LEARNING EMBEDDINGS FOR TEXT AND IMAGES FROM STRUCTURE OF THE DATA

by

Behrouz Haji Soleimani

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
June 2019

Dedicated to my beloved parents,

for their endless love, support, encouragement and sacrifices.

Table of Contents

List of Tables

List of Figures

Abstract

In the big data era, most of the data generated every day are high dimensional such as text and image data. Learning compact representations from the input data can help in dealing with the high dimensionality and visualization of the data. These representations can be learned to map the input space into a latent space in which, e.g., complex relationships present in the data become more explicit, while preserving the structure and geometrical properties of the data. In the case of discrete input features such as text data, embedding algorithms try to learn a vector space representation of inputs while preserving certain aspects of similarity or relatedness. Training these embeddings is time-consuming and requires a large corpus. This makes it difficult to train task-specific embeddings due to insufficient data in the downstream tasks which has given rise to pre-trained models and embeddings.

In this thesis, we first study dimensionality reduction methods and propose Unit Ball Embedding (UBE), a spherical representation learning method that learns the structure of manifolds and maximizes their separability, which makes it suitable for both clustering and visualization. We then generalize the algorithm and apply it to learn word embeddings, taking into account the contexts of individual words. Our word embedding solution learns a better distribution of words in the latent space by pushing unrelated words away from each other. We investigate and address the naive procedure of negative sampling in Word2Vec and exploit the full capacity of negative examples. We also analyze frequency-based and spectral word embedding methods and show how the idea of negative context can be used in these types of algorithms. We propose EigenWord, a spectral word embedding with an intuitive formulation that makes use of negative examples, and theoretically show that it has an optimal closed-form solution. Finally, we tackle the Word Sense Disambiguation (WSD) problem and propose a multi-sense embedding algorithm based on EigenWord. We evaluated the proposed algorithms using both intrinsic and extrinsic evaluation of embeddings. Extensive experiments on word similarity datasets, emotion recognition from tweets, and sentiment analysis show the effectiveness of our proposed methods.

List of Abbreviations

Notation	**Description**
AIT	Affect In Tweets
API	Application Program Interface
AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformers
biLM	Bidirectional Language Model
BOW	Bag Of Words
CBOW	Continuous Bag-Of-Words
CNN	Convolutional Neural Network
CoVe	Context Vectors
DISC	Distant Supervision Corpus
EI-Reg	Emotion Intensity Recognition
ELMo	Embeddings from Language Models
EM	Expectation Maximization
FA	Factor Analysis
GNB	Gaussian Naive Bayes
GPLVM	Gaussian Process Latent Variable Model
HTML	HyperText Markup Language
ICA	Independent Component Analysis

Notation	Description
IMDB	Internet Movie DataBase
KL	Kullback-Leibler
KLD	Kullback-Leibler Divergence
KNN	K Nearest Neighbor
KUBWE	Kernelized Unit Ball Word Embedding
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Latent Dirichlet Allocation
LDA	Linear Discriminant Analysis
LLE	Locally Linear Embedding
LMNN	Large Margin Nearest Neighbor
LoCH	Local Convex Hull
LPP	Locality Preserving Projection
LS	Least Squares
LSTM	Long Short-Term Memory
LTSA	Local Tangent Space Alignment
MCML	Maximally Collapsing Metric Learning
MDS	Multi-Dimensional Scaling
MNB	Multinomial Naive Bayes
MSE	Mean Squared Error
MVU	Maximum Variance Unfolding
MWE	Multi-Word Expression
NB	Naive Bayes
NCA	Neighborhood Component Analysis
NLI	Natural Language Inference

Notation	Description
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NMF	Non-negative Matrix Factorization
NMI	Normalized Mutual Information
NNDSVD	Non-Negative Double Singular Value Decomposition
NPE	Neighborhood Preserving Embedding
OOV	Out-Of-Vocabulary
PCA	Principal Components Analysis
PMI	Pointwise Mutual Information
PPCA	Probabilistic Principal Components Analysis
PPMI	Positive Pointwise Mutual Information
PSD	Positive Semi-Definite
RF	Random Forest
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SGNS	Skip-Gram with Negative Sampling
SNE	Stochastic Neighbor Embedding
SPE	Structure Preserving Embedding
SPPMI	Shifted Positive Pointwise Mutual Information
SVD	Singular Value Decomposition
SVD-NS	Singular Value Decomposition with Negative Sampling
SVM	Support Vector Machine

Notation	Description
SVR	Support Vector Regression
t-SNE	t-distributed Stochastic Neighbor Embedding
TF-IDF	Term Frequency Inverse Document Frequency
UBE	Unit Ball Embedding
UICA	Undercomplete Independent Component Analysis
ULMFiT	Universal Language Model Fine-Tuning
URL	Uniform Resource Locator
VP-Tree	Vantage Point Tree
WNDCG	Weighted Normalized Discounted Cumulative Gain
WSD	Word Sense Disambiguation
WSI	Word Sense Induction
XML	Extensible Markup Language

Acknowledgements

This dissertation would not have been possible without the support of many people. First and foremost, I would like to express my sincere gratitude to my advisor, Professor Stan Matwin, for his continuous support throughout my PhD. He is an excellent mentor who provided me with the perfect balance of guidance and freedom, and gave me the opportunity to pursue my interest in Machine Learning, Deep Learning, and Natural Language Processing. I am forever grateful for his support and encouragement and for the lessons I learned from him.

I would also like to thank my thesis committee: Dr. Vlado Keselj, Dr. Daniel Silver, Dr. Sageev Oore, and Dr. Robert Beiko who have generously given their time and expertise, and provided me with constructive feedback to better my work. Special thanks to Dr. Jackie Cheung from McGill University, for kindly accepting to be my thesis external examiner, and for his insightful questions and suggestions that certainly improved my thesis.

My deep appreciation to my colleagues at the Institute for Big Data Analytics and MALNIS research group for all the brainstormings, inspiring discussions, and for all the great times that we have had in the last couple of years. Special thanks to Dr. Armin Sajadi for helping me to ramp up in the embedding space and NLP and for providing me with the clean text of Wikipedia, Xiang Jiang for the great discussions we had and for letting me use his GPUs for my thesis experiments, Dr. Erico Neves de Souza for his valuable feedback on my work and papers, Casey Hilliard and Dr. Amilcar Soares for their help and technical support with respect to BigData servers. It is due to the friendly and supportive environment in the Institute for Big Data Analytics, and the overall atmosphere of the Faculty of Computer Science at Dalhousie that I was lucky enough to find so many great people to work with.

Last but not the least, I would like to thank my family, especially my parents, who did everything in their power for me to succeed. Thanks for always being there for me. Anything good that has come to my life has been because of your support, love and encouragement. I will never be able to thank you enough.

Chapter 1

Introduction

Machine learning is a growing field of research which has a wide spectrum of applications ranging through text mining, computer vision, speech recognition, spam detection, weather forecast, bioinformatics, etc. Most of these applications of machine learning deal with high-dimensional data. For instance, in a collection of text documents, there are tens of thousands of unique words which are usually used as features. In an image collection, each picture consists of tens of thousands of pixels. Dealing with high-dimensional data brings on difficult challenges. Recent progress in deep learning resulted in breakthroughs in the performance of learning algorithms working on image and text data. Even though deep learning algorithms have been successful in computer vision and Natural Language Processing (NLP), they require large amounts of data to be trained on. This has led to an increasing interest in learning representations from high-dimensional and unstructured data.

One way to deal with the high dimensionality is to learn a low-dimensional compact representation of input data. These mappings typically take into account some notion of similarity in the input space and try to preserve the structure and geometrical properties of the data during the mapping. These representations can often make the complex relationships present in the data more explicit. For instance, if the data consists of low-dimensional manifolds embedded in a high-dimensional space, mappings can unveil the structure of manifolds and map them to a low-dimensional space by unfolding the twisted shapes [93, 165]. Such manifolds can be formed due to slight changes in a certain aspect of input datapoints. For instance, slight changes in lighting conditions or slight rotations in images from the same object can form a manifold. In this thesis, we review and study dimensionality reduction and manifold learning algorithms and propose a spherical representation learning algorithm that learns the structure of manifolds. One of the key distinguishing factors of our algorithm is that it maximizes the separability of manifolds by using a strong repulsion force in the optimization.

Dealing with high-dimensional data becomes even more challenging in NLP where we have to work with discrete features (i.e. words). The traditional ways of representing documents with Bag Of Words (BOW) [133] or bag of n-grams [50] have strong limitations. The bag of words model does not consider the word order in the sentences. Bag of n-grams alleviates this to some extent by considering phrases, but it still does not capture semantic aspects of the documents in a corpus. Recently, word embedding methods consider the context of words and embed words in a vector space based on semantic or topical similarity. Most word embedding algorithms are pivoted around the distributional hypothesis that suggests words that share similar context have similar meanings [136]. Considering context in the embedding helps in capturing semantic and topical information as well as word relatedness and syntactic relations to a great extent. To clarify how context helps in identifying these relations, consider the following examples:

- **Disambiguation:** The word “bank” in “The bank will not be accepting cash on Saturdays.” and “The river overflowed the bank.” refers to two distinct meanings of the word “bank” that can be disambiguated using the context.

- **Semantic similarity:** The words “brother” in “I help my younger brother in his homeworks.” and “sister” in “I help my younger sister in her homeworks.” are semantically similar with a lot of common context.

- **Topic similarity:** The words “coffee” in “I drink coffee every morning.” and “milk” in “I drink milk every morning.” are both drinkable and in food/beverage category.

- **Syntactic similarity:** The words “go” in “I go to school.” and “went” in “I went to school.” are tenses of the same verb.

- **Relatedness:** The words “money” and “bank” in “Demand deposit withdrawals can be performed in person, via checks or bank drafts, using automatic teller machines (ATMs), or through online banking.” are related and have a lot of context words in common such as “deposit”, “withdrawals”, “checks”, “drafts”, etc.

Word embedding algorithms make use of this rich context information and try to preserve the word similarity and relatedness in the embedding space, so that words that share similar context will be placed close to each other. In this thesis, we propose a word embedding technique that also uses the context information when constructing the vector space. However, we apply a refinement technique on the contexts and remove the uninformative and insignificant co-occurrences. Another key distinguishing element of our method is our effective use of negative context: word pairs that never co-occur. This negative context is ignored in most algorithms, or is naively used in others. For instance, algorithms like Word2Vec [101] that only observe a local context at a time have no knowledge about negative context and cannot predict it either. Consequently, they use a naive sampling technique to incorporate negative context, hoping that the sampled words are not related to the current word. On the contrary, we calculate the global co-occurrence statistics, and therefore we accurately know the strength of positive association as well as the exact negative context. By exploiting the negative context, our algorithm pushes away unrelated words from each other while making the semantically similar words close to each other. This way, our method improves the distribution of words in the latent space and provides a higher quality embedding.

There exist another family of word embedding algorithms that work directly on the word-word co-occurrence matrix and are mostly known as frequency-based or count-based methods. These algorithms usually employ eigenanalysis on the co-occurrence matrix and extract the word vectors using algebraic techniques. Even for the modern embedding algorithms such as Word2Vec [101], it is shown that its objective function can be interpreted as an implicit matrix factorization problem. GloVe [122] is no exception either and is implicitly factorizing the log co-occurrence matrix. Although this family of algorithms have their own drawbacks, it is shown that there is no consistent advantage between prediction-based and frequency-based embeddings if they are used and tuned properly [90]. In this thesis, we analyze frequency-based word embeddings and propose a new spectral word embedding that takes into account the notion of negative examples. This was completely ignored in the previous algorithms and we shed some light on the usage of negative context in spectral methods. We provide an intuitive formulation for the embedding problem that justifies the use of negative examples and has an optimal closed-form solution through eigenanalysis.

An important challenge in natural language processing when using pre-trained embeddings is Word Sense Disambiguation (WSD). Since most popular embeddings provide a single vector for each word, it is not possible to disambiguate and identify the correct sense of the words. In fact, the learned word vectors are the weighted average of all the meanings of the words. Traditional disambiguation techniques based on context clustering are helpful in identifying the different senses, however, they do not provide embeddings for the senses, e.g. to be used in neural networks. Later in this thesis, we study the different WSD methods and propose a multi-sense embedding technique based on EigenWord. Our solution works on the co-occurrence matrix and using a soft clustering approach it breaks each global context vector into multiple sense-specific context vectors in order to obtain the sense-word co-occurrence matrix. Since there exist lots of common context among different senses of ambiguous words, the soft clustering strategy enables the fuzzy assignment of context words into different senses and enhances the disambiguation quality. At the end, EigenWord is applied on the sense-word co-occurrence matrix in order to obtain the sense embeddings.

Embedding algorithms require large amounts of training data to learn the embeddings and it is shown that the size of the corpus has a significant impact on the quality of embeddings. We are living in an era where huge amounts of textual data is generated every day, from blogs and tweets to product reviews and websites. This abundance of data readily available to researchers has made training of complex learning systems possible. On the one hand there exist huge amounts of unlabeled text data on the web, but on the other hand task-specific labeled data is rather small. This makes training of embedding directly for the downstream NLP tasks difficult and has given rise to the pre-trained embeddings. In fact, in most NLP applications pre-trained embeddings are used as input features to the models and it is shown that embeddings improve the accuracy of many NLP tasks by a great margin. The solutions that we provide in this thesis are also considered as methods for pre-training embeddings. These embeddings are usually trained on large corpora such as Wikipedia, large collections of tweets, Common Crawl, etc. It is often recommended to use a pre-trained embedding that is trained on a corpus consistent with the downstream

NLP task. For instance, if one is building a neural network model for Twitter data analysis, it is better if they use pre-trained embeddings on Twitter data as well. In our experiment, we also take this into account and pre-train embeddings appropriate to the underlying task.

We have conducted extensive experiments and evaluated our embedding algorithms both intrinsically and on downstream NLP tasks including emotion recognition and sentiment analysis. We have compared our proposed solutions with traditional approaches such as TF-IDF as well as with state-of-the-art methods like FastText [19] and Word2Vec. Results show the effectiveness of our proposed solutions and significant improvements in many application areas.

1.1 Contributions

The outcome of this research is a collection of methods to train high quality embeddings for natural language processing and manifold learning for images. The contributions of the thesis are[1]:

1. We propose a spherical representation learning algorithm for manifold learning in image datasets. We also propose an adaptive neighborhood and variance calculation for manifolds that better learns the structure of data. We show that the spherical representation combined with the Cosine metric enhances the separability of classes, and consequently, improves the clustering quality and visualization.

2. We study word embedding algorithms and propose a PMI-based embedding technique based on global co-occurrence statistics.

3. We analyze the use of negative context in the embeddings and propose a way to maximally exploit the negative examples in order to improve the distribution of words in the latent space.

4. We propose a kernel similarity measure specifically designed for high-dimensional data where the distances become unreliable. The kernel similarity measure gives the learning algorithm more discriminative power in the metric space.

[1]Parts of this thesis is published earlier in our research articles [142, 141, 144, 140, 139, 143, 138, 112, 111, 110].

5. We propose a fast nearest neighbor search for high-dimensional data based on Vantage Point (VP) trees. Almost all fast nearest neighbor search algorithms are ineffective in high dimensions and their performance is almost equal to exhaustive search. We use a modified VP-tree approach to approximate the forces in our algorithm in order to improve the computational complexity.

6. We introduce a way to enable the use negative examples in frequency-based embedding techniques. We propose a spectral word embedding algorithm that incorporates the negative context to achieve higher quality embedding.

7. We propose an intuitive formulation for the word embedding problem that justifies the use of negative context. We theoretically show through eigenanalysis that our proposed algorithm has an optimal closed-form solution.

8. We present a multi-sense embedding solution to tackle the Word Sense Disambiguation (WSD) problem. We show that learning the distribution of words across senses by employing a soft clustering of context vectors can better handle the common context among senses and results in better disambiguation.

1.2 Outline

The overall structure and organization of the thesis is as follows.

Chapter 2 In this chapter, we introduce a dimensionality reduction algorithm for image datasets, Unit Ball Embedding (UBE), that learns the structure of data and is particularly suitable for datasets with manifold structure. We propose an adaptive neighborhood construction techniques and use a spherical representation which together help in separability of manifolds and ultimately result in an inherent clustering of manifolds in the latent space. We apply the algorithm on various face, digit, and object recognition datasets and show the effectiveness of the proposed approach in clustering and visualization of high-dimensional data. Chapter 2 presents the main contributions from our research articles [139, 140, 143].

Chapter 3 In this chapter, we generalize the UBE algorithm and make it applicable to text domain and use it to learn word embeddings. In this case, we utilize the global co-occurrence statistics and refine it using Pointwise Mutual Information (PMI) and use the PMI matrix as the neighborhood connectivity structure in the algorithm. We also examine the use of negative context in the embeddings and propose a way to exploit the full power of negative examples in the embedding. Moreover, we propose a kernel similarity measure for the embedding space that gives the algorithm the discriminatory power to distinguish between the small and large distances in high dimensions. Furthermore, we present a fast version of the algorithm that approximates the forces from the negative context by adopting a VP-tree based fast nearest neighbor search. The original VP-tree search is not efficient for high-dimensional data, therefore, we propose a modified version of VP-tree that is designed to tackle curse of dimensionality. Chapter 3 is an extended version of our research article [142].

Chapter 4 In this chapter, we study the frequency-based word embeddings and propose a way to incorporate negative context in these types of embeddings. We introduce a new intuitive formulation for the word embedding problem that justifies the use of negative examples, and we theoretically show that it has an optimal closed-form solution through eigenanalysis. Consequently, we propose EigenWord, a spectral word embedding technique that exploits the negative examples to a great extent. Parts of Chapter 4 is published in our earlier research article [141].

Chapter 5 In this chapter, we investigate the Word Sense Disambiguation (WSD) problem and study the existing approaches. We then propose a multi-sense embedding method based on EigenWord. The proposed solution in this chapter uses an efficient soft clustering of context vectors to break down the global context vector of ambiguous words into multiple sense-specific context vectors. The soft clustering approach is perfectly suitable for this purpose since it inherently handles the context similarity among different senses. We show the effectiveness of this approach both qualitatively and quantitatively. Chapter 5 is an extended version of our research article [144].

Chapter 6 We evaluate our proposed embedding algorithms intrinsically in this chapter. We use Wikipedia as the training corpus and train the algorithms on all the articles of Wikipedia. We compare the quality of our embeddings with the state-of-the-art embedding techniques on multiple word similarity datasets. Results in this chapter show significant improvements achieved by our proposed algorithms on the word similarity task.

Chapter 7 This chapter presents the experimental results of our algorithms on two real-world downstream NLP tasks: emotion recognition from tweets and sentiment analysis from movie reviews. We compare our proposed solutions to the state-of-the-art word embedding algorithms as well as traditional methods (e.g. BOW and TF-IDF) and also document embedding solutions. Results show the effectiveness of the proposed algorithms in both tasks.

Chapter 8 A review of our findings and concluding remarks are presented in this chapter. We also briefly present our future research directions, which spawn from the research presented in this thesis.

Chapter 2

Dimensionality Reduction for Image Clustering

Mining, visualizing, and making sense of high-dimensional data is of great interest in machine learning community. In this chapter, we present an unsupervised nonlinear dimensionality reduction algorithm which is aimed to preserve the local structure of data by building and exploiting a neighborhood graph. Many high-dimensional data, such as object or face images, lie on a union of low-dimensional subspaces which are often called manifolds. The proposed method is able to learn the structure of these manifolds by exploiting the local neighborhood arrangement around each point. It tries to preserve the local structure by minimizing a cost function that measures the discrepancy between similarities of points in the high-dimensional data and similarities of points in the low-dimensional embedding. Our proposed method, Unit Ball Embedding (UBE), creates larger gaps between the manifolds compared to the state-of-the-art methods which results in better separability of clusters. By maximizing the within-cluster cohesion and between-cluster separation, UBE remarkably improves the quality of clustering algorithms on the low-dimensional embedding. Our experiments on face, object, and handwritten digit recognition datasets show that UBE can learn the structure of manifolds and it significantly improves the clustering quality and unsupervised learning.

