sum_k^V C_k}{|V|} \tag{3.15}$$

We will now provide a comparison of our approach (using (3.11) and (3.15)) with the other approaches mentioned in Section 3.5.

Approach	Accuracy
Baseline	33%
BoW	60%
PV-DM	61%
Averaging	55%
Weighted Averaging	64%
CRP - Variant 1	51%
CRP - Variant 2	50%
CRP - Variant 1 (IDF Selection)	46%
CRP - Variant 2 (IDF Selection)	45%
GloVe Paragraph	53%

Table 3.3: Comparison of different approaches on Hindi Wikipedia task

Approach	Accuracy
Baseline	33%
BoW	49%
PV-DM	57%
Averaging	65%
Weighted Averaging	68%
CRP - Variant 1	53%
CRP - Variant 2	51%
CRP - Variant 1 (IDF Selection)	50%
CRP - Variant 2 (IDF Selection)	51%
GloVe Paragraph	47%

Table 3.4: Comparison of different approaches on English Wikipedia task

3.7.2 SemEval Dataset

We also tested our approach on the SemEval 2014 Task 3 dataset 3.6.4, where we look at word-phrase comparisons. In this case, modifying the F_{AB} functions seemed to have little effect, while the F_{iA} functions changed the results considerably.

For the following approaches (except the state-of-the-art approach), we have calculated the cosine similarity between the test word and test phrase, and multiplied it by 4. This approach gives a value of 4.0 for perfectly aligned vectors, and a value of 0.0 for completely misaligned vectors. This is a naive approach, limited by the fact that no pre-training was done on these methods.

Approach	Pearson Correlation
Baseline	0.000
Meerkat (State-of-the-art)	0.457
PV-DM	0.103
Averaging	0.053
Weighted Averaging	0.063
GloVe Paragraph	0.075

Table 3.5: Comparison of different approaches on SemEval 2014 subtask

As we can see, our approach outperforms the naive averaging and weighted averaging approaches, and its performance is close to the PV-DM approach, which is the current unsupervised state-of-the-art.

Chapter 4

Conclusions

Our implementation is open source and freely available at

https://github.com/Shashwat986/thesis

4.1 Summary

In this work, we discuss the problem of aligning multiple mappings from the vocabulary space to a vector space. We compare two different approaches, one based on local context alignment, and another based on computing the global alignment. We present the results of our approach, which improve over the existing word vectors due to the smoothing inherent in the technique, and demonstrate its viability with regards merging multiple corpora and vectors. We test our approach on the word-analogy task, and report interesting results.

Further, we have provided a very generalizable implementation for extending the word vector approach used by GloVe to calculate paragraph vectors of an unseen paragraph. We test the created word vectors using two datasets we have created using Wikipedia, and the SEMEVAL 2014 - Task 3, and the results seem promising.

4.2 Scope for further work

In the long term, we hope the community will find these approaches useful and will extend the work carried out in this thesis.

The approach used to generalize the GloVe algorithm currently uses trial-and-error to select the best combination of functions for word-document and document-document co-occurrence counts. A better, stable function is still required in the approach.

The results of the word-phrase similarity in Dataset 3.6.4 can be further improved by thresholding the cosine/euclidean similarity using the training set.

The matrix alignment approach using simultaneous equations needs to be tested further with regards its applicability to parallel corpora.

The matrix alignment approach using nearest neighbours seems to be giving better results than the original vectors. This needs to be further explored.

References

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics* (pp. 16–23). Association for Computational Linguistics.

Korhonen, A., Krymolowski, Y., & Briscoe, T. (2006). A large subcategorization lexicon for natural language processing applications. In *Proceedings of lrec* (Vol. 6).

Atwell, E., Demetriou, G., Hughes, J., Schiffrin, A., Souter, C., & Wilcock, S. (2000). A comparative evaluation of modern english corpus grammatical annotation schemes. *ICAME Journal: International Computer Archive of Modern and Medieval English Journal*, *24*, 7–23.

DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, *14*(1), 31–39.

Gallant, S. I. (1988). Connectionist expert systems. *Communications of the ACM*, *31*(2), 152–169.

Schütze, H. (1993). Word space. In *Advances in neural information processing systems 5*. Citeseer.

Morin, F. & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics* (pp. 246–252). Citeseer.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *JAsIs*, *41*(6), 391–407.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*, 3.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: global vectors for word representation. In *Proceedings of emnlp*.

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384–394). Association for Computational Linguistics.

Collobert, R. & Weston, J. (2008a). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). ACM.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. Retrieved from http://arxiv.org/abs/1301.3781

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th an-*

nual meeting of the association for computational linguistics: human language technologies-volume 1 (pp. 142–150). Association for Computational Linguistics.

Schütze, H. (1998). Automatic word sense discrimination. *Computational linguistics*, *24*(1), 97–123.

Collobert, R. & Weston, J. (2008b). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). ACM.

Le, Q. V. & Mikolov, T. (2014). Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053.*

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, *3*, 1137–1155.

Harris, Z. S. (1954). Distributional structure. *Word.*

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., & Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 281–285). ACM.

Turney, P. D., Pantel, P. et al. (2010). From frequency to meaning: vector space models of semantics. *Journal of artificial intelligence research*, *37*(1), 141–188.

Mitchell, J. & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive science*, *34*(8), 1388–1429.

Singh, P. & Mukerjee, A. (2015). Decompositional semantics for document embedding. Retrieved July 16, 2015, from http://www.cse.iitk.ac.in/users/spranjal/thesis/

Miao, Y. (2014). From word2vec to doc2vec: an approach driven by chinese restaurant process. Retrieved June 11, 2015, from http://eng.kifi.com/from-word2vec-to-doc2vec-an-approach-driven-by-chinese-restaurant-process/

Al-Rfou, R., Perozzi, B., & Skiena, S. (2013, August). Polyglot: distributed word representations for multilingual nlp. In *Proceedings of the seventeenth conference on computational natural language learning* (pp. 183–192). Sofia, Bulgaria: Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W13-3520

Al-Rfou, R. (2013). Polyglot - download wikipedia text dumps. Retrieved June 11, 2015, from https://sites.google.com/site/rmyeid/projects/polyglot#TOC-Download-Wikipedia-Text-Dumps

CFILT. (2010). Resource center for indian language technology. Retrieved June 11, 2015, from http://www.cfilt.iitb.ac.in/Downloads.html

David Jurgens, M. T. P. & Navigli, R. (2014). Cross-level semantic similarity - semeval 2014 task 3. Retrieved June 11, 2015, from http://alt.qcri.org/semeval2014/task3/

Jurgens, D., Pilehvar, M. T., & Navigli, R. (2014). Semeval-2014 task 3: cross-level semantic similarity. *SemEval 2014*, 17.

Kashyap, A., Han, L., Yus, R., Sleeman, J., Satyapanich, T., Gandhi, S., & Finin, T. (2014). Meerkat mafia: multilingual and cross-level semantic textual similarity systems. In *Proceedings of the 8th international workshop on semantic evaluation* (pp. 416–423). Association for Computational Linguistics.

Harris, Z. (1954). Distributional structure. *Word*, *10*(2/3), 146–62.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).