2.1 Introduction

Data visualization and knowledge discovery using machine learning is of particular interest since most of the data being generated everyday is unlabeled and unstructured (e.g. text data). Dimensionality reduction and embedding algorithms are one of the core components in such unsupervised applications. The low-dimensional representation allows us to train learning algorithms more efficiently as well as reducing the chances of overfitting. It also enables us to visualize and make sense of the data more easily.

Training learning algorithms using huge number of variables will increase the chances of overfitting. For instance, training a deep neural network or any parametric model on a high-dimensional data will result in an exorbitant amount of parameters which in turn requires enormous data to be trained on [77]. There exist many application areas (e.g. medical domain) where we simply do not have enough data to train complicated neural networks in order to solve the problem. In these types of settings, a dimensionality reduction technique can become helpful to tackle high number of features compared to little amounts of data.

Dimensionality reduction is being referred by different names in the literature, such as feature reduction/transformation, multidimensional projection, representation learning, manifold learning and it is also related to distance metric learning. Generally speaking, the aim of dimensionality reduction methods is to embed input observations into a low-dimensional vector space. In the next section, we will categorize these methods and discuss them in details.

The two main categories of dimensionality reduction techniques are global and local methods, each of which has its own drawbacks. Global methods such as Multi-Dimensional Scaling (MDS) [36] and Sammon [135] try to preserve the global structure and configuration of points while local methods such as Locally Linear Embedding (LLE) [130], Laplacian eigenmaps [12], and t-distributed Stochastic Neighbor Embedding (t-SNE) [161] try to preserve the local neighborhood structure. Local approaches are also known as manifold learning methods. A manifold is basically a topological space that locally resembles Euclidean space around each point. These algorithms try to capture the manifold structure by assuming and exploiting the locally smooth properties of the data.

Global approaches have several drawbacks and disadvantages. They require notably more resources than local methods both in terms of computational complexity and memory complexity, mostly because of calculating, storing, and maintaining all pairwise distances. Moreover, they are less accurate than local methods. This is because they are driven to preserve small and large distances to the same extent. Having to preserve large distances makes the algorithm to lose the ability to model the local neighborhoods' structure.

Local approaches also have their own weaknesses. They depend on the smoothness assumption and the neighborhood connectivity graph. These local approaches work pretty well in datasets with manifold structure. However, there are certain situations when manifold learning approaches may fail: noise around the manifold, high curvature of manifold, high intrinsic dimensionality of manifold, and lastly presence of many manifolds in the data [15].

In this chapter, we propose a nonlinear local dimensionality reduction method, Unit Ball Embedding (UBE), which tries to address some of the aforementioned drawbacks of previous methods. The proposed algorithm uses a sparse representation of input data similarities, and consequently, does not suffer from quadratic memory complexity of global approaches. Moreover, we use adaptive variance calculation in our input space Gaussian similarities which helps to identify multiple manifolds. This adaptive variance technique helps in both tackling high intrinsic dimensionality of manifolds and also dealing with presence of many manifolds. The algorithm is designed in a way that exploits the repulsion force between the neighborhoods in the optimization which leads to creation of large gaps between the manifolds. This property makes UBE a very useful visualization tool. Furthermore, by ensuring a larger margin between the classes, UBE results in clustering quality higher than in other embedding spaces, including the original feature space.

2.2 Background and Related Work

In this chapter, we will use a consistent notation for the equations as described in Table 2.1.

Dimensionality reduction is being referred by different names in the literature such as feature reduction/transformation, multidimensional projection, representation learning, manifold learning and it is also related to distance metric learning. There are supervised as well as unsupervised algorithms for dimensionality reduction. The supervised algorithms make use of the labels to transform the data into some low-dimensional space in which the classes are easily separable. On the other hand, unsupervised techniques look solely at the structure of points and their relative distances without consideration of labels. Most notable supervised algorithms are Linear Discriminant Analysis (LDA) [17] and its variants such as kernelized LDA [100], Neighborhood Component Analysis (NCA) [129]. Some supervised methods find the

Table 2.1: The symbols and notation used in the chapter.

Symbol	**Dimension**	**Description**
n	integer	# of datapoints
p	integer	# of dimensions in the input feature space
d	integer	# of dimensions in the output space (mapped data)
k	integer	Maximum # of neighbors in the k-NN graph
$\mathbf{X}$	$n \times p$	Input data matrix
x_i	$p \times 1$	i-th datapoint in the input space
$\mathbf{Y}$	$n \times d$	Output data matrix
y_i	$d \times 1$	i-th datapoint in the target space
$\mathbf{W}$	$n \times n$	Sparse adjacency matrix (i.e. k-NN graph)
w_{ij}	double	Adjacency value between x_i and x_j
d_{ij}	double	Distance between x_i and x_j
α	double	Step size or learning rate in gradient descent
λ_i	double	Lagrange multiplier for i-th constraint
λ	double	Regularization Parameter

mapping by learning a distance metric. The distance metric learning incorporates the labels, and thus, the resulting target space will have better class separability. Among these methods Large Margin Nearest Neighbor (LMNN) [168] and Maximally Collapsing Metric Learning (MCML) [52] can be mentioned. Although these algorithms have been used in a wide range of applications, supervised methods are outside of the scope of this thesis.

Unsupervised dimensionality reduction methods can also be divided into linear and nonlinear methods, described in the following sections.

2.2.1 Linear Dimensionality Reduction

In linear algorithms each attribute in the target space is considered to be a linear combination of input features. More formally, if $\mathbf{X}_{n\times p}$ is the coordinates of points in the original feature space and $\mathbf{Y}_{n\times d}$ is the coordinates of points in the target space, where p and d are the dimensionality of input space and target space, respectively, then $\mathbf{Y} = \mathbf{XP}$ where $\mathbf{P}_{p\times d}$ defines the linear mapping.

One of the traditional linear methods for dimensionality reduction which is widely used in many different application is PCA [70]. It picks the dimensions of the greatest variance in the data and maps all the points to the new coordinate system which is built by the maximum variance directions.

Some of the linear algorithms such as PCA and classical MDS impose orthogonality constraint on the attributes of the target space. Orthogonality of the attributes is of great comfort to visualizations and many application areas since it ensures that different types of structure will not be artificially created in the data. Among other linear methods that use unconstrained objective functions we can mention Undercomplete Independent Component Analysis (UICA) [169], Probabilistic PCA (PPCA) [153] and Factor Analysis (FA) [72].

ICA specifies the data $\mathbf{X}$ as a mixture of unknown and independent sources $\mathbf{Y}$. The independence in ICA is different from uncorrelatedness in PCA. Here the independence means that for all points y_i in $\mathbf{Y}$ we have $p(y_i) \approx \prod_{j=1}^{d} p(y_i^j)$ where the $p(y_i^j)$ are the univariate marginals of the lower dimensional data. PPCA adds a prior to PCA and assumes that high-dimensional data is a linear mapping of low-dimensional data plus a Gaussian noise. This model yields a natural likelihood objective which has a closed form solution using singular value decomposition. Factor Analysis is a more general case of PPCA, in which noise is fit per observation rather than across all the data. This will lead to a different likelihood function that, unlike PPCA, does not have closed form solution. Typically, Expectation-Maximization (EM) or gradient descent is used for optimizing the likelihood in Factor Analysis.

2.2.2 Nonlinear Dimensionality Reduction

Linear dimensionality reduction algorithms cannot capture nonlinearities in the data. Nonlinear dimensionality reduction methods namely Isomap [151], Laplacian eigenmap [12], Locally Linear Embedding (LLE) [130], etc. have been proposed to overcome this issue. In this chapter, we will mainly focus on nonlinear algorithms since our proposed algorithm is also a nonlinear method. Nonlinear dimensionality reduction algorithms can further be broken down into global and local approaches.

2.2.2.1 Global Approaches

Global methods try to preserve the global structure of the data by sustaining pairwise distances between all pairs of points. Among these methods we can mention Multi-Dimensional Scaling (MDS) [36], modern MDS [20], Sammon [135], Isomap [151], multilayer autoencoders [60], Maximum Variance Unfolding (MVU) [167], and Diffusion Maps [80]. Almost all methods in this category try to somehow preserve the big picture and global structure of the data. Modern MDS methods optimize a stress objective function of the following form:

$$\mathcal{J}(\mathbf{Y}) = \sqrt{\frac{\sum_i \sum_j (d_{ij} - \|y_i - y_j\|)^2}{\sum_i \sum_j d_{ij}^2}} \tag{2.1}$$

where d_{ij} are the distances in the original input space. It basically tries to maintain all the pairwise distances. Sammon mapping [135] uses a similar objective function but it is different in scaling and normalization. Onclinx et al. [118] proposed a rank preserving dimensionality reduction method which uses a spherical representation and tries to preserve the ranks of distances instead of the distance values. However, this rank-based dimensionality reduction method also suffers from the same drawbacks of global approaches, since the order/rank of large distances should not be as important as the order/rank of small distances.

Isomap [151] initially builds a *knn* or *ϵ-radius* neighborhood graph, then it recalculates all the pairwise distances as the shortest distance on the graph. These shortest path distances on the graph are called geodesic distances. Finally, it uses MDS to find a coordinate system that preserves the geodesic distances.

MVU [167] also builds a neighborhood graph G and uses the connectivities of the graph as constraints in its optimization. It tries to maximize all pairwise distances with the constraints that the distance of those points connected in the graph should be preserved:

$$\begin{gathered} Maximize \sum_i \sum_j \|y_i - y_j\|^2 \\ subject\ to: \|y_i - y_j\| = d_{ij}\ for\ \forall (i,j) \in G \end{gathered} \tag{2.2}$$

MVU uses semidefinite programming to optimize its cost function. Landmark MVU [37] is another improved variant which is more efficient for large datasets.

Recently, deep neural networks are gaining the attention of researchers [83]. They can be used for representation learning since each layer of the network provides a new representation of the data [13]. For instance, autoencoders [60] consist of an encoder and a decoder. The encoder maps the input to some low-dimensional representation and the decoder takes the low-dimensional representation and map it to some high dimensional space. Autoencoders try to minimize the reconstruction error between the original input and the reconstructed high-dimensional output. If the reconstruction error is low, it means that the output of the encoder (i.e. low-dimensional representation) is capturing most of the structure and information in the input.

2.2.2.2 Local Approaches

The last group of methods to consider is the nonlinear local dimensionality reduction approaches. The intuition for local methods is that if two points are close in the high-dimensional space, they should be close in the low-dimensional embedding as well. Among them we can mention LLE [130], Laplacian eigenmap [12], Locality Preserving Projections (LPP) [117], Hessian LLE [40], Local Tangent Space Alignment (LTSA) [179], Linear Local Tangent Space Alignment (LLTSA) [178], Local Convex Hull (LoCH) [45]. All these methods look at neighborhood structures and try to preserve the local geometry of the data, but the global structure may change.

Locally Linear Embedding (LLE) [130] describes the local properties of the manifold around point x_i by representing the datapoint as a linear combination of its neighbors. A sparse matrix $\mathbf{W}$ of weights is calculated from the high-dimensional data and the same weights are used to reconstruct the low dimensional representation that results in the following objective function.

$$\mathcal{J}(\mathbf{Y}) = \sum_{i=1}^{n} \|y_i - \sum_{j=1}^{n} w_{ij} y_j\|_2^2 \tag{2.3}$$

Using matrix notation it can be written down as:

$$\mathcal{J}(\mathbf{Y}) = \|\mathbf{Y} - \mathbf{W}\mathbf{Y}\|_F^2 \tag{2.4}$$

where $\|.\|_F$ is the Frobenius norm. Equation 2.4 can be written in terms of trace of a matrix:

$$\begin{aligned} \mathcal{J}(\mathbf{Y}) &= \|\mathbf{Y} - \mathbf{W}\mathbf{Y}\|_F^2 \\ &= trace[\mathbf{Y}^T(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})\mathbf{Y}] \end{aligned} \tag{2.5}$$

Based on a theorem in matrix trace optimization, if we add an orthonormal constraint on output data $\mathbf{Y}^T\mathbf{Y} = \mathbf{I}$, then the eigenvectors of $(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ corresponding to the smallest nonzero eigenvalues form the optimal solution that minimizes the cost function [76].

Laplacian eigenmap [12] constructs a *knn* neighborhood graph in which each point is connected to its k nearest neighbors. A sparse matrix $\mathbf{W}$ of *knn* distances is obtained and used for optimization. This matrix can be interpreted as a weighted adjacency graph. If two points are connected in the high dimensional space, their distance in low dimensional representation is minimized, otherwise their distance does not have any effect in the cost function.

$$\mathcal{J}(\mathbf{Y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\|y_i - y_j\|_2^2 \tag{2.6}$$

If we define a diagonal matrix $\mathbf{M}$ with diagonal elements $m_{ii} = \sum_{i=1}^{n} w_{ij}$ then the Laplacian $\mathbf{L}$ of the graph $\mathbf{W}$ can be defined as $\mathbf{L} = \mathbf{M} - \mathbf{W}$. This Laplacian allows us to formulate the minimization problem as an eigenproblem.

$$\begin{aligned} \mathcal{J}(\mathbf{Y}) &= \sum_i\sum_j w_{ij}(y_i^T y_i + y_j^T y_j - 2y_i^T y_j) \\ &= 2\sum_i y_i^T y_i \sum_j w_{ij} - 2\sum_i\sum_j w_{ij} y_i^T y_j \end{aligned} \tag{2.7}$$

By introducing $\mathbf{M}$ and using matrix notation we have:

$$\begin{aligned} \mathcal{J}(\mathbf{Y}) &= 2trace[\mathbf{Y}^T\mathbf{M}\mathbf{Y}] - 2trace[\mathbf{Y}^T\mathbf{W}\mathbf{Y}] \\ &= 2trace[\mathbf{Y}^T(\mathbf{M} - \mathbf{W})\mathbf{Y}] \end{aligned} \tag{2.8}$$

Similar to LLE it has a closed form optimal solution in that $\mathbf{Y}$ is built by eigenvectors of the Laplacian $\mathbf{L}$ corresponding to its smallest nonzero eigenvalues.

Hessian LLE is a variant of LLE that tries to minimize the curvature of the high dimensional manifold when finding the low dimensional embedding. The curvature of the manifold is measured using the local Hessian at each datapoint. Similar to Hessian LLE, LTSA also describes the local properties of the data using the local tangent space of each datapoint. LTSA is based on the idea that if local linearity of the manifold is assumed, there exists a linear mapping from a point to its local tangent space and there also exists a linear mapping from the corresponding low dimensional representation of the point to the same local tangent space [179].

Stochastic Neighbor Embedding (SNE) [59] is another successful method in that for each point x_i, a probability distribution over its neighbors is calculated as follows.

$$p_{ij} = \frac{exp(-d_{ij}^2)}{\sum_{k \neq i} exp(-d_{ik}^2)} \tag{2.9}$$

Then an iterative optimization procedure minimizes the sum of Kullback-Leibler divergences between the distributions in high dimensional space and low dimensional embedding. The objective function of SNE has the following form.

$$\mathcal{J} = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \log \frac{p_{ij}}{q_{ij}} = \sum_{i=1}^{n} KL(P_i || Q_i) \tag{2.10}$$

where q_{ij} is the same as p_{ij} but it is calculated in low dimensional space. Several variants of SNE such as Symmetric SNE [172] and t-distributed Stochastic Neighbor Embedding (t-SNE) [161] have been introduced. t-SNE is a successor which uses a t-distribution rather than Gaussian for calculating the similarities in the low-dimensional embedding. The t-distribution is heavier-tailed than Gaussian and results in more gaps between groups of points. This property makes t-SNE a great visualization tool that is widely used for many application areas. An accelerated version of t-SNE is also proposed that uses tree based algorithms to improve its computational complexity [160]. Accelerated t-SNE uses Vantage-Point (VP) trees for finding nearest neighbors, as well as quad-trees and oct-trees combined with Barnes-Hut technique [8] to approximate the gradient during the optimization. However, due to the structure of trees, its computational and memory complexities are exponential in terms of the dimensionality of embedding. This makes it only suitable for visualization purposes.

Local Convex Hull (LoCH) [45] is also a recent neighborhood based multidimensional projection technique that works based on local convex hulls of datapoints. A detailed comparative study of nonlinear dimensionality reduction is provided by Van Der Maaten et al. [159].

Our proposed method follows the locality preserving approach. It tries to preserve the neighborhood connectivity graph in the low-dimensional embedding. The dimensionality of the target space can be chosen by the user and the method locates the points on the surface of a hypersphere in such a way that it retains most of the neighborhood structure.

2.3 Proposed Dimensionality Reduction Method

Our proposed method, Unit Ball Embedding (UBE), is a local dimensionality reduction method which tries to preserve the neighborhood structure. We use a nearest neighbor graph to capture the local structure and define an objective function that preserves the structure. The objective function is defined in a way to minimize the discrepancy between similarities of points in the input space and similarities of points in the transformed feature space. The dimensionality of the target space can be chosen by the user.

As the name of the algorithm suggests, we use a spherical representation for the low-dimensional embedding in which points are located on the surface of a hypersphere. This type of spherical representation is of great interest in the natural language processing community and in text mining applications. It can also be embedded in interactive visualization tools to provide insightful and interactive visualizations.

In the following, we first propose an effective way to determine the structure of input data using a sparse representation for input similarities. We then propose our objective function that is aimed to minimize the squared difference between similarities of points in input and output spaces. Eventually, we investigate two different optimization technique for minimizing the cost function followed by experimental results and visualizations of the embedding space.

2.3.1 Calculating Input Similarities

The proposed method starts by calculating k_{max} nearest neighbors for each point using Euclidean distance. The set of neighboring points for $\mathbf{x}_i$ is represented as

$\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_{k_{max}}^{(i)}\}$ where $\mathbf{x}_j^{(i)}$ denotes the j-th nearest neighbor of $\mathbf{x}_i$ and $d_j^{(i)}$ denotes the distance between $\mathbf{x}_i$ and its j-th nearest neighbor. Distances to the nearest neighbors are then converted to affinity values by centering a Gaussian function on top of each datapoint. We adaptively choose the number of neighbors as well as the variance of the Gaussian for each datapoint since the density around points can be very different. This will ensure that the algorithm can model the local neighborhood structure even when having various density levels in the dataset.

One of the most important factors in choosing the right number of neighbors is the intrinsic dimensionality of the manifolds. If the data lies on a line or plane in a local neighborhood, a few number of neighbors is enough to reasonably represent the structure of the manifold in that region. But as the intrinsic dimensionality of manifold increases, more neighbors are needed to effectively capture the structure. In this work, we do not estimate or approximate the intrinsic dimensionality of manifold, however, we propose an elbow based technique on the distances to find a good cut-off neighbor/distance for each datapoint. If we consider the first k_{max} neighbors of $\mathbf{x}_i$ ordered in increasing distance, we find the elbow point after which there is no significant increase in the distances. This can be a good indication of moving to another cluster or manifold in the data which we exploit in order to maximize the separation of different manifolds. Figure 2.1 illustrates the 100 nearest neighbor distances for a sample point in COIL20 image dataset. The elbow point $\mathbf{x}_{elbow}^{(i)}$ is represented as red which has the maximum distance from the line connecting first and last point.

After detecting the elbow, we choose the variance of the Gaussian for each datapoint in a way that its elbow point falls exactly on 3σ of the Gaussian. Then we will set the similarities for points outside of 3σ zone to zero. More formally, the input similarities are calculated as follows:

$$w_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}) & \mathbf{x}_j \in \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_{elbow}^{(i)}\} \\ 0 & otherwise \end{cases} \tag{2.11}$$

$$\sigma_i = \frac{\|\mathbf{x}_i - \mathbf{x}_{elbow}^{(i)}\|}{3} = \frac{d_{elbow}^{(i)}}{3} \tag{2.12}$$

where $\mathbf{x}_j^{(i)}$ is the j-th nearest neighbor to the i-th datapoint and w_{ij} is their corresponding similarity. Unlike global dimensionality reduction methods (e.g. MDS,

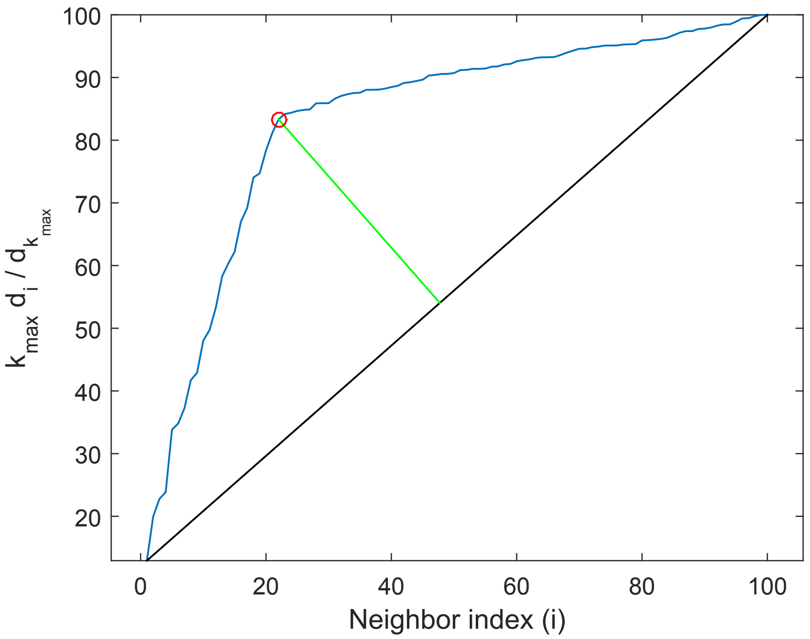


Figure 2.1: The elbow point shown as red for a randomly selected point in COIL20 dataset, found based on the distribution of distances to its neighbors.

Sammon, ISOMAP), UBE does not need to store all the pairwise distances which requires $O(n^2)$ memory where n is the number of points in the data. Since each point is only connected with a few neighbors, the entire input similarity graph can be stored as a sparse adjacency matrix. This sparse matrix of input similarities, $\mathbf{W}$, contains at most $O(nk_{max})$ non-zero elements which is linear in terms of input datapoints. Having a low memory complexity is a vital property for running the algorithm on large datasets.

2.3.2 Cost Function

Our proposed method uses a spherical representation for the output data which is interesting for interactive visualization tools and also for natural language processing applications. In this representation, we use cosine similarity as our similarity measure between datapoints:

$$S(\mathbf{y}_i, \mathbf{y}_j) = \frac{\mathbf{y}_i^T \mathbf{y}_j}{\|\mathbf{y}_i\| \|\mathbf{y}_j\|} \tag{2.13}$$

The objective function is then defined to minimize the sum of squared differences between similarities in the input space and the similarities in the transformed feature space:

$$\mathcal{J}(\mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\big(w_{ij} - S(\mathbf{y}_i, \mathbf{y}_j)\big)^2 \tag{2.14}$$

where w_{ij} and $S(\mathbf{y}_i, \mathbf{y}_j)$ are calculated using Equations 2.11 and 2.13, respectively. Here we propose a constrained variant of the objective function which makes the normalizations fade away as well as resulting in a simplified gradient:

$$\mathcal{J}(\mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\big(w_{ij} - \mathbf{y}_i^T\mathbf{y}_j\big)^2$$

$$\textit{subject to}: \mathbf{y}_i^T\mathbf{y}_i = 1 \ \forall i \tag{2.15}$$

Here, the objective function forces the points to be on the surface of a hypersphere. The unit length constraints is not restrictive, since most of the volume of a hypersphere is in its crust, and therefore its surface has enough room to locate all the points in a desired way. More precisely, the surface of the hypersphere has only one less degrees of freedom than the entire volume of the hypersphere. In the constrained form of the objective function, the dot product of output vectors gives the cosine similarity and is used as a kernel in the transformed feature space to weight the closer points more than the farther points in a nonlinear manner.

The similarities of the points in the input space and transformed feature space are calculated using Gaussian and Cosine kernels, respectively. However, Gaussian ranges in $[0, +1]$ and cosine ranges in $[-1, +1]$. For this reason, we normalize the cosine kernel in $[0, +1]$ by using $\frac{1}{2}(\mathbf{y}_i^T\mathbf{y}_j + 1)$. This way, if the Gaussian similarity of two points in the input space is high, the method tries to place them as close as possible. And if they are not connected in the input space, $w_{ij} = 0$, the method tries to place them as far as possible.

2.3.3 Optimization

We find the embedding in the low-dimensional space in an iterative optimization procedure. We have employed two different optimization techniques to minimize our objective function. In both cases, we initialize the datapoints randomly on the surface of a hypersphere and try to change the configuration of points to get a better objective value. We have used the general projected gradient descent framework [91] to project the solution of each iteration onto the constraints, $\mathbf{y}_i^+ = P_{\mathcal{Y}}(\mathbf{y}_i^- - \alpha\frac{\partial \mathcal{J}}{\partial \mathbf{y}_i})$ where $P_{\mathcal{Y}}(y) = \arg\min_{z \in \mathcal{Y}} \|y - z\|^2$ is the projection that finds the closest solution in the feasible region (i.e. satisfying the constraints). In other words, the solution is projected onto the feasible region after each iteration.

2.3.3.1 Stochastic Gradient Descent Optimization

If we expand the cost function we will get:

$$\mathcal{J}(\mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Big[w_{ij}^2 - 2w_{ij}(\mathbf{y}_i^T\mathbf{y}_j) + (\mathbf{y}_i^T\mathbf{y}_j)^2\Big] \tag{2.16}$$

In each iteration of the algorithm, datapoints are being relocated in order to achieve a lower loss value. The gradient of the cost function with respect to a datapoint $\mathbf{y}_i$ is:

$$\begin{aligned}\nabla\mathcal{J}(\mathbf{y}_i) = \frac{\partial \mathcal{J}}{\partial \mathbf{y}_i} &= \sum_{j=1}^{n}\Big[-w_{ij}\mathbf{y}_j + (\mathbf{y}_i^T\mathbf{y}_j)\mathbf{y}_j\Big] \\ &= \overbrace{-\sum_{j=1}^{n} w_{ij}\mathbf{y}_j}^{v_1} + \overbrace{\sum_{j=1}^{n}(\mathbf{y}_i^T\mathbf{y}_j)\mathbf{y}_j}^{v_2}\end{aligned} \tag{2.17}$$

$$\mathbf{y}_i^+ = \mathbf{y}_i^- - \alpha\nabla\mathcal{J}(\mathbf{y}_i) \tag{2.18}$$

where α is the learning rate, $\mathbf{y}_i^-$ is the coordinates of the point before updating and $\mathbf{y}_i^+$ is its coordinates after being updated. We have used a time decaying learning rate for the experiments.

The gradient consists of two components v_1 and v_2. The former, v_1, defines the sum of attractive forces being applied to the point while the latter, v_2, defines the sum of repulsive forces being applied to the point. This idea of attractive and repulsive forces in the embedding is similar to the ones in Elastic Embedding algorithm [28], however, the affinity measures, formulations of the problem, and the objective functions are different. The attractive force tries to attract the point of interest to the center of gravity of its neighbors in the input space. The repulsive force tries to repel all other points from the point of interest. Since $\mathbf{W}$ is a sparse affinity matrix, only the connected edges in the neighborhood graph will have a contribution in the attractive force v_1. Please note that, we normalize the similarities of points in the transformed feature space in [0, +1] by using $\frac{1}{2}(\mathbf{y}_i^T\mathbf{y}_j + 1)$ instead of the dot product. This way, v_2 will be the sum of weighted repulsive forces applied from different points in the data.

Minimizing v_1 leads to maximization of the similarity in target space whenever two points are connected in input space while minimizing v_2 is equivalent to pushing all points away from each other. From this perspective, UBE and Maximum Variance Unfolding (MVU) [167] share the same interest: maximize all the pairwise distances except for the ones that are close in the input space.

The optimization continues until some stopping criteria are met. The algorithm can be stopped after a certain number of iterations have passed or simply if no significant improvement in the objective value is observed. Of course, the cost function cannot be minimized to zero, but it converges to a stable configuration of points in the target space. The computational complexity of our optimization is $O(in^2d)$ where i is the number of iterations in the optimization. Therefore, it is equal to that of t-SNE.

2.3.3.2 Newton's Optimization Method

Higher order derivatives can generally be used to extract more information about the curvature and have a more accurate approximation of the cost function behavior at a particular point during the optimization [26]. Consequently, it leads to faster optimization and having better convergence rate than first order gradient based optimization techniques. In our method, the second order derivatives can be obtained and the Hessian can be formed easily. Based on Equation 2.17, first and second order derivatives with respect to different dimensions of a datapoint can be obtained by:

$$\frac{\partial \mathcal{J}}{\partial y_{ir}} = -\sum_{j=1}^{n} w_{ij} y_{jr} + \sum_{j=1}^{n} (\mathbf{y}_i^T \mathbf{y}_j) y_{jr} \tag{2.19}$$

$$\frac{\partial \mathcal{J}}{\partial y_{ir} \partial y_{is}} = \sum_{j=1}^{n} y_{jr} y_{js} \tag{2.20}$$

where y_{ir} is the r-th dimension of i-th datapoint in transformed feature space. Then the Hessian matrix at point $\mathbf{y}_i$ can be calculated in the following form:

$$\begin{aligned}\mathbf{H}(\mathbf{y}_i) &= \nabla^2 \mathcal{J}(\mathbf{y}_i) \\ &= \begin{bmatrix} \sum_{j=1}^{n} y_{j1}y_{j1} & \dots & \sum_{j=1}^{n} y_{j1}y_{jd} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{n} y_{jd}y_{j1} & \dots & \sum_{j=1}^{n} y_{jd}y_{jd} \end{bmatrix} = \mathbf{Y}^T\mathbf{Y}\end{aligned} \tag{2.21}$$

Then, the Newton direction $N(\mathbf{y}_i)$ will be the solution of the following linear system:

$$\nabla^2 \mathcal{J}(\mathbf{y}_i) N(\mathbf{y}_i) = -\nabla \mathcal{J}(\mathbf{y}_i) \tag{2.22}$$

The Newton direction can be obtained efficiently without requiring matrix inversion and by using LU factorization of the Hessian matrix [155]. Eventually, the location of points can be updated using the Newton direction:

$$\mathbf{y}_i^{+} = \mathbf{y}_i^{-} + \alpha N(\mathbf{y}_i) \tag{2.23}$$

Newton's optimization method has better convergence rate than first order gradient based optimization techniques and is mainly used in convex optimization [92]. Using Newton's method in non-smooth and non-convex functions may cause instability or even overshooting and divergence form the solution. However, in smooth non-convex functions as ours, it can be used by employing a decreasing learning rate [175].

In terms of computational complexity, the amount of time required to compute the Hessian for a given point, $\mathbf{H}(\mathbf{y}_i) = \mathbf{Y}^T\mathbf{Y}$, is $O(nd^2)$ which compared to $O(nd)$ for calculating the first order gradient, is a bit costlier. However, since d is usually very small, the difference in timing is negligible. In general, d is 2-3 for visualization and around 50 for vector embedding applications. In total, the Newton's optimization method will require $O(in^2d^2)$ time, where i is the number of iterations in the optimization.

Algorithm 1 summarizes all the steps in our embedding algorithm. UBE is implemented in C using OpenMP parallel computing library. The datasets used in this chapter are available online[1] and the source code of the algorithm is available at our

[1] `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/UXU6Z3`

Algorithm 1 Unit Ball Embedding (UBE)

```
Input: $\mathbf{X}_{n\times p}, d, k_{max} = 50, l = 3, Iter = 250$
Output: $\mathbf{Y}_{n\times d}$
$\mathbf{Y} = rand(n, d)$
$\mathbf{W} = \mathbf{0}$
for $i = 1$ to $n$ do
    $\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_{k_{max}}^{(i)}\} = knn(\mathbf{x}_i, k_{max})$
    $\mathbf{x}_{elbow}^{(i)} = elbow(\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_{k_{max}}^{(i)}\})$
    $\sigma_i = \frac{\|\mathbf{x}_i - \mathbf{x}_{elbow}^{(i)}\|}{3} = \frac{d_{elbow}^{(i)}}{3}$
    for $\mathbf{x}_j \in \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_{elbow}^{(i)}\}$ do
        $w_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2})$
    end for
end for
for $t = 1$ to $Iter$ do
    for $i = 1$ to $n$ do
        $v_1 = -\sum_{j=1}^{n} w_{ij}\mathbf{y}_j$            // Attractive force
        $v_2 = \sum_{j=1}^{n} (\mathbf{y}_i^T \mathbf{y}_j)\mathbf{y}_j$          // Repulsive force
        $\nabla \mathcal{J}(\mathbf{y}_i) = v_1 + v_2$
        $\nabla^2 \mathcal{J}(\mathbf{y}_i) = \mathbf{Y}^T \mathbf{Y}$
        $\nabla^2 \mathcal{J}(\mathbf{y}_i) N(\mathbf{y}_i) = -\nabla \mathcal{J}(\mathbf{y}_i)$  // Solve linear system
        $\mathbf{y}_i = \mathbf{y}_i + \alpha N(\mathbf{y}_i)$               // Newton's method
        $\mathbf{y}_i = \frac{\mathbf{y}_i}{\|\mathbf{y}_i\|}$
    end for
    $\alpha = \frac{0.1}{\sqrt{t}}$
end for
```

GitHub page[2].

2.4 Experiments and Results

Image recognition datasets are considered as classical examples of high-dimensional data since they consist of large number of pixels. If each pixel is treated as a feature,

[2] `https://github.com/behrouzhs/kube`

Table 2.2: Image recognition datasets used for comparison of dimensionality reduction methods and clustering.

Dataset Name	**Type**	**Images**	**Classes**	**Dimensions**
Yale [11]	Face	165	15	64×64
Olivetti [134]	Face	400	40	64×64
Umist [53]	Face	575	20	112×92
CMUfaces [105, 42]	Face	640	20	128×120
COIL-20 [114]	Object	1440	20	32×32
Optdigits [42]	Digit	5620	10	32×32
YaleB [51, 87]	Face	2414	38	32×32
COIL-100 [113]	Object	7200	100	32×32
USPS [65, 84]	Digit	9298	10	16×16
MNIST (test set) [85]	Digit	10000	10	28×28

images can easily have tens of thousands of features. In this work 10 different image recognition tasks are chosen to examine the effect of dimensionality reduction on the data. We have used face recognition, handwritten digit recognition and object recognition as benchmark datasets. Table 2.2 describes the datasets used in this chapter and their number of images, classes, as well as the dimensions of the images. Using these datasets, we treat each image as a vector of pixel intensities. The number of pixels in these datasets varies from 256 in USPS handwritten digits to 15386 in CMUfaces dataset.

We have chosen several state-of-the-art dimensionality reduction algorithms to transform each dataset to a three dimensional space and compared the quality of our embedding with those algorithms.

For evaluating the quality of mapping, we have run unsupervised learning algorithms on the output of each embedding method to see how learning algorithms can perform on the low-dimensional embedding. In fact, we associate the quality of clustering algorithms on transformed feature space to the embedding algorithm. In particular, we have used k-means clustering [57] and spectral clustering [115] on low-dimensional data to see how good the clustering results will be on the transformed feature space. The output of clustering is evaluated using Normalized Mutual Information (NMI) [4, 44]. We have also used Silhouette metric for measuring cohesion and separation of clusters in the embedding space. Another way to evaluate the low-dimensional embeddings is to use them in supervised tasks such as classification. In

Table 2.3: Results of k-means clustering on the low-dimensional embedding evaluated by Normalized Mutual Information (NMI) multiplied by 100. The first row for each dataset shows the mean and standard deviation of NMI values in 50 runs and the second row shows the p-value of t-test against the best performing method on that dataset.

	UBE	**t-SNE**	**Raw Data**	**LLE**	**PCA**	**Sammon**
Yale	**61.9 ± 1.2**	60.5 ± 1.9	53.3 ± 3.4	52.1 ± 1.9	49.2 ± 1.6	48.7 ± 1.3
	1.0000	$\approx 10^{-5}$	$\approx 10^{-29}$	$\approx 10^{-50}$	$\approx 10^{-65}$	$\approx 10^{-71}$
Olivetti	79.8 ± 0.6	**80.5 ± 1.1**	73.8 ± 1.6	71.8 ± 0.7	63.3 ± 0.7	64.5 ± 0.7
	0.0007	1.0000	$\approx 10^{-41}$	$\approx 10^{-65}$	$\approx 10^{-94}$	$\approx 10^{-91}$
Umist	**84.4 ± 1.4**	74.2 ± 1.4	64.5 ± 2.0	58.0 ± 0.5	58.4 ± 1.3	59.5 ± 1.0
	1.0000	$\approx 10^{-56}$	$\approx 10^{-75}$	$\approx 10^{-107}$	$\approx 10^{-97}$	$\approx 10^{-99}$
CMUfaces	**89.4 ± 0.8**	82.8 ± 2.6	75.7 ± 3.0	74.3 ± 1.5	59.5 ± 1.2	59.0 ± 1.2
	1.0000	$\approx 10^{-30}$	$\approx 10^{-51}$	$\approx 10^{-78}$	$\approx 10^{-113}$	$\approx 10^{-114}$
COIL20	**92.6 ± 1.3**	84.9 ± 2.3	75.4 ± 1.6	75.4 ± 1.3	71.9 ± 1.0	71.2 ± 1.7
	1.0000	$\approx 10^{-36}$	$\approx 10^{-78}$	$\approx 10^{-82}$	$\approx 10^{-94}$	$\approx 10^{-85}$
Optdigits	**86.0 ± 0.8**	84.5 ± 4.7	73.6 ± 2.4	75.3 ± 1.7	61.3 ± 1.5	52.8 ± 1.7
	1.0000	0.0307	$\approx 10^{-55}$	$\approx 10^{-61}$	$\approx 10^{-100}$	$\approx 10^{-109}$
YaleB	25.6 ± 0.4	25.7 ± 1.1	13.1 ± 0.8	**33.8 ± 0.4**	10.1 ± 0.2	10.9 ± 0.3
	$\approx 10^{-97}$	$\approx 10^{-69}$	$\approx 10^{-118}$	1.0000	$\approx 10^{-151}$	$\approx 10^{-144}$
COIL100	**90.1 ± 0.3**	88.4 ± 0.5	76.0 ± 0.5	74.2 ± 0.5	71.4 ± 0.2	72.4 ± 0.3
	1.0000	$\approx 10^{-31}$	$\approx 10^{-117}$	$\approx 10^{-123}$	$\approx 10^{-145}$	$\approx 10^{-139}$
USPS	84.9 ± 1.1	**85.8 ± 3.2**	60.9 ± 0.9	34.8 ± 0.6	41.3 ± 1.0	44.7 ± 0.5
	0.0706	1.0000	$\approx 10^{-73}$	$\approx 10^{-104}$	$\approx 10^{-97}$	$\approx 10^{-95}$
MNIST	81.2 ± 1.8	**81.8 ± 2.4**	50.9 ± 1.7	60.4 ± 1.0	38.9 ± 0.3	36.7 ± 0.7
	0.1657	1.0000	$\approx 10^{-87}$	$\approx 10^{-76}$	$\approx 10^{-109}$	$\approx 10^{-109}$

terms of classification accuracy, our method provides similar results to the state-of-the-art algorithms such as t-SNE. Moreover, several graph-based evaluation measures have been recently proposed [107] for the assessment of dimensionality reduction methods. However, in this work we have mainly evaluated different methods by using the embeddings in unsupervised machine learning tasks since the embeddings themselves are trained in an unsupervised way.

Table 2.3 represents the results of k-means clustering on the low-dimensional embeddings found by different algorithms and evaluated by NMI. K-means clustering is run 50 times in order to get the average and standard deviation of accuracies. The first row in each dataset shows the mean and standard deviation of NMI and the second row shows the p-value of t-test against the best performing method on that dataset. The best performing method in each dataset is represented as boldface. The third column, "Raw Data", shows the results of clustering on the original

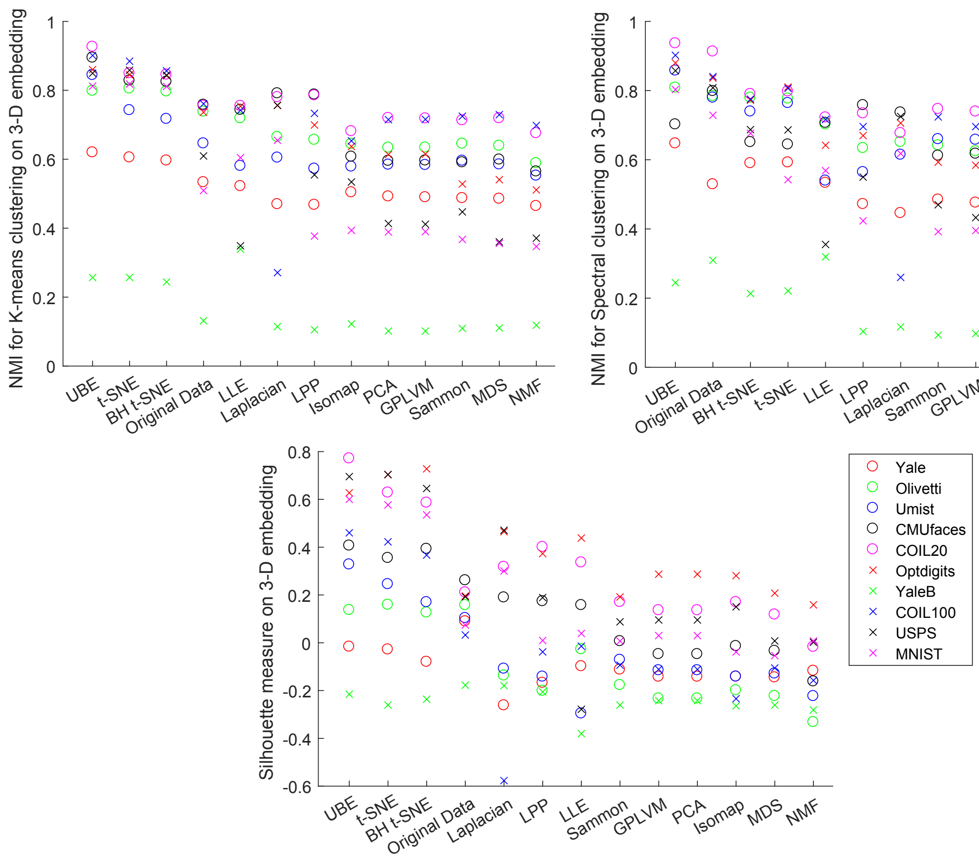


Figure 2.2: Performance of different dimensionality reduction algorithms on 10 datasets for 3-dimensional output. In this figure, only the average performance values are shown. Top left: average NMI values on 50 runs of k-means clustering. Top right: average NMI values on 50 runs of spectral clustering. Bottom: Silhouette measure of cohesion and separation between the embeddings and the true labels.

high-dimensional data. As we can see from the table, our proposed method is significantly better than all other methods in six datasets. These datasets include face recognition (Yale, Umist, CMUfaces), object recognition (COIL20, COIL100) and digit recognition (Optdigits). In Olivetti and YaleB face datasets, t-SNE and LLE are significantly better than others, respectively. In USPS and MNIST handwritten digits, our proposed algorithm and t-SNE are performing equally well. It is also noteworthy to mention that some of the embeddings such as LLE, PCA, and Sammon are usually leading to a lower quality clustering compared to the clustering on the original data. This is because of losing some valuable structure information during the transformation. However, our proposed method and t-SNE always result in a better clustering performance compared to the original data. The main reason for achieving a higher quality clustering is that they make the clusters more separable by creating large gaps between them.

Figure 2.2 illustrates the overall performance of different dimensionality reduction algorithms on 10 datasets. Each dataset is mapped to a 3-D space by different methods and evaluated by three measures: 1) NMI value for k-means clustering on low-dimensional embedding, 2) NMI value for spectral clustering on low-dimensional embedding, and 3) Silhouette measure between the 3-D embedding and the true labels. In this figure only the average performance values are shown and the dimensionality reduction methods are ordered based on their overall performance on 10 datasets. As we can see from the figure, UBE has the best overall performance in all three metrics. It preserves the neighborhood graph structure which results in retaining the clustering structure. Spectral clustering algorithms also use the idea of nearest neighbor graphs and as we can see, they work extremely well on UBE embedding output. Therefore, our proposed methods will perform exceptionally well in datasets with manifold structures since they can be well-represented by a neighborhood graph.

2.5 Visualizations of the Embedding Space

Figure 2.3 illustrates the three dimensional visualizations of UBE, t-SNE, Sammon, LPP, and PCA algorithms on different datasets. As we can see, our proposed method provides great visualizations with better cluster separability. t-SNE which is one of the state-of-the-art visualization techniques also results in good visualizations. However, our method creates a larger gap between the clusters and makes them more separable. The reason behind that is the use of Cosine kernel whose rate of decrease is lower than t-student kernel used in t-SNE. Consequently, the repulsive force applied in our algorithm pushes clusters farther away from each other compared to t-SNE.

Another observation from Figure 2.3 is that global dimensionality reduction algorithms such as PCA and Sammon cannot identify and isolate the manifolds in the embedding since they consider the global configuration of points in the mapping which makes it difficult to preserve local properties of the data.

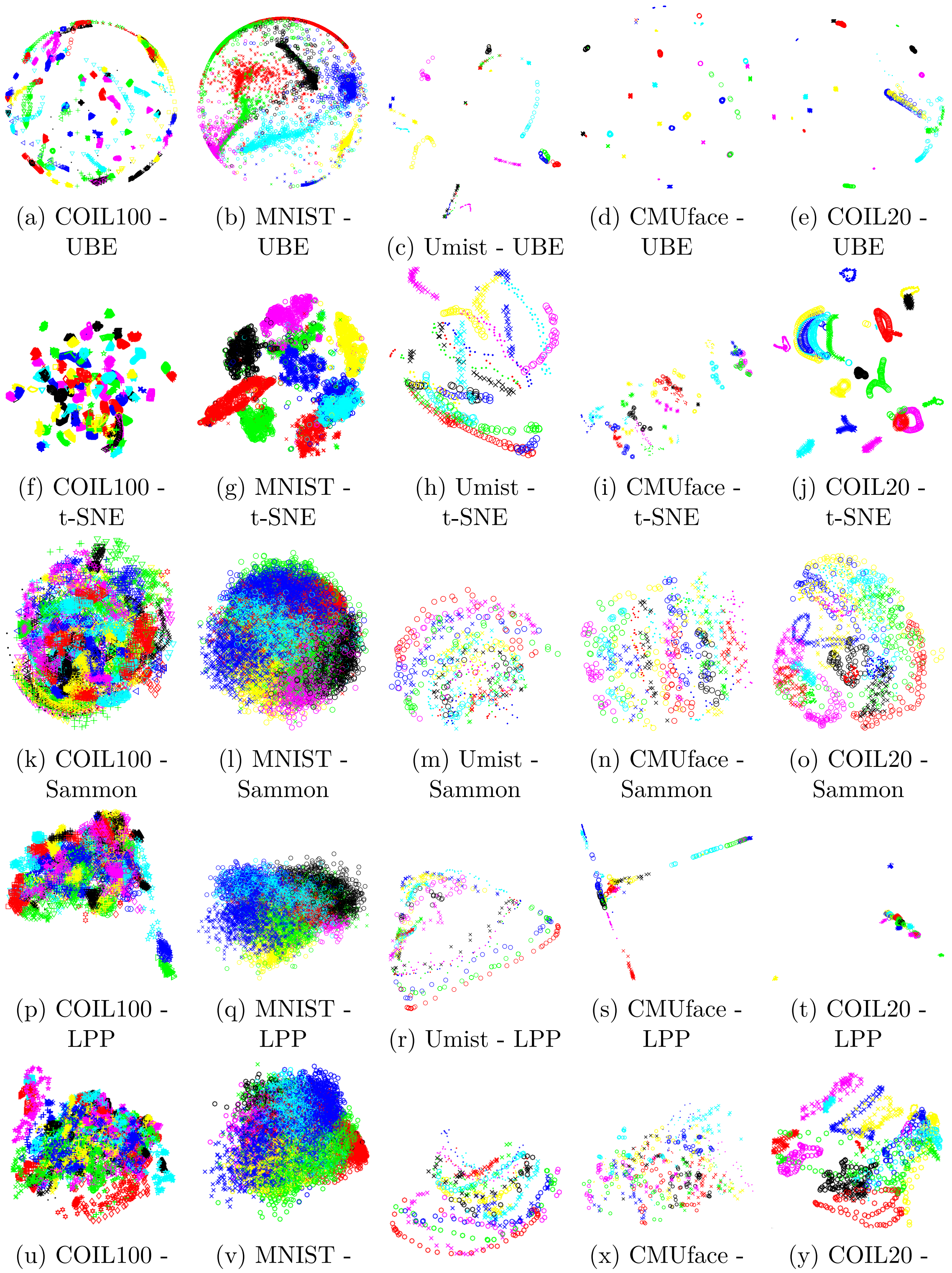
We have to mention that UBE cannot be used for visualization of multi-class datasets into 2-D space since it effectively maps the data onto a line (i.e. the circumference of the circle) and one dimension is not enough to make the manifolds separable. However, as long as $d \geq 3$, UBE will work effectively and can be used for visualization.

2.6 Conclusion

In this chapter, we presented a novel dimensionality reduction algorithm, UBE, that can be used for visualization, vector embedding applications, or as a pre-step for clustering of data. UBE builds a neighborhood graph and exploits the local structure around each point to find an embedding that preserves the structure the most. We have defined our objective function so as to minimize the squared difference between the similarities in the input and output spaces. The method uses a spherical representation for a low-dimensional embedding in which cosine metric is used to measure the similarities of points. Experiments on 10 different image clustering tasks show that our proposed method, UBE, significantly improves the quality of clustering and provides very good visualizations by maximizing the separability of clusters.

The proposed method tries to address some of the weaknesses of local dimensionality reduction methods. In particular, having many manifolds in the data is not a problem in our method. As we saw in the experiments (e.g. COIL100), if there exist many disjoint or loosely connected manifolds in the data, our methods separates them and makes enough gaps between the manifolds. Moreover, by using the elbow technique, we try to determine the right number of neighbors for each datapoint to capture the local structure.

In the next chapter, we generalize the proposed algorithm and make it applicable for the text domain and natural language processing. In particular, we modify it and use it to learn word embeddings from large corpora. We will show that the algorithm can learn high quality embeddings in any domain as long as we can calculate the input similarities in a meaningful way.



(a) COIL100 - UBE (b) MNIST - UBE (c) Umist - UBE (d) CMUface - UBE (e) COIL20 - UBE

(f) COIL100 - t-SNE (g) MNIST - t-SNE (h) Umist - t-SNE (i) CMUface - t-SNE (j) COIL20 - t-SNE

(k) COIL100 - Sammon (l) MNIST - Sammon (m) Umist - Sammon (n) CMUface - Sammon (o) COIL20 - Sammon

(p) COIL100 - LPP (q) MNIST - LPP (r) Umist - LPP (s) CMUface - LPP (t) COIL20 - LPP

(u) COIL100 - PCA (v) MNIST - PCA (w) Umist - PCA (x) CMUface - PCA (y) COIL20 - PCA

Figure 2.3: 3-D Visualizations of UBE, t-SNE, Sammon, LPP, and PCA algorithms on COIL-100, MNIST, Umist, CMUface, and COIL-20 datasets.

Chapter 3

Kernelized PMI Based Word Embedding

Continuous word representations that can capture the semantic information in the corpus are the building blocks of many natural language processing tasks. Pre-trained word embeddings are being used for sentiment analysis, text classification, question answering and so on. In this chapter, we analyze different word embedding methods and propose a new word embedding algorithm that addresses the problem of negative sampling to improve the distribution of word in the embedding space. In particular, we generalize the UBE algorithm that we proposed in previous chapter and make it applicable for learning word vectors. Our algorithm calculates the global word-word co-occurrence statistics and works on a smoothed Positive Pointwise Mutual Information (PPMI) matrix. We model the input similarities using PPMI which is known to capture word associations well. One of our major contributions is to propose an objective function and an optimization framework that exploits the full capacity of "negative examples", the unobserved or insignificant word-word co-occurrences. This function pushes unrelated words away from each other in the embedding, which improves the distribution of words in the latent space. We also propose a kernel similarity measure for the latent space that can effectively calculate the similarities in high dimensions. Moreover, we propose an approximate alternative to our algorithm using a modified Vantage Point (VP) tree [173]. This advanced data structure reduces the computational complexity of the algorithm to $|V| \log |V|$ with respect to the number of words in the vocabulary. We have trained various word embedding algorithms on articles of Wikipedia with 2.1 billion tokens, and shown that our method outperforms the state-of-the-art in most word similarity tasks by a good margin.

3.1 Introduction

Learning continuous representations of words (i.e. word embeddings) is increasingly popular in the machine learning and Natural Language Processing (NLP) community.

The goal is to learn a vector space representation for words aiming to capture semantic similarities and syntactic relationships between words. These representations are typically learned in an unsupervised mode from large, unlabeled corpora.

Classical vector space models use linear algebraic techniques on the matrix of word-word co-occurrence counts [158, 9]. These methods include Latent Semantic Analysis [38], factorizations of the co-occurrence matrix, factorizations of the Point-wise Mutual Information (PMI) and Positive PMI (PPMI) matrices, etc., which can be collectively referred to as count-based methods. GloVe [122] is among the most popular word embedding algorithms that directly work on the co-occurrence matrix. One of its main advantages over the unweighted optimization, used of the matrix factorization-based approaches, is its hand-crafted word-pair frequencies weighting optimization scheme. Another family of word embedding algorithms uses neural network based approaches to learn the word vectors. Collobert and Weston [32] proposed to learn the word embeddings using a feed-forward neural network that predicts a word by looking two words ahead and two words behind. More recently, Mikolov et al. [101] proposed log-bilinear models known as Continuous Bag-Of-Words (CBOW) and Skip-Gram to learn continuous representations of words.

Lately, contextual word representations have been shown to be effective in many NLP tasks. CoVe [99] contextualizes the word vectors using a deep attentional sequence-to-sequence model that is trained on machine translation. The authors show that using these context vectors improves performance compared to using only unsupervised word vectors. More recently, ELMo [124, 123] uses bidirectional language models (biLMs) trained on large corpus and uses the internal states of the model as contextualized word vectors. BERT [39] uses bidirectional transformers [163] where the representations are jointly conditioned on both left and right context. BERT has shown to be very successful by improving the state-of-the-art on eleven NLP tasks, surpassing human performance in some tasks.

In this chapter, we propose a new word embedding algorithm that exploits some information that other algorithms pay little or no attention to. Most word embedding algorithms only use the word pairs that occur in the corpus (i.e. positive examples) and maximize the similarity of those word vectors based on how frequent they co-occur. This can result in a *concentration effect*: word clusters from totally different topics can be placed somewhat close to each other. There are lots of possible word pairs that never co-occur in the corpus or they co-occur insignificantly, which we call negative examples. We argue that minimizing the similarity of negative examples is also crucial for the quality of the final embedding and results in a better distribution of words in the latent space. Our first major contribution is to design an optimization framework that exploits the full capacity of negative examples in order to push unrelated words away from each other, which leads to a better use of the latent space and improves the distribution of words. Skip-Gram with Negative Sampling (SGNS) [101] makes use of the negative examples to a small extent in that for each word, it randomly samples k context words as negative examples and minimizes the similarity of the word with those k context words. However, in SGNS [101], these contexts are employed in an unweighted manner in that they all have the same strength in the optimization. Our second major contribution is that we incorporate a kernelized weighting scheme for the negative examples where their influence in the optimization is proportionate to their kernel similarity with the word. We show that our kernel similarity measure is a more powerful way of calculating similarities in high-dimensional embeddings where the dimensionality d is greater than 50, and that it enables the algorithm to differentiate between the closer and further points and to employ them accordingly. Our third major contribution is that we propose a modified Vantage Point (VP) tree data structure and make it suitable for high dimensional vectors. We then use this VP-tree and propose an approximate solution to our optimization, which improves the computational complexity of the method from $|V|^2$ to $|V| \log |V|$ with respect to the size of the vocabulary.

We have trained our algorithm as well as several others on the articles of Wikipedia and compared the quality of embeddings on various word similarity and analogy tasks. Results show that our algorithm outperforms the state-of-the-art in most of the tasks.

3.2 Related Work

Distributed word representation algorithms have been shown to be very effective in capturing certain aspects of similarity between words. Many neural-network based approaches have been proposed for learning distributed word representations [16, 32, 101, 103]. Skip-Gram with Negative Sampling (SGNS) [101] uses a shallow neural network and trains the network in a way that given a word, it predicts the probability of each context word [101]. SGNS is still the state-of-the-art word embedding algorithm and is successfully applied in a variety of linguistic tasks [101, 103]. Researchers have proposed various modifications to the Skip-Gram model and tried to enrich that with other information. For instance, Levy and Goldberg [88] proposed to use the dependency parsing information and use a dependency-aware context for each word, rather than considering all the neighbors in a fixed window. This customization makes the algorithm to learn more from the syntax and less from the semantics. Recently, FastText [19], a library developed by Facebook, enriches the Skip-Gram word embeddings with sub-word information. It considers character n-grams of different lengths and represents words as the sum of their n-gram vectors. This enrichment has been shown to substantially improve the performance of NLP tasks on morphologically rich languages, such as Turkish or Finnish. It is also shown to be very effective for text classification [71]. A major advantage of the FastText is that it can handle Out-Of-Vocabulary (OOV) words by predicting their word vectors based on the learned character n-grams embeddings. Another line of work to improve the word embeddings using external resources includes retrofitting embeddings [47] to semantic lexicons such as WordNet and counter-fitting word vectors to consider antonymy and synonymy constraints [108]. However, this line of work requires external resources and is not completely unsupervised and therefore, beyond the scope of this thesis.

Pointwise Mutual Information (PMI) is an information theoretic corpus-based measure that can be used for finding collocations or associations between words [31] and is widely used in count-based and matrix factorization based word embeddings. For any word pair (w_i, w_j), PMI is defined as the log ratio between their joint probability and product of their marginal probabilities:

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \tag{3.1}$$

Based on the formulation, if two words co-occur more often in a given corpus than being independent, then their PMI will be positive, and if they co-occur less frequent than being independent then their PMI will be negative. Since the co-occurrence matrix is sparse, PMI is only calculated for the non-zero entries. Another commonly accepted approach is to use Positive PMI (PPMI) matrix by replacing all the negative values with 0.

$$PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0) \tag{3.2}$$

In fact, a traditional approach to word representation is to use explicit PPMI representation in which each word is described by its corresponding sparse row vector in the PPMI matrix. Such PPMI-based representation is shown to outperform the PMI approach on semantic similarity tasks [25]. Levy and Goldberg [89] showed that SGNS is implicitly factorizing a Shifted Positive Pointwise Mutual Information (SPPMI) matrix, and they argue that the shift parameter is almost equivalent to the negative sampling parameter k in SGNS. However, their proposed alternative approach using SVD on shifted PPMI matrix provides lower quality embeddings than SGNS, mainly because of the unweighted L_2 optimization in SVD, which causes frequent and infrequent word pairs to have the same amount of influence on the reconstruction error of the matrix. We have shown that keeping a fraction of small negative PMI values outperforms the PPMI approach [141]. Other than using the raw PPMI matrix or the factorizations of the PPMI matrix, the idea of PMI-based embedding has also been proposed and studied in the literature. Arora et al. [5] proposed an objective function similar to that of GloVe which implicitly factorizes the PMI matrix instead of the log co-occurrence matrix. They also showed that if the PMI values are good approximations of the dot products of vectors, then the linear algebraic relations (i.e. word analogies) will hold. Nevertheless, the novelty of our method is in our optimization and the effective use of negative examples.

Another research direction in learning word embeddings is to apply sparsity constraints on the word vectors [150, 148] and is shown to improve the quality of embeddings when the desired dimensionality is greater than 300. For other tips on training good quality embeddings one can refer to [81] which study the effect of various hyperparameters in different embedding algorithms.

3.3 Kernelized Unit Ball Word Embedding (KUBWE)

At a glance, our approach builds a symmetric co-occurrence matrix from the corpus, then calculates an adjusted form of the PMI matrix to remove insignificant and uninformative co-occurrences, and finally obtains the embedding by minimizing a sum of squared error between the PMI values and the cosine similarity of word vectors in the embedded space.

The intuition behind our algorithm is that if two words have a high degree of association (i.e. high PMI), their embedded word vectors must be similar (i.e. high cosine similarity), and if their degree of association is low or zero, they should not be placed close to each other. The second part of the intuition is usually ignored in other algorithms and is the main strength of our method.

Our proposed algorithm, KUBWE, is aimed to preserve the word-word connectivity structure (encoded in the PMI matrix) in the final embedding. It uses a spherical representation for the latent space in which points are located on the surface of a hypersphere. The spherical representation is not restrictive, as other algorithms also normalize the word vectors to unit length before using them in NLP tasks. Similar spherical embeddings have been proposed and used for visualization [140] and image clustering application [139].

In the following, we first propose an effective way to measure word-word associations. We then propose our objective function and its gradient descent optimization. Afterward, we propose a kernelized version of the algorithm which enhances the similarity calculations in the latent space. Eventually, we propose a modification to VP-trees to make them suitable for high dimensional data and use them to reduce the computational complexity of the method.

3.3.1 Preparing the Input for the Optimization

We first calculate the global symmetric word-word co-occurrence counts matrix X by moving an L-sized context window over the corpus. We use a weighted count strategy similar to the one used in GloVe [122] in which co-occurrences are weighted inversely proportional to the their distance in the text. This co-occurrence matrix is the main source of information for many word embedding algorithms including GloVe. We also use the same co-occurrence matrix but not in the raw format since many of the co-occurrences are meaningless.

PPMI can be used to filter out uninformative co-occurrences. However, a recognized shortcoming of PMI and PPMI is their bias towards infrequent events [158]. This happens when a rare context word w_j co-occurs with a word w_i a few times (or even once) and this often results in a high PMI value since $P(w_j)$ in PMI's denominator is very small. To overcome this situation, we smooth the distribution of context words in which all context counts are raised to the power of α. Hence, we use the following adjusted form of PPMI as input to our optimization:

$$PPMI_{\alpha}(w_i, w_j) = \max(PMI_{\alpha}(w_i, w_j), 0) \tag{3.3}$$

$$PMI_{\alpha}(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P_{\alpha}(w_j)} \tag{3.4}$$

$$P_{\alpha}(w_j) = \frac{\#(w_j)^{\alpha}}{\sum_{i=1}^{|V|} \#(w_i)^{\alpha}} \tag{3.5}$$

where $P(w_i)$ and $P_{\alpha}(w_j)$ are the unsmoothed and smoothed distribution of words. Context distribution smoothing alleviates PMI's bias towards rare words, like other smoothing techniques [119, 157]. It increases the probability of a rare context $P(w_j)$, which in turn reduces the PMI of (w_i, w_j) for any w_i co-occurring with the rare context w_j.

We refer to the adjusted PPMI matrix as A with entries $a_{ij} = PPMI_{\alpha}(w_i, w_j)$. In all our experiments, we used $\alpha = 0.75$ which is known to be a good smoothing factor [103].

3.3.2 Cost Function

Our proposed method uses a spherical representation for the latent space, which is common for natural language processing tasks. In this representation, we use cosine similarity as our similarity measure between word vectors:

$$S(\vec{w_i}, \vec{w_j}) = \frac{\vec{w_i}.\vec{w_j}}{\|\vec{w_i}\|\|\vec{w_j}\|} \tag{3.6}$$

The objective function is then defined to minimize the sum of squared differences between the smoothed PPMI values a_{ij} and the similarities in the embedded space:

$$\mathcal{J}(W) = \frac{1}{2} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \big(a_{ij} - S(\vec{w_i}, \vec{w_j})\big)^2 \tag{3.7}$$

where a_{ij} and $S(\vec{w_i}, \vec{w_j})$ are calculated using equations (3.3) and (3.6), respectively. Here we propose a constrained variant of the objective function which makes the normalizations fade away, as well as resulting in a simplified gradient:

$$\mathcal{J}(W) = \frac{1}{2}\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}\big(a_{ij} - \vec{w}_i.\vec{w}_j\big)^2$$
$$subject\ to : \vec{w}_i.\vec{w}_i = 1\ \forall i \tag{3.8}$$

Here, the objective function forces the word vectors to be on the surface of a hypersphere. The unit length constraints is not restrictive, since the surface of a hypersphere has only one fewer degree of freedom than the entire volume. Therefore, the surface will have enough room to locate all the points in the desired way. In the constrained form of the objective function, the dot product of output vectors is the cosine similarity and is used as the similarity measure between word vectors in the embedded space.

Cosine values range in $[-1, +1]$ and we normalize them to $[0, +1]$ by using $\frac{1}{2}(\vec{w}_i.\vec{w}_j + 1)$. This is done because a_{ij} values are also positive. In fact, PMI values a_{ij} can be greater than 1, however, we handle this in the next section. By normalizing the cosine values to $[0, +1]$, if the PMI of two words is high (i.e. they co-occur to a significant degree), then the method tries to place them as close as possible. And if the PMI is zero (i.e. the co-occurrence is unobserved or insignificant), $a_{ij} = 0$, then the method tries to put them as far as possible.

3.3.3 Optimization

We use stochastic gradient descent to minimize our objective function. We initialize the word vectors randomly on the surface of a hypersphere and incrementally change the configuration of them to get a better objective value. We have used the general projected gradient descent framework [91] in order to satisfy the constraints. In this setting, the solution is projected onto the feasible region after each iteration. The gradient of the cost function with respect to a word vector $\vec{w}_i$ is:

$$\nabla\mathcal{J}(\vec{w}_i) = \frac{\partial\mathcal{J}}{\partial\vec{w}_i} = \sum_{j=1}^{|V|}\Big[-a_{ij}\vec{w}_j + (\vec{w}_i.\vec{w}_j)\vec{w}_j\Big]$$
$$= \overbrace{-\sum_{j=1}^{|V|}a_{ij}\vec{w}_j}^{v_1} + \overbrace{\sum_{j=1}^{|V|}(\vec{w}_i.\vec{w}_j)\vec{w}_j}^{v_2} \tag{3.9}$$

The gradient consists of two components v_1 and v_2. The former, v_1, defines the sum of attractive forces being applied to the word vector while the latter, v_2, defines the sum of repulsive forces being applied to the word vector. In fact, each context word is having a contribution in the gradient of $\vec{w_i}$. Since W is a sparse affinity matrix, only the non-zero PMIs will have a contribution in the attractive force v_1. However, all other words vectors will have a contribution to the repulsive force v_2 based on their similarity to the word vector being tuned. In other words, $\vec{w_i}$ will be attracted to its significant context vectors and it will be pushed away from its current neighbors if it shouldn't be close to them. Please note that, by normalizing the similarities in [0, +1], v_2 will be the sum of weighted repulsive forces applied from different word vectors.

One of the distinguishing characteristics of our method is that unlike SGNS and GloVe which update a word vector based on each entry in the matrix (or each occurrence of two words in the moving window), our algorithm updates the word vector $\vec{w_i}$ based on its entire row in the smoothed PMI matrix A. This way, all the attractive forces in v_1 are automatically weighted according to their smoothed PMI value. Therefore, using an auxiliary weighting function as in GloVe is totally unnecessary and here, the weighting is done seamlessly. As for the negative force v_2, we take out the non-zero PMIs and calculate the negative force only based on the word pairs with zero PMI. This slightly improves the distribution of words, as each context will have either an attractive or repulsive effect when updating a particular word vector.

The time complexity for calculating the gradient vector for a particular word $\vec{w_i}$ is $O(|V| \times d)$ where $|V|$ and d are the vocabulary size and the dimensionality of the embedded space, respectively. The calculation of the repulsive force is more costly than the attractive force because of the similarity computations. In total, the optimization requires $O(i \times |V|^2 \times d)$ time, where i is the number of iterations in the optimization. However, the size of the vocabulary $|V|$ is several orders of magnitude less than the number of tokens in the corpus. For instance, in our experiments on Wikipedia (dump of March 2016), 163,188 words were extracted from 2.1 billion tokens. Later, we will adopt an approximation technique to reduce the time complexity.

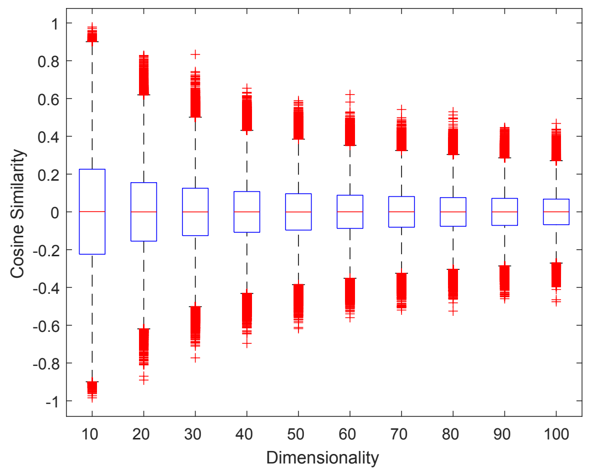


Figure 3.1: Distribution of cosine similarity of 100,000 pairs of random vectors. The distribution of cosine similarities is $\mathcal{N}(0, \frac{1}{\sqrt{d}})$.

3.3.4 Kernelized Objective Function

Figure 3.1 illustrates the distribution of cosine similarities between thousands of random vectors in different dimensionality. We generated 100,000 pairs of random vectors in each case and calculated their cosine similarity and plotted the distribution of similarities with respect to the dimensionality of vectors. As we can see from the figure, in lower dimensions we have a wider distribution between [-1, +1]. But, as we increase the dimensionality, the distribution narrows down and the chances of getting two similar or dissimilar vectors is getting lower and lower. In fact, in higher dimensions, almost all vectors will be equidistant and almost orthogonal to each other. Specialized distance measures have been proposed in the literature [143], but the curse of dimensionality in metric spaces is not well-studied.

Considering ineffectiveness of cosine similarity (and every other metric) in higher dimensions, we propose a kernelized variant of our objective function which improves the distribution of similarities and enables the algorithm to differentiate between the closer and further vectors.

Looking at Equation 3.9, the repulsive force v_2 consists of a dot product of vectors in the embedded space which measures their similarity. Here, we can apply a kernel to calculate the similarities of vectors in an implicit high-dimensional feature space. If $\phi(\cdot)$ is the implicit mapping function to the high-dimensional space, the kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ will compute the inner product of those vectors in an efficient way $K(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle$. The gradient of the objective function is then:

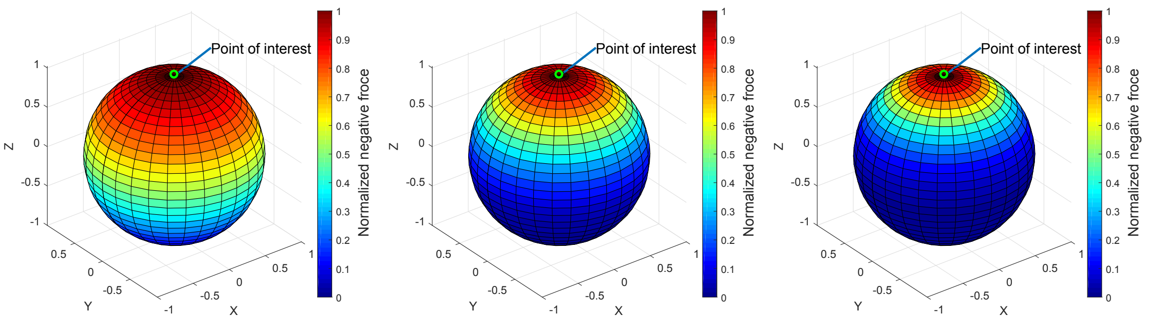
$$\frac{\partial \mathcal{J}}{\partial \mathbf{y}_i} = -\sum_{j=1}^{n} w_{ij}\mathbf{y}_j + \sum_{j=1}^{n} K(\mathbf{y}_i, \mathbf{y}_j) \times \mathbf{y}_j \tag{3.10}$$

Here we apply the kernel just in the repulsive force and not in the objective function directly. PMI has proven to be able to capture the strength of association of words very well [25, 31]. Therefore, we do not need to adjust the attractive force and we only want to tune the similarities in the embedded space. Moreover, applying the kernel in the repulsive force will simplify the formulation and consequently, simplify the numerical optimization by preventing the derivative of $\phi(.)$ to appear in the gradient. Here, we propose to use a polynomial kernel to adjust the nonlinearity and further strengthen the effect of closer points in negative force.

$$K(\vec{w_i}, \vec{w_j}) = (\vec{w_i}.\vec{w_j} + 1)^p \tag{3.11}$$

where p is the degree of the polynomial kernel. Please note that in the kernelized form, there is no need to normalize the dot product between [0, +1] because of the increment in the formula which makes all similarities positive. In fact, by using a polynomial kernel $p \geq 2$, the negative cosine similarities are weakened while the positive cosine similarities are strengthened. This provides a more powerful similarity measure for higher dimensions and more discriminative power for our learning algorithm.

We have illustrated the effect of the polynomial kernel in adjusting the repulsive force in Figure 3.2. Figures 3.2a, 3.2b, and 3.2c depict the effects of linear kernel ($p = 1$), degree 3 polynomial, and degree 5 polynomial, respectively. Considering the word being updated at any given time at the top of the sphere, the colors represent the strength of negative force from other words that is dependent to their distance, farther the other words lesser the negative force associated to them. The kernel function sharpens this effect by adjusting the similarities in the embedding. Please note that this is only a 3D illustration to clarify the impact of kernel, however, in high-dimensional embedding spaces where $d \geq 100$, the outcome is a bit different. In a high-dimensional embedding space, if we assume the current word at the top again, then almost all other words will be close to the equator of the hypersphere. In this case, the kernel strengthens the negative force from the ones on the northern hemisphere and weakens the force from the ones on the southern hemisphere. Consequently,



(a) Linear kernel ($p = 1$) (b) Degree 3 polynomial (c) Degree 5 polynomial

Figure 3.2: The effect of polynomial kernel degree in the repulsive force in KUBWE algorithm: (a) linear kernel ($p = 1$), (b) degree 3 polynomial, and (c) degree 5 polynomial. Considering the word being updated at the top of the sphere, the colors represent the strength of negative force from other words that is dependent to their distance.

this leads to a more effective approximation of forces in the optimization.

3.3.5 Reducing the time complexity by approximating the repulsive force

The most time consuming part of our algorithm is the calculation of repulsive force that requires all the pairwise similarity calculations. Here, we propose an alternative, efficient solution that reduces the complexity from $|V|^2$ to $|V| \log |V|$ with respect to the size of the vocabulary. In this fast version of the algorithm we only take into account the k nearest neighbors of each word for calculating the repulsive force v_2 in Equation 3.9. This way, each word w_i attracts words w_j if $a_{ij} > 0$ (i.e. positive PMI) and pushes away words w_j for which $a_{ij} = 0$ that are among its nearest neighbors $w_j \in knn(w_i)$. In fact, we do not need to push words further away if they are already far apart, but if they are mistakenly close to the word, we use the repulsive force to improve the distribution of words.

Nearest neighbor search has a long history and is well-studied in the literature. *kd*-trees [49] are the most recognized method for fast nearest neighbor search. They hierarchically partition the space into two axis parallel regions where the median value in the splitting axis is used as the splitting threshold. The search is done by traversing the tree from the root to a leaf node. Some extra checking is also needed to see whether there could be any points on the other sides of the splitting planes.

The main drawback of *kd*-trees is their inefficiency when used with high-dimensional data. As a general rule, if the dimensionality of the data is d, then the number of points in the data, N, should be $N > 2^d$. Otherwise, most of the branches in the tree will be visited and the efficiency is no better than exhaustive search [154].

Another approach for nearest neighbor search is to use a Vantage Point tree (VP-tree) [173]. A VP-tree is a binary tree that is hierarchically built by randomly selecting a point as a vantage point and calculating the distance from the vantage point to every other point. Then using the median distance as the splitting threshold, half of the points fall under the left child (closer than median) and the other half fall under the right child (farther than median). The splitting process for each node using the median distance can be thought of as centering a hyper-sphere on the vantage point in a way that half of the point are inside the hyper-sphere, and the other half are outside of it. The main advantage of VP-trees is that they do not depend on the distribution of the data and they only work based on the distances.

VP-trees [173] along with *kd*-trees [49] and almost any other data structure suffer from the curse of dimensionality and are only suitable for low-dimensional data. The problem arises due to the ineffectiveness of distance measures in high dimensions [1]. In fact, in high dimensions all the points will be almost equidistant and it is hard to distinguish between the closest and farthest points. Therefore, while searching a query point in a VP-tree, if it falls within the hypersphere of a vantage point we still need to check the outside points since the query will be very close to the median distance and the nearest neighbor may be on the other side. And similarly if the query point falls outside of the hypersphere of the vantage point we still need to check the inside points. Consequently, this leads to traversing the entire tree and its performance will be equivalent to that of exhaustive search.

There exist many approximate nearest neighbor search methods as summarized in [109]. However, due to the nature of our problem and its known structure we propose our own alternative. We use a modified Vantage Point tree (VP-tree) [173] combined with a heap data structure to calculate and maintain the nearest neighbors of each word. In our algorithm, all the words are distributed on the surface of a hypersphere, and therefore their dot product is equivalent to their cosine similarity.

Cosine similarity of vectors in d dimensions has a distribution of $\mathcal{N}(0, \frac{1}{\sqrt{d}})$ [146]. Similarly, the cosine distances are distributed from $\mathcal{N}(1, \frac{1}{\sqrt{d}})$. For instance, considering $d = 100$, then the cosine distances will be in [0.7, 1.3] range with 99.7 probability (i.e. $\mu \pm 3\sigma$). We incorporate this information in our VP-tree search in order to decide whether or not the other branch needs visiting. Using this technique we get more than 99% accuracy on our k nearest neighbor search, while ensuring the $\log |V|$ search time for each word.

By adopting the aforementioned technique, the time complexity of our algorithm is $O(id|V|(\bar{p}+(k+\log|V|)\log\bar{p}))$ where $\bar{p}$ is the average number of positive examples per word ($\bar{p} \ll |V|$) and k is the number of nearest neighbors (i.e. negative examples) that is used. $O(id|V|\bar{p})$ corresponds to the attractive force calculations, while $O(id|V|(k + \log|V|)\log\bar{p})$ corresponds to the repulsive force calculations. $\log\bar{p}$ correspond to the binary search inside the positive indices to ensure that the negative set does not overlap with the positives. Algorithm 2 describes the search mechanism in our high-dimensional VP-tree algorithm. The tree construction procedure is identical to the original VP-tree. However, in the search time we calculate d_{min} as the minimum possible distance between words, and use it to eliminate unnecessary branch visits. The variable τ represents the furthest neighbors' distance found so far and is initially set to infinity.

KUBWE is implemented in C using the OpenMP parallel computing library and the source code can be found on GitHub[1].

3.4 Experiments

We have used all the articles of English Wikipedia (dump of March 2016) as the training corpus, which has around 2.1 billion tokens after applying a few basic pre-processing steps. As for the vocabulary, we have limited the vocabulary to English words by using the WordNet database which resulted in about 163K words. In our experiments, all the algorithms were trained on the exact same preprocessed input corpus to ensure a fair comparison. We have also used the exact same vocabulary for all the algorithms.

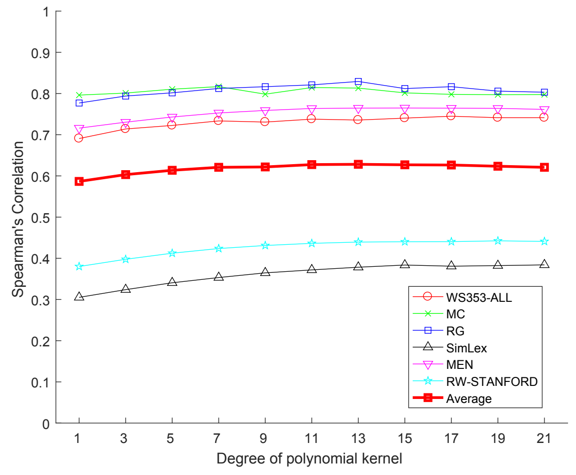
For the quantitative evaluation of algorithms we have used two well-known tasks

[1]`https://github.com/behrouzhs/kubwe`

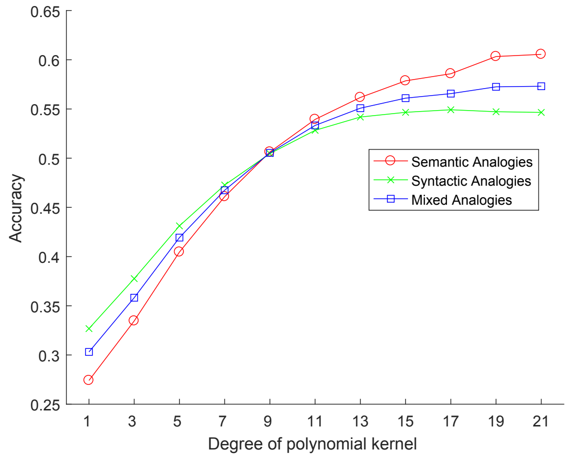
Algorithm 2 VPtreeSearch — modified Vantage Point tree search

```
**Input:** node, heap, query, positiveIndices, $k$, $\tau$, $d$
**Output:** heap, $\tau$      //$\tau$ is the furthest neighbors' distance found so far
$d_{min} = 1 - \frac{3}{\sqrt{d}}$
$dist$ = CosineDistance(node.data, query.data)
if node.idx != query.idx && BinarySearch(node.idx, positiveIndices) is False then
    if heap.count $< k$ then
        heap.Push($dist$, node.idx)
        if heap.count $= k$ then
            $\tau$ = heap.top.dist
        end if
    else if $dist < \tau$ then
        heap.Pop()
        heap.Push($dist$, node.idx)
    end if
end if
if node.left = NULL && node.right = NULL then
    return heap, $\tau$
end if
if $dist <$ node.medianThreshold then
    heap, $\tau$ = VPtreeSearch(node.left, heap, query, positiveIndices, $k$, $\tau$, $d_{min}$)
    if $dist + \tau - d_{min} \geq$ node.medianThreshold then
        heap, $\tau$ = VPtreeSearch(node.right, heap, query, positiveIndices, $k$, $\tau$, $d_{min}$)
    end if
else
    heap, $\tau$ = VPtreeSearch(node.right, heap, query, positiveIndices, $k$, $\tau$, $d_{min}$)
    if $dist - \tau + d_{min} \leq$ node.medianThreshold then
        heap, $\tau$ = VPtreeSearch(node.left, heap, query, positiveIndices, $k$, $\tau$, $d_{min}$)
    end if
end if
```

of word similarity and word analogy. For the word similarity task, there exist several datasets containing word pairs with their corresponding human-assigned similarity

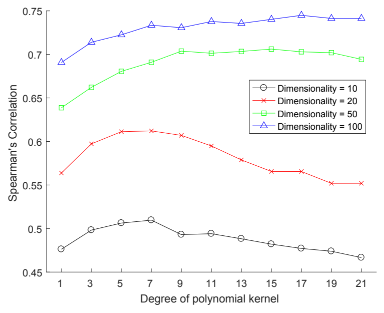


(a) Word similarity task

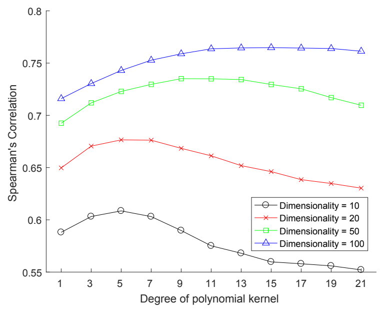


(b) Word analogy task

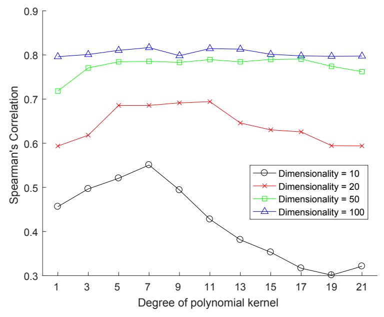
Figure 3.3: Quality of embeddings ($d = 100$) obtained from KUBWE using different kernel degrees measured on different (a) word similarity, and (b) word analogy tasks.



(a) WordSim353



(b) MEN



(c) MC

Figure 3.4: The effect of polynomial kernel degree in different dimensionality in KUBWE algorithm evaluated on (a) WordSim353, (b) MEN, (c) MC.

score. In this task we have used 8 different dataset including *WordSim353* (WS-ALL), *WordSim Similarity* (WS-SIM) and *WordSim Relatedness* (WS-REL), *MEN*, *SimLex*, *MC*, *RG*, and *Stanford Rare Words* (RW-STN). For the analogy task, we have used Google's analogy dataset [101] which contains 19,544 questions of the form "a is to a^* as b is to b^*". Given three of the words, the algorithm is expected to predict the fourth word. About half of the questions are semantic (e.g. "*father* is to *son* as *mother* is to *daughter*") and the other half are syntactic questions (e.g. "*big* is to *bigger* as *tall* is to *taller*").

3.4.1 Analysis of the Polynomial Kernel Degree

We first analyze the effect of the kernel degree in a fixed dimensionality of 100. Our algorithm is trained using various polynomial degrees $1, 3, \ldots, 21$ and the quality of embeddings is measured on different word similarity and word analogy tasks. Figure 3.3 shows the performances with respect to the kernel degree. As we can see from Figures 3.3a and 3.3b the general trend is increasing as we increase the degree of the polynomial. This shows that in 100-dimensional space using a high degree polynomial kernel significantly improves the distribution of words, nonetheless, it reaches a plateau at some point.

In another experiment, we run the algorithm with different embedding dimensionality (10, 20, 50, and 100) and in each case, we use various kernel degrees $1, 3, \ldots, 21$. Figure 3.4 illustrates the performance of our algorithm on different dimensionality using different kernel degrees. As we can see from the figures, generally we get better embedding as we increase the dimensionality of the embedding. This is true in other algorithms as well. We can also observe that in lower dimensions (10 and 20), using a high degree kernel will degrade the quality of embedding significantly. This is because of distribution of cosine similarities (i.e. dot products) in the first place. If we look back at Figure 3.1 we see that when $d = 10$ and $d = 20$ the similarities are spread all over [-1, +1]. Using a high degree polynomial in such cases causes a few negative examples to have extremely high kernel similarity with the word being updated and they dominate all the rest of the negative examples. This leads to inappropriate use of negative examples which in turn deteriorates the quality of the embedding. However, in higher dimensions where the distributions of similarities are close to zero (almost orthogonal vectors), using a higher degree polynomial will further improve the similarity calculations in the repulsive force.

3.4.2 Quantitative Evaluation

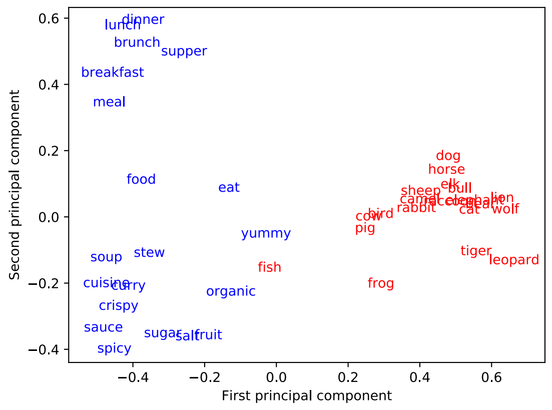
Table 3.1 compares 14 algorithms on 8 word similarity datasets. The numbers in the table are Pearson's correlation between the rankings provided by the algorithms and the rankings of the human-scoring. These algorithms are selected mainly because of their popularity and the reproducibility/availability of their source code. SVD, SVD-Log, and SVD-Sqrt are the factorizations of the co-occurrence, the log

Table 3.1: Evaluation of different word embedding algorithms on 8 word similarity datasets. The dimensionality of the embeddings is 100 for the top part and 300 for the bottom 5 rows. Numbers in the table are Pearson's rank-order correlation between the human scores and scores from algorithms.

	WS-SIM	WS-REL	WS-ALL	MC	RG	MEN	SimLex	RW-STN
# of word pairs	203	252	353	30	65	3000	999	2034
SVD	0.533	0.282	0.410	0.331	0.491	0.390	0.202	0.229
SVD-Sqrt	0.754	0.605	0.681	0.729	0.667	0.657	0.286	0.395
SVD-Log	0.741	0.629	0.699	0.783	0.693	0.712	0.328	0.386
SVD-PPMI	0.720	0.638	0.692	0.803	0.740	0.740	0.318	0.381
SVD-SPPMI	0.669	0.593	0.646	0.774	0.709	0.716	0.297	0.360
GloVe ($x_m = 100$)	0.674	0.553	0.599	0.664	0.706	0.704	0.315	0.329
CBOW ($k = 10$)	0.740	0.584	0.665	0.703	0.756	0.709	0.326	0.393
CBOW ($k = 5$)	0.745	0.585	0.671	0.742	0.773	0.707	0.327	0.398
FastText	0.765	0.649	0.713	0.793	0.787	0.741	0.326	0.442
SGNS ($k = 10$)	**0.774**	0.650	0.712	0.801	0.789	0.732	0.324	0.423
SGNS ($k = 5$)	0.758	0.651	0.709	0.794	0.783	0.730	0.323	0.421
Fast KUBWE	0.746	0.663	0.728	0.805	0.807	0.735	0.367	**0.444**
KUBWE ($p = 13$)	0.770	**0.692**	**0.740**	**0.809**	**0.827**	**0.761**	**0.376**	0.439
GloVe ($x_m = 100$)	0.695	0.572	0.621	0.749	0.744	0.726	0.354	0.353
FastText	**0.794**	0.669	0.733	0.821	0.805	0.766	0.381	**0.483**
SGNS ($k = 10$)	0.792	0.667	0.732	0.830	0.799	0.753	0.383	0.455
Fast KUBWE	0.781	0.704	0.753	0.828	0.836	0.765	**0.421**	0.451
KUBWE ($p = 13$)	0.783	**0.710**	**0.759**	**0.851**	**0.845**	**0.775**	0.411	0.452

co-occurrence, and the square root of co-occurrence matrices, respectively. SVD-PPMI and SVD-SPPMI are the SVD factorization of the PPMI and Shifted PPMI (with shift parameter of $-\log 5$) matrices, respectively. SVD-NS is the factorization of thresholded PMI table which incorporates a fraction of negative PMI values [141]. GloVe is trained with its recommended parameter setting (i.e. $x_{max} = 100$). FastText is trained with the recommended parameter settings that considers character n-grams of length 3 to 6. CBOW and SGNS are trained with negative sampling set to 5 and 10. Our proposed algorithm, KUBWE, is trained with $p = 13$, and the fast KUBWE is trained with $k = 3000$. The dimensionality of embeddings is 100 in the top part of the table and 300 in the bottom 5 rows.

As we can see from Table 3.1, our algorithm provides the best results on 7 out of 8 datasets using 100-dimensional embeddings and on 6 out of 8 datasets using 300-dimensional embeddings. It is noteworthy to mention that even the fast approximate version of KUBWE outperforms the state-of-the-art in 6 out of 8 word similarity tasks. Using 100-dimensional embeddings, SGNS is the best on only WordSim Similarity dataset, and on the rest of the datasets, our method outperforms others by


(a) KUBWE

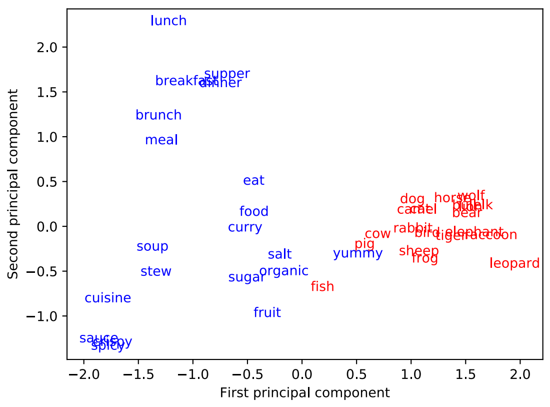

(b) SGNS

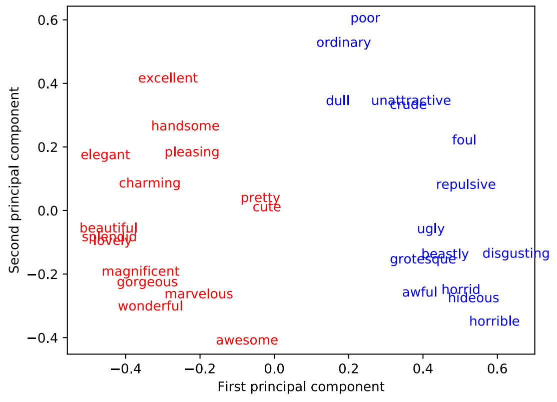
Figure 3.5: Distribution of 40 word vectors from two groups of 20 animal names and 20 food related words. PCA algorithm is applied on 100-dimensional vectors from KUBWE (left) and SGNS (right) to obtain a 2-d visualization.

a good margin. We have to mention that in the analogy task GloVe provides the best results and is better than KUBWE and SGNS. However, GloVe's performance on word similarity tasks is not comparable with that of KUBWE and SGNS.

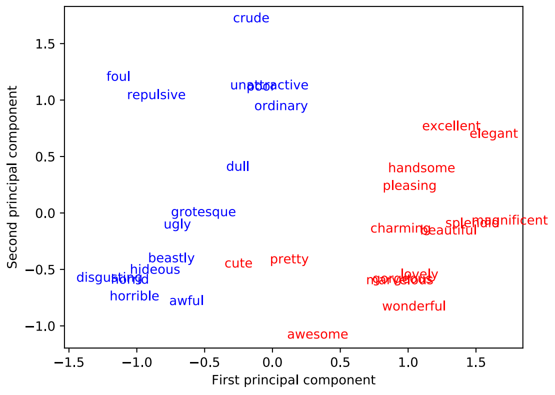
3.4.3 Qualitative Evaluation

Figure 3.5 illustrates the 2-d distribution of 40 word vectors (20 animal names and 20 food related words) obtained by applying Principal Components Analysis (PCA) on the resulting embedding vectors from KUBWE and SGNS. As we can see, our algorithm provides a better separation and a larger gap between the two word clusters by pushing unrelated words away from each other which is the consequence of the repulsive force and better utilization of negative examples. Moreover, in the SGNS distribution "yummy" is closer to the animal group which is not correct, and "salt" and "organic" are also very close to the boundary.

Figure 3.6 illustrates the 2-d distribution of 30 word vectors (15 positive and 15 negative adjectives) obtained by applying PCA on KUBWE and SGNS embeddings. Again we can see that our algorithm provides a better semantic distribution in the latent space with a much clearer separation between antonym word clusters. In the SGNS embedding, "pretty" and "cute" are quite close to the negative adjectives such as "beastly" and "aweful".



(a) KUBWE



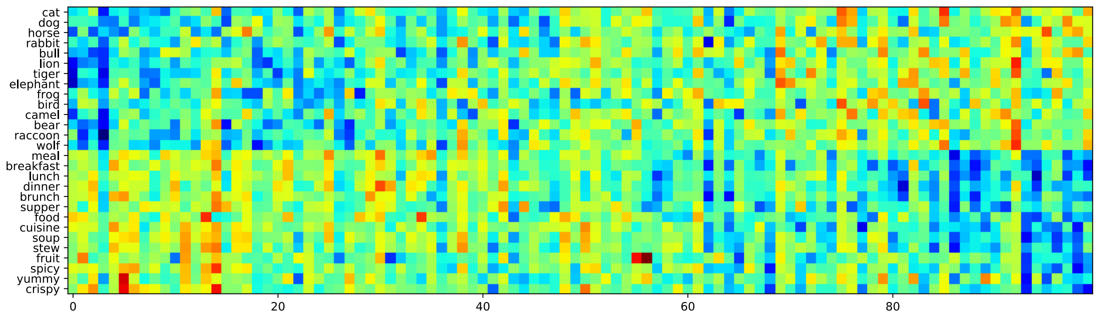
(b) SGNS

Figure 3.6: Distribution of 30 word vectors (15 positive and 15 negative adjectives). PCA algorithm is applied on 100-dimensional vectors from KUBWE (left) and SGNS (right) to obtain a 2-d distribution.

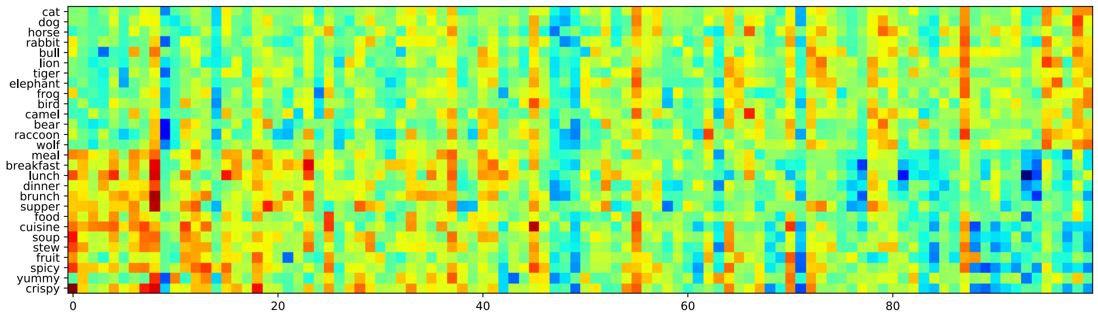
Figure 3.7 shows the heatmaps of 28 word vectors (14 animal names at the top and 14 food related words at the bottom) from the KUBWE and SGNS embeddings. In each of the heatmaps, columns are ordered by the difference between the average magnitude of the features among the two word groups. Here we can see that our algorithm provides a better inter-group dissimilarity and a nicer distinction between two unrelated word groups. This property will potentially improve the accuracy of classifiers in many NLP tasks including text classification.

3.5 Conclusion

In this chapter, we analyzed different word embedding algorithms and proposed our algorithm KUBWE. Our method has clear advantages over matrix factorization methods, since the attractive and repulsive forces in the optimization are weighted according to the similarities in the input (i.e. smoothed PPMI) and similarities in the output (i.e. polynomial kernel), respectively. It has also advantages over “prediction-based” methods such as SGNS and GloVe for two main reasons: 1) The smoothed PPMI input to our algorithm is more reliable than the raw co-occurrence counts. 2) The adaptive way of utilizing the negative examples prevents the concentration effect and improves the distribution of words in the final embedding. Moreover, the effect of cosine similarity in higher dimensions is analyzed and a kernelized way of calculating



(a) KUBWE



(b) SGNS

Figure 3.7: Heatmap of 28 word vectors obtained from KUBWE (top) and SGNS (bottom). The top 14 rows in the heatmaps are animal names and the bottom 14 rows are food related words.

similarities is suggested to alleviate the ineffectiveness of cosine similarity. Furthermore, by adopting a modified Vantage Point-tree and approximating the repulsive force in the optimization we reduced the computational complexity of our algorithm by orders of magnitude. Our algorithm has only one parameter, which is the polynomial degree p in the exact version, and the number of negative neighbors k in the fast approximate version. As a rule of thumb, one should pick the degree proportionate to the log of embedding dimensionality $p \approx \log d$, and the number of negative neighbors roughly equal to the square root of the number of words in the vocabulary $k \approx \sqrt{|V|}$. In the next chapter, we focus on spectral methods for word embedding and propose an alternative spectral word embedding that takes into account the notion of negative examples.

Chapter 4

EigenWord and Spectral Word Embeddings

In this chapter, we investigate word embedding algorithms from a different perspective. In particular, we examine the notion of “negative examples”, the unobserved or insignificant word-context co-occurrences, from the perspective of the spectral methods. We provide a new formulation for the word embedding problem by proposing a new intuitive objective function that perfectly justifies the use of negative examples. In fact, our algorithm not only learns from the important word-context co-occurrences, but also it learns from the abundance of unobserved or insignificant co-occurrences to improve the distribution of words in the latent embedded space. We analyze the algorithm theoretically and provide an optimal closed-form solution for the problem using spectral analysis. We have trained various word embedding algorithms on articles of Wikipedia with 2.1 billion tokens and show that negative sampling can boost the quality of spectral methods. Our algorithm provides results as good as the state-of-the-art but in a much faster and efficient way.

4.1 Introduction

In recent years there has been an increasing interest in learning compact representations (i.e embeddings) for a set of input datapoints. In these approaches, input data is mapped to a low-dimensional latent space with the goal of preserving the geometrical properties of data with respect to some similarity measure in the input space. That is, similar datapoints in the input space should be mapped to nearby points in the latent embedded space. In the embedded space, each input datapoint is described with a dense d-dimensional continuous-valued vector representing the coordinates of the datapoint in the latent space.

In natural language processing, embedding algorithms are used to learn a vector space representation for words aiming to capture semantic similarities and syntactic relationships between words. The traditional way of treating individual words as unique symbols and representing documents by sparse word count vectors, known as Bag-of-Words (BOW) representation [133], has strong limitations since it does not exploit countless semantic and syntactic relations encoded in the corpus. Moreover, it does not take into account the ordering of the words, therefore, two different sentences can have the same representation as long as they use the same words. n-gram models [149, 23, 50] tried to overcome these limitations by counting the occurrences of sequences of words rather than individual words ($1 \leq n \leq 5$). They consider the word order in a short context but suffer from the curse of dimensionality, since the number of n-grams increases dramatically as n increases. They also do not capture the semantics of the words or more formally the similarities between the words [82].

In recent years, distributed word representations or word embedding algorithms have shown to be very effective in capturing certain aspects of similarity between words. Statistical language modeling methods take advantage of the fact that words that are temporally closer in a sentence are statistically more dependent, and therefore, model the probability of a word conditioned on the previous words in the sentence [14]. The idea of using a moving context window and assuming words in the same window are semantically similar is widely exploited [103, 101, 122]. GloVe calculates the global co-occurrence statistics first using a fixed-size context window, and then minimizes its least squares objective function using stochastic gradient descent which is essentially factorizing the log co-occurrence matrix [122]. Many neural-network based approaches have been proposed for learning distributed word representations [16, 32, 101, 103]. Skip-Gram with Negative Sampling (SGNS) is still the state-of-the-art word embedding algorithm and is successfully applied in a variety of linguistic tasks [101, 103].

Levy et al. in [90] have studied the effect of various hyper-parameters in different embedding algorithms and showed that many of these parameters can be transferred to traditional methods (e.g SVD) to boost their performance. In fact, they showed that explicit matrix factorization methods can provide competitive results if used properly, and there is no significant advantage over any of the algorithms. Levy and Goldberg in [89] showed that SGNS is implicitly factorizing a Shifted Positive Point-wise Mutual Information (SPPMI) matrix and they argue that the shift parameter is almost equivalent to the negative sampling parameter k in SGNS. However, their proposed alternative approach using SVD provides lower quality embedding than SGNS, mainly for two reasons: first, the unweighted L_2 optimization in SVD which gives equal importance to frequent and infrequent pairs, and second, the shift cannot capture certain aspects of the parameter k in SGNS: higher k in SGNS results in using more data and better estimating the distribution of negative examples [90]. Therefore, SGNS remains superior to others with a small margin in most NLP applications.

In this work, we provide a different perspective for looking at the negative samples, word pairs that never co-occur. Most embedding algorithms only use the word pairs that occur in the corpus and maximize the similarity of those word vectors based on how frequent they co-occur. This can result in *concentration effect*: word clusters from totally different topics can be placed somewhat close to each other. There are lots of possible word pairs that never co-occur in the corpus or they co-occur insignificantly, which we call them negative examples. We argue that minimizing the similarity of negative examples is also crucial in the quality of final embedding, and results in better distribution of words in the latent space. We show how matrix factorization methods can benefit from the abundance of information (i.e. negative examples) which was disregarded previously since they were considered useless. We incorporate the notion of negative sampling in standard matrix factorization methods by randomly choosing a tiny fraction of zeros in the PMI matrix and assigning negative values to them or simply by not ignoring all negative PMI values. We formulate the problem as an optimization task by proposing an intuitive objective function which perfectly justifies the use of negative values in the PMI matrix. Our optimization has an optimal closed-form solution and we make a theoretical connection between our solution and SVD.

4.2 Background

4.2.1 Notation

Let's assume we have a text corpus which is a sequence of words $w \in V_W$ where V_W is the word vocabulary. The context of a word w_i is commonly defined as the words surrounding it in a window of size L, $w_{i-L}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+L}$. This results in a set of context words $c \in V_C$ with their corresponding context vocabulary V_C. The sizes of the vocabularies are typically in $10^5 - 10^6$ range.

For each word-context pair (w, c), we use $\#(w, c)$ to denote the number of times they co-occurred in the corpus. Similarly, we use $\#(w) = \sum_{i=1}^{|V_C|} \#(w, c_i)$ and $\#(c) = \sum_{i=1}^{|V_W|} \#(w_i, c)$ for denoting the number of times w and c occurred in the corpus, respectively.

By moving an L-sized context window over the corpus, a global co-occurrence matrix X of size $|V_W| \times |V_C|$ can be built where each matrix entry $x_{ij} = \#(w_i, c_j)$ denote the number of times w_i and c_j co-occurred. The co-occurrence matrix is a very sparse matrix with lots of zero entries since most possible word-context pairs never co-occur in the corpus.

Most word embedding algorithms embed all words and contexts in a d-dimensional space where each word $w \in V_W$ and each context $c \in V_C$ is represented with a vector $\vec{w} \in \mathbb{R}^d$ and $\vec{c} \in \mathbb{R}^d$. The set of all word vectors is commonly represented by a matrix W of size $|V_W| \times d$. Similarly, context vectors are the rows in a $|V_C| \times d$ matrix C.

4.2.2 Pointwise Mutual Information (PMI)

The co-occurrence matrix contains the global statistics of the corpus and is the primary source of information for most algorithms. However, not all co-occurrences are meaningful. Pointwise Mutual Information (PMI) is an information theoretic measure that can be used for finding collocations or associations between words [31] and can detect significant versus insignificant co-occurrences to some extent. For any word-context pair (w, c), PMI is defined as the log ratio between their joint probability and product of their marginal probabilities:

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)} \tag{4.1}$$

Based on the formulation, if two words co-occur more often than being independent then their PMI will be positive, and if they co-occur less frequent than being independent then their PMI will be negative. For instance, in the data we used for the chapter, the words “right” and “align” have a high PMI of 10.38[1] which indicates a strong collocation, but words “school” and “species” have a low PMI of -6.88 which means they are not likely to happen in the same context.

[1]Base 2 logarithms has been used here as well as all the experiments in the thesis.

Calculating PMI values for all word-context pairs gives us a $|V_W| \times |V_C|$ PMI matrix which we call it M^{PMI}. Most of entries in the co-occurrence matrix are zero $\#(w,c) = 0$, for which $PMI(w,c) = \log 0 = -\infty$. A common approach to handle the situation is to use M_0^{PMI} in which $PMI(w,c) = 0$ whenever $\#(w,c) = 0$. Another commonly accepted approach is to use Positive PMI (PPMI) matrix M^{PPMI} by replacing all the negative values with 0.

$$PPMI(w,c) = \max(PMI(w,c), 0) \tag{4.2}$$

In fact, a traditional approach to word embedding is to use explicit PPMI representation in which each word is described by its corresponding sparse row vector in the PPMI matrix M^{PPMI} and it is shown that it outperforms M_0^{PMI} on semantic similarity tasks [25].

A recognized shortcoming of PMI and consequently PPMI is their bias towards infrequent events [158]. This happens when a rare context c co-occurs with a word w a few times (or even once) and this often results in a high PMI value since $P(c)$ in PMI's denominator is very small. However, explicit PPMI representation is a well-known approach in distributional-similarity models.

4.2.3 Singular Value Decomposition (SVD)

SVD is a matrix factorization technique in linear algebra which has a broad range of applications. It is widely used for image compression and also dimensionality reduction. SVD on the matrix $\mathbf{X}_{n\times p}$ gives a factorization of the form $\mathbf{X} = \mathbf{U\Sigma V}^T$ in which $\mathbf{V}_{p\times p} = [\mathbf{v}_1 \ \mathbf{v}_2 \ ... \ \mathbf{v}_p]$ is an orthonormal basis for the row space of $\mathbf{X}$, $\mathbf{U}_{n\times n} = [\mathbf{u}_1 \ \mathbf{u}_2 \ ... \ \mathbf{u}_n]$ is an orthonormal basis for the column space of $\mathbf{X}$ and $\mathbf{\Sigma}$ is a diagonal matrix of singular values $\sigma_1, \sigma_2, ..., \sigma_p$. If $n > p$ then the corresponding $\sigma_{p+1}, ..., \sigma_n$ will be 0. The matrix $\mathbf{X}$ can be seen as a transformation matrix between the two bases $\mathbf{X}[\mathbf{v}_1 \ \mathbf{v}_2 \ ... \ \mathbf{v}_p] = [\sigma_1\mathbf{u}_1 \ \sigma_2\mathbf{u}_2 \ ... \ \sigma_n\mathbf{u}_n]$ or $\mathbf{XV} = \mathbf{U\Sigma}$.

Using this factorization one can just choose the largest singular values from $\mathbf{\Sigma}$ and eliminate the vectors in $\mathbf{U}$ and $\mathbf{V}$ corresponding to the smallest singular values to compress the data. In fact, the largest singular values and their corresponding columns in $\mathbf{U}$ and $\mathbf{V}$ can explain most of the information in the matrix, and hence we can reconstruct the original matrix even after removing the smallest singular values.

If we just pick the r largest singular values $\mathbf{X}_r = \mathbf{U}_{n\times r}\mathbf{\Sigma}_{r\times r}\mathbf{V}^T_{r\times p}$, then $\mathbf{X}_r$ will be the best rank r approximation of $\mathbf{X}$.

SVD can be applied to either co-occurrence matrix or the PPMI matrix to obtain the embedding. The word embedding is taken as the first d columns of U usually weighted by the singular values or the square root of singular values.

4.2.4 Skip-Gram with Negative Sampling (SGNS)

Word2vec [101] is the most popular and state-of-the-art method for training vector space representations for words. There are two types of training for the word2vec model: Continuous Bag-Of-Words (CBOW) and skip-gram. Word2vec scans the corpus and employs a moving window which defines the context of the words. The intuition of the method is that co-words that frequently appear in the same window have high semantic relatedness and hence, they should have similar word vectors. Both variants use a single layer neural network but their objective is different. In the CBOW model, the aim is to predict a word given its context while in the skip-gram the objective is to predict the context given the word itself. The error term is defined in such a way that maximizes the dot product of words and their context words' vectors. At the end of the optimization, weights of the network are considered as the word vectors.

Word2vec learns the words and context words' vectors separately. Updating the weights of the network for context words is computationally expensive and requires iterating over the entire vocabulary for each input instance. For this reason, hierarchical Softmax and negative sampling methods as optimization tricks have been proposed in order to improve the efficiency [103]. Negative sampling works better in practice, in which k words are randomly sampled from the vocabulary (i.e. negative samples) and considered as negative context, hoping that the sampled words are not related to the current word being processed. Then the weights for these negative context are updated accordingly. Since Word2Vec algorithm only observes the local context at any given time, it has no other choice than adopting such naive sampling strategy. The objective function of Skip-Gram with Negative Sampling (SGNS) [101] can be written as:

$$\mathcal{J} = \frac{1}{T}\sum_{t=1}^{T}\log\sigma(\vec{w}.\vec{c}) + k.\mathbb{E}_{c_N\sim P_D}[\log(-\vec{w}.\vec{c_N})]$$

where T is the size of the corpus, k is the number of negative samples, $\vec{c}$ and $\vec{c_N}$ are the current local context and the sampled negative context, respectively.

4.2.5 Global Vectors (GloVe)

The GloVe model [122] scans the corpus to determine the vocabulary and then build the global co-occurrence table by moving a context window over the text and counting the co-occurrences of the words. The co-occurrence table contains the global statistics about words that appear together in a window and is the primary source of information for unsupervised learning of word vectors. They exploit the information encoded in this table and formulate the problem as a sum of squared error minimization between the dot product of word and context word vectors and the log of their co-occurrences. Another weighting function is also applied to each error term to make the importance of error terms proportional to their co-occurrence value. This way, more frequent co-words will have more weight in updating their vectors. The objective function of GloVe is as follows:

$$\mathcal{J} = \sum_{i,j=1}^{V} f(X_{ij})(\vec{w_i}.\vec{w_j} + b_i + b_j - \log X_{ij})^2 \tag{4.3}$$

$$f(x) = \begin{cases} (x/x_{max})^{0.75} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{4.4}$$

where b_i and b_j are the bias terms for $\vec{w_i}$ and $\vec{w_j}$, X_{ij} is the co-occurrence count of the words, x_{max} is the only hyper-parameter to be tuned, and $f(X_{ij})$ is the handcrafted weighting function to adjust the error term proportional to the co-occurrence value.

4.3 EigenWord: Spectral Word Embedding with Negative Sampling

At a glance, our approach builds a symmetric co-occurrence matrix, calculates the PMI matrix, applies a threshold on PMI matrix to remove only some of the negative PMI values, and finally factorizes the resulting matrix to find the embedding.

In the first step, we use a fixed-size context window and scan through the corpus to build the global co-occurrence statistics matrix, just like many other methods such as GloVe. One of the key differences of our algorithm to others such as SGNS, is that we use a symmetric context in which we update X_{ij} and X_{ji} symmetrically whenever two words w_i and w_j appear in the same window. This will provide nice properties for the matrix in our optimization. Please note that from now on, we do not refer to context words as c since there is no distinction between words and contexts. Consequently, in our approach $|V_W| = |V_C| = n$ and hence both X and M^{PMI} are symmetric matrices of dimension $n \times n$.

After calculating the global co-occurrence matrix X, we apply PMI on it to obtain M^{PMI}. Here, instead of using M^{PPMI}, we propose to apply a threshold α on the PMI table to obtain M^{α} where each entry in the matrix is calculated as follows:

$$m_{ij}^{\alpha} = \begin{cases} PMI(w_i, w_j) & PMI(w_i, w_j) > \alpha \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

We refer to the word pairs with $m_{ij}^{\alpha} > 0$ as positive examples, and word pairs with $m_{ij}^{\alpha} < 0$ as negative examples. M^{α} is the final matrix that we factorize and the negative values in this matrix correspond to the negative examples that we employ in our model. Please note that by setting $\alpha = 0$ we get M^{PPMI} which does not exploit the negative examples. However, by choosing a negative threshold $\alpha < 0$, we keep a portion of negative values while disregarding the extreme cases. In fact, by adjusting this parameter α, we control the amount of negative examples to be used in the model.

It is noteworthy to mention that this thresholded PMI matrix, M^{α}, is different from what is being used in Levy and Goldberg's work [89]. In the shifted PPMI approach proposed by Levy and Goldberg [89], the PMI values are shifted by $-\log k$ and then all the negative values are removed. This way, not only all the originally negative PMI values are removed, but also some of the small positive PMI values (which are less than $\log k$) are eliminated as well. On the contrary, our method does the opposite operation by encouraging to keep the negative values. In the following, we will see how this negative threshold α can provide a better approximation to the true PMI matrix.

4.3.1 EigenWord Formulation

We propose an intuitive objective function that justifies the use of negative examples in the PMI matrix.

$$\mathcal{J}(W) = \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^{\alpha}(\vec{w_i}.\vec{w_j}) \tag{4.6}$$

where $\vec{w_i}$ and $\vec{w_j}$ are the d-dimensional embedded word vectors for w_i and w_j in the vocabulary, and W is the $n \times d$ matrix of all word vectors. Here, we formulated the problem as a maximization task where $\mathcal{J}(W)$ has to be maximized.

In this maximization formulation, the algorithm will maximize the similarity of word vectors, $\vec{w_i}.\vec{w_j}$, whenever $m_{ij}^{\alpha} > 0$ (i.e. positive examples), and it will minimize the similarity of word vectors whenever $m_{ij}^{\alpha} < 0$ (i.e. negative examples). Describing this maximization in different terms, word pairs with a strong degree of association will be placed close to each other while the negative examples will be placed far apart in the latent embedded space. The use of negative examples is ignored in most embedding algorithm since zero or negative PMI values in M^{PMI} are typically ignored and considered useless as in the explicit PPMI [25] and factorizations methods on PPMI [89]. Here, we show that the appropriate use of negative examples can be beneficial to the spectral algorithms and factorization based embedding methods.

Please note that most of the entries in M^{α} are zero, and word pairs with $m_{ij}^{\alpha} = 0$ have no effect on our optimization. It is also noteworthy to mention that positive examples are indeed more informative than negative examples. This means that excessive use of negative examples can have destructive effects on the quality of final embedding since the algorithm will mostly focus on making negative examples far apart rather than making word vectors with strong association closer to each other. However, we show that appropriate use of negative examples improves the distribution of words in the embedded space and prevents the concentration effect.

Considering Equation 4.6, we can rewrite our objective function as a trace optimization:

$$\begin{aligned}\mathcal{J}(W) &= \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^{\alpha}(\vec{w_i}.\vec{w_j}) = \sum_{i=1}^{n} \vec{w_i}.\Big(\sum_{j=1}^{n} m_{ij}^{\alpha}.(\vec{w_j})\Big) \\ &= Tr[W^T M^{\alpha} W] \end{aligned} \tag{4.7}$$

Then, our objective function has an optimal closed-form solution as a result of the following two theorems.

Theorem 1 ***(Courant-Fischer Theorem)*** *Let A be a symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $u_1, u_2, \ldots, u_n$, then:*

$$\lambda_1 = \min_{\|b\|=1} b^T Ab = \min_{b \neq 0} \frac{b^T Ab}{b^T b} = u_1^T A u_1, \tag{4.8}$$

$$\lambda_2 = \min_{\substack{\|b\|=1 \\ b \perp u_1}} b^T Ab = \min_{\substack{b \neq 0 \\ b \perp u_1}} \frac{b^T Ab}{b^T b} = u_2^T A u_2, \tag{4.9}$$

$$\vdots$$

$$\lambda_n = \lambda_{max} = \min_{\substack{\|b\|=1 \\ b \perp u_1 \perp \cdots \perp u_{n-1}}} b^T Ab = \max_{\|b\|=1} b^T Ab \tag{4.10}$$

$$= \max_{b \neq 0} \frac{b^T Ab}{b^T b} = u_n^T A u_n \tag{4.11}$$

Proof of the Courant-Fischer theorem can be found in standard linear algebra textbooks [43]. The following theorem [120] is an immediate consequence of the Courant-Fischer theorem.

Theorem 2 *Given a symmetric matrix $A_{n \times n}$, and an arbitrary unitary matrix $B_{n \times d}$, then the trace of $B^T AB$ is maximized when B is an orthonormal basis for the eigenspace of A associated with its algebraically largest eigenvalues [76]. In particular, the eigenbasis itself is an optimal solution: If $U = [u_1, \ldots, u_d]$ is the set of eigenvectors associated with d largest eigenvalues $\lambda_1, \ldots, \lambda_d$, and $U^T U = I$, then:*

$$\max_{\substack{B \in \mathbb{R}^{n \times d} \\ B^T B = I}} Tr[B^T AB] = Tr[U^T AU] = \lambda_1 + \cdots + \lambda_d \tag{4.12}$$

Our thresholded PMI matrix, M^α, is symmetric and therefore, our optimization in Equation 4.7 fits into theorem 2. Consequently, our optimization has an optimal closed-form solution in which the word vectors matrix W is formed by the eigenvectors of M^α corresponding to its d algebraically largest eigenvalues. This formulation is simply a more accurate approximation of the true objective than SVD factorization of PPMI or shifted PPMI.

4.3.2 Connection to SVD and an alternative solution (SVD-NS)

Consider the singular value decomposition of the thresholded PMI matrix, $M^\alpha = U\Sigma U^T$. We know that U is the set of eigenvectors of $M^\alpha M^{\alpha T}$ and V is the set of eigenvectors of $M^{\alpha T} M^\alpha$. In case of symmetric input, $M^\alpha M^{\alpha T} = M^{\alpha T} M^\alpha$, and consequently, $U = V$ and the well-formed unique factorization of $U\Sigma U^T$ can be obtained.

Moreover, it is easy to show that eigenvectors of powers k of a matrix, A^k, are equivalent to the eigenvectors of the matrix itself, but, their corresponding eigenvalues will be taken to the power k.

$$Av = \lambda v \tag{4.13}$$

$$A^2 v = AAv = A(\lambda v) = \lambda(Av) = \lambda\lambda v = \lambda^2 v \tag{4.14}$$

Therefore, U which is the set of eigenvectors of $M^\alpha M^{\alpha T} = M^{\alpha 2}$ is equivalent to the eigenvectors of M^α, and Σ will be the absolute value (i.e. magnitude) of eigenvalues of M^α in decreasing order. As a consequence, one can apply SVD on the symmetric thresholded PMI matrix M^α and consider the first d columns of U, corresponding to d largest singular values of the matrix, as the final word embedding.

We should mention that this alternative solution is equivalent to the originally proposed solution only if the d largest singular values correspond to the d algebraically largest eigenvalues. More formally, if $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues in decreasing order, then the condition is satisfied if $\nexists \lambda_k < 0$ *such that* $|\lambda_k| \geq \lambda_d$. This is a guaranteed case when M^α is Positive Semi-Definite (PSD) since all the eigenvalues will be positive. Nevertheless, in case of non-PSD, M^α can be easily converted into a PSD matrix since it is already symmetric. This can be done by making it a diagonally dominant matrix, in that each diagonal element is greater than or equal to the sum of all non-diagonal elements on that row, $m^\alpha_{ii} \geq \sum_{j,j\neq i} m^\alpha_{ij}$. In fact, by adjusting only the diagonal elements of M^α which are the strength of association of words with themselves, we can make it a PSD matrix. This process should not have drastic effects on the quality of embedding since it does not change the strength of pairwise word relations. However, in our experiments in this chapter, we have used the alternative SVD solution on the original M^α without converting it to PSD, and still achieved promising results. We refer to this alternative solution as SVD-NS.

Let us analyze the SVD-NS algorithm in the case of non-PSD M^{α} matrix. Since the matrix U in SVD is the set of eigenvectors of $M^{\alpha}M^{\alpha T}$, applying SVD on M^{α} and taking the first d dimensions of U as word vectors is equivalent to plugging in $M^{\alpha}M^{\alpha T}$ instead of M^{α} in our EigenWord formulation in Equation 4.7. This is not a bad solution at all, since the elements in $M^{\alpha}M^{\alpha T}$ correspond to context similarity of words. We can say that elements in M^{α} are the first-order affinities or first-order similarities of words directly through co-occurrence and elements in $M^{\alpha}M^{\alpha T}$ are the second-order affinities of words through context similarity. Each row of M^{α} corresponds to the global context vector of a particular word. Since M^{α} is symmetric, each element of $M^{\alpha}M^{\alpha T}$ corresponds to the dot product of two context vectors that gives us their context similarity. To clarify this, consider the words "go" and "went" in sentences "I go to school" and "I went to school". The words "go" and "went" may never co-occur directly in the corpus resulting the corresponding entry in M^{α} to be zero, however, they have a lot of common context which makes them similar and is quantifiable in $M^{\alpha}M^{\alpha T}$. Please note that the context-based similarity $M^{\alpha}M^{\alpha T}$ is a dense matrix and is computationally impractical to compute, however, SVD-NS implicitly solves the formulation for context-based similarity without needing to explicitly compute the $M^{\alpha}M^{\alpha T}$. In fact, EigenWord and SVD-NS both use the same formulation with the difference that one works on the co-occurrence based similarity and the other works on the context based similarity. The source code of both algorithms is available at our GitHub page: `https://github.com/behrouzhs/svdns`.

4.4 Experiments

4.4.1 Data and Vocabulary

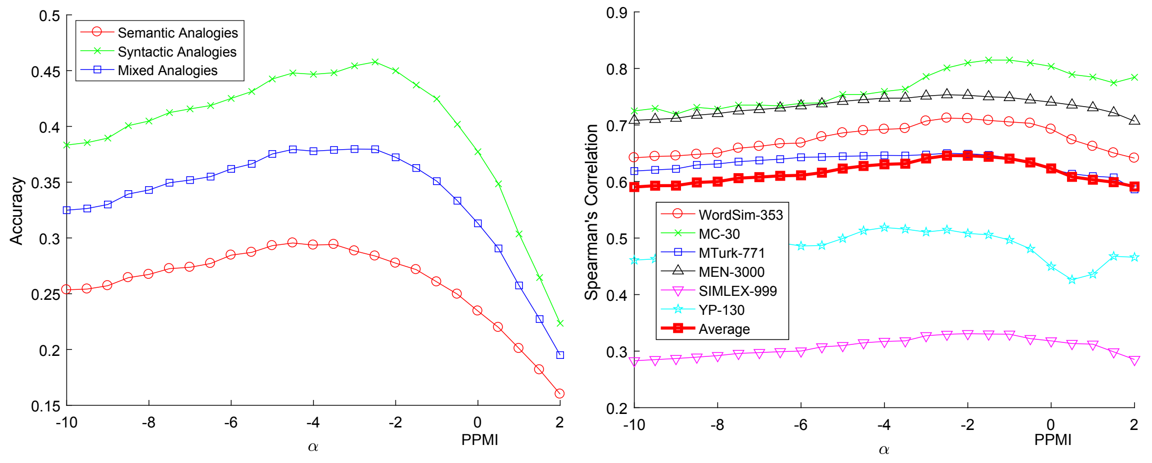
For the training of models, we have used English Wikipedia dump of March 05, 2016. After stripping the HTML tags and extracting the clean text of Wikipedia articles, we removed all the special characters and punctuation from the text. Then we converted the text into lowercase and tokenized it. The data have 2.1 billion tokens resulting in 8.9 million unique words as vocabulary. However, most of the vocabulary are infrequent words or they belong to named entities such as names of people, places, and organizations.

We have applied a filtering on the vocabulary to target the words in English dictionary as well as to reduce the size of vocabulary to a reasonable range. First, we filtered out all the words that appeared in the data less than 5 times. This is a common threshold that is used in other algorithms as well (e.g GloVe). We also removed all the words that contained non-English alphabet (e.g. names of people). Afterwards, we used WordNet database to identify words that exist in the English dictionary. WordNet is an ontology defining the relationship between the words and has an internal categorization of words into verbs, nouns, adjectives, and adverbs. We have used all the words that appeared in WordNet database as the English vocabulary. We also kept all the words with more than 3000 occurrences which do not necessarily exist in the English dictionary. After these filtering steps, the 8.9M unique words in Wikipedia were filtered and resulted in 163188 words.

In our experiments, all the algorithms were trained on the exact same preprocessed input text data. We have also used the exact same vocabulary for all the algorithms. This will ensure a fair comparison as the size of the training data has a great influence on the quality of embeddings. The dimensionality of embeddings is 100 in all our experiments.

4.4.2 Evaluation method

For the evaluation of algorithms, we have used two well-known tasks of word similarity and word analogy. For the word similarity task, there exist several datasets containing word pairs with their corresponding human-assigned similarity score. These datasets are commonly used to evaluate the quality of word embeddings and to see how good they have modeled the word similarities in the embedding. The quantitative values of scores are not important here, but rather the ranking of different word pairs is the main focus. Therefore, Pearson's rank-order correlation is always used in word similarity task. In this task we have used 8 different dataset including *WordSim353* (WS-M) [48], *WordSim Similarity* (WS-S) and *WordSim Relatedness* (WS-R), MEN [24], *Mechanical Turk* (MTurk), SimLex [58], MC, and YP [171]. For the analogy task, we have used Google's analogy dataset [101] which contains 19544 questions of the form "a is to a^* as b is to b^*". Given three of the elements, the algorithm has to predict the missing element. About half of questions are semantic



(a) Word analogy task (b) Word similarity task

Figure 4.1: Accuracy of SVD-NS algorithm on (a) word analogy (b) word similarity tasks with respect to the amount of negative samples.

(e.g. "*brother* is to *sister* as *son* is to *daughter*") and the other half are syntactic questions (e.g. "*cold* is to *colder* as *short* is to *shorter*").

4.4.3 Analysis of the amount of negative examples

Let us first analyze the effect of the quantity of negative examples in the model. For this reason, we have run SVD-NS algorithm with various PMI thresholds ranging in $[-10, +2]$ and evaluated each case on analogy and word similarity tasks. Figure 4.1 illustrates the results of this experiment. Figures 4.1a and 4.1b show the accuracy of our algorithm on word analogy and word similarity tasks, respectively. Please note that the special case of $\alpha = 0$ is equivalent to the SVD factorization of PPMI matrix [90]. As we can see from both figures, using negative examples can boost the performance of factorization methods dramatically and yields a better quality embedding. In both analogy and word similarity tasks, the peak performance and accuracy has achieved when $\alpha = -2.5$ and in that case the number of positive examples is 1.54 times the number of negative examples. As we discussed before, excessive use of negative examples will have harmful effects since it will shift the focus of the algorithm to less important information. For instance, as we can observe from the Figure 4.1 and it was previously shown in [25], using the entire PMI matrix M_0^{PMI} and factorizing it using SVD yields worse results than factorizing M^{PPMI}.

Table 4.1: Evaluation of different word embedding algorithms on 8 word similarity datasets.

	WS-S	WS-R	WS-M	MC	MEN	MTurk	SimLex	YP
	203	252	353	30	3000	771	999	130
SVD	0.4913	0.1890	0.3624	0.4096	0.2722	0.1600	0.0881	0.122
SVD-L	0.2705	0.1091	0.1866	0.1301	0.2321	0.2650	0.0882	0.0878
SVD-S	0.6245	0.3235	0.4854	0.6483	0.5356	0.4124	0.212	0.2445
PPMI-SVD-US	0.6865	0.4893	0.5968	0.7257	0.6823	0.5713	0.2683	0.3710
PPMI-SVD-Us	0.7189	0.6007	0.6674	0.7749	0.7302	0.6155	0.2976	0.4490
PPMI-SVD-U	0.7201	0.6387	0.6925	**0.8035**	0.7402	0.6217	0.3180	0.4496
SPPMI-SVD-Us	0.6469	0.5416	0.6046	0.7405	0.7075	0.5735	0.2728	0.4472
SPPMI-SVD-U	0.6412	0.5813	0.6238	0.7566	0.7000	0.5766	0.2764	0.4574
GloVe	0.6743	0.5534	0.5991	0.6649	0.7040	0.6227	0.3154	0.4872
CBOW	0.7457	0.5853	0.6714	0.7427	0.7079	0.6132	0.3274	0.3403
SGNS	**0.7587**	0.6519	0.7096	0.7946	0.7306	0.6441	0.3236	0.4679
SVD-NS	0.7519	**0.6537**	**0.7120**	0.8008	**0.7534**	**0.6499**	**0.3297**	**0.5145**

4.4.4 Quantitative Evaluation

Table 4.1 compares 12 algorithms on 8 word similarity datasets. The numbers in the table are Pearson's correlation between the rankings provided by the algorithms and the rankings of the human-scoring.

SVD, SVD-L, and SVD-S are the factorizations of co-occurrence, log co-occurrence, and square root of co-occurrence matrices, respectively. This type of factorization is popularized in NLP through Latent Semantic Analysis [38]. PPMI-SVD-US, PPMI-SVD-Us, and PPMI-SVD-U are all SVD factorizations on PPMI matrix but they have different singular vector weighting scheme in that, they weight the singular vectors by singular values, square root of singular values, and no weighting, respectively. SPPMI-SVD-Us and SPPMI-SVD-U are the SVD factorizations on shifted PPMI matrix [89] where the shift is $\log 5$. GloVe is trained with its recommended parameter setting (i.e. $x_{max} = 100$), CBOW and SGNS are trained with negative sampling set to 5. Our proposed algorithm, SVD-NS, is trained with $\alpha = -2.5$.

As we can see from Table 4.1, our algorithm provides the best results in 6 out of 8 datasets. SGNS is the best on only WordSim Similarity dataset, and PPMI-SVD-U is the best on MC dataset and on the rest of the datasets our method outperforms others by a fair margin. We have to mention that in analogy task SGNS provides better results than SVD-NS. However, our main goal is to boost matrix factorization methods with the use of negative examples. We have parallelized SVD-NS algorithm using OpenMP and it is one order of magnitude faster than multi-threaded SGNS.

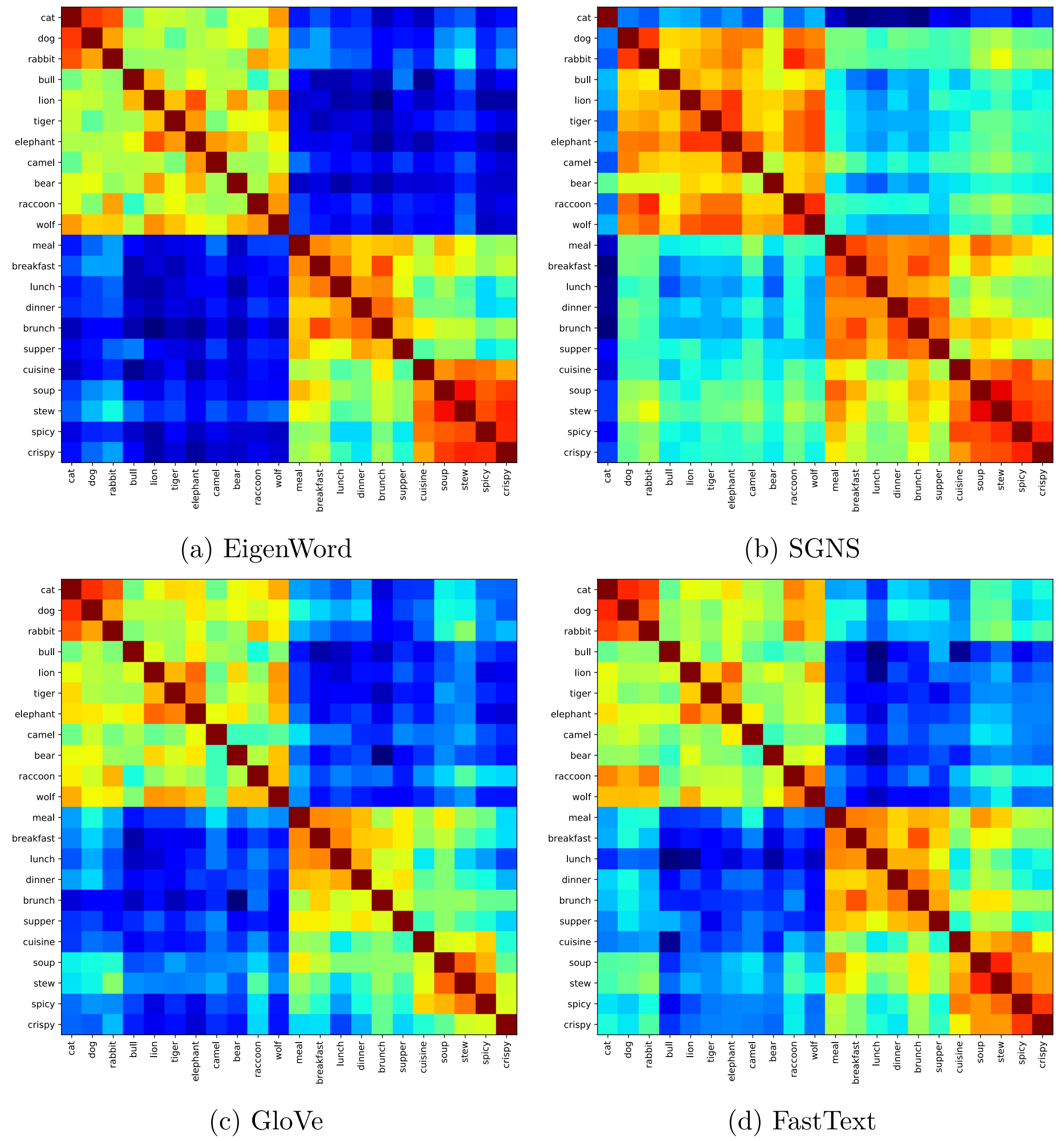
4.4.5 Qualitative Evaluation

For the qualitative evaluation of embeddings, we selected 22 words from two groups of 11 animal names and 11 food related words and visualize the pairwise Cosine similarity of words in the embedding spaces. Figure 4.2 illustrates the similarity matrix for four algorithms: EigenWord, SGNS, GloVe, and FastText. Colors range from dark blue for lowest similarity to dark red for highest similarity and the color-bar is removed to save space.

As we can see from Figure 4.2, almost all embedding algorithms are able to capture semantic similarity and consequently, they all have great within-group cohesion. However, our proposed algorithm, EigenWord, has remarkably better between-group separation, and the two unrelated word clusters are well separated in the embedding. This is the result of effectively utilizing negative examples that leads to better use of the space in the embedding.

4.5 Conclusion

In this chapter, we analyzed the use of negative examples in word embedding context both theoretically and empirically. We proposed an intuitive objective function for the word embedding problem and provided an optimal closed-form solution for it using matrix factorization techniques. Our thresholded PMI matrix is simply a better approximation of the word associations. And our objective function is a more accurate approximation of the true objective than SVD factorization of PPMI or shifted PPMI. Our algorithm removes the use of many hyper-parameters in other algorithms, such as eigenvalue weighting and dealing with word and context vectors separately. In fact, our optimal solution is achieved using the eigenvectors themselves and no additional weighting is needed. Moreover, it builds, maintains, and exploits symmetric co-occurrence and PMI matrices, and consequently, its word and context vectors happen to be the same in our algorithm. The only parameter in our method is α which controls the amount of negative examples to be used in the algorithm. As a rule of thumb, one should not have more negative examples than $\frac{2}{3}$ of positive ones. In the next



(a) EigenWord (b) SGNS (c) GloVe (d) FastText

Figure 4.2: Cosine similarity matrix of 22 words in different embedding spaces (a) EigenWord, (b) SGNS, (c) GloVe, and (d) FastText. Words are from two groups of 11 animal names and 11 food related words. Dimensionality of embeddings is 100 for all embeddings. Colors range from dark blue for lowest similarity to dark red for highest similarity.

chapter, we extend the approach and propose a multi-sense embedding technique to tackle Word Sense Disambiguation (WSD).

Chapter 5

Efficient Word Sense Disambiguation

In previous chapters, we analyzed word embedding algorithms and proposed two embedding methods, KUBWE and EigenWord, that both address the negative sampling problem in order to improve the distribution of words in the latent space. In this chapter, we take it a step further and present an efficient word sense disambiguation and embedding algorithm that learns multi-prototype sense vectors to accommodate different meanings of words. Our method provides both sparse sense vectors to be used for word sense induction, and dense sense embeddings to be used in training of machine learning models for downstream NLP tasks. We disambiguate the ambiguous words by clustering their context words. However, unlike previous methods where each context is assigned to one sense, we use a soft clustering approach to learn the distribution of context words over different senses, and use it to break the global context vectors into multiple sense-specific sparse vectors. After constructing the sense-word relation matrix from the word-word co-occurrence matrix, we use it for learning sense embeddings using a fast matrix factorization approach. Our experiments on SemEval 2013 competition data show that our method achieves the state-of-the-art accuracy in a more efficient way.

5.1 Introduction

Continuous-valued word representations have become very popular since they improve the performance of many natural language processing applications. These distributional representations are learned from large text corpora and can explain many semantic properties and relationships between concepts represented by words. Many neural-network based approaches [16, 32, 101, 103] as well as count-based [122, 5] and matrix factorization-based methods [89, 141] have been proposed for learning distributed word representations. The Skip-Gram model was very successful so that many improvements and variants are proposed for it. For instance, FastText [19] enriches the Skip-Gram word embeddings with sub-word information by considering character n-grams of different lengths and representing words as the sum of their n-gram vectors.

Conventional word embeddings learn a unique representation for each word and therefore, they cannot deal with word ambiguity which is an important property of the natural language. For example, the word "right" may refer to a direction or to being correct depending on the context. Consequently, the learned vector representation is either dominated by the most frequent meaning or it is a weighted average of all the meanings [6]. Clearly neither case is desirable for practical applications. Since word representations have been used as word features in many NLP tasks including dependency parsing [29], named-entity recognition [156], emotion recognition from tweets [112] and sentiment analysis [97], using a multi-prototype representation can potentially increase the performance of such representation-based approaches.

Several approaches have been proposed for learning sense-specific word vectors. Reisinger and Mooney [127] proposed a clustering based approach in which context words are clustered in order to produce groups of similar context vectors. An average "prototype" vector is then computed separately for each cluster, producing a set of sparse sense vectors for each word. Huang et al. [64] uses a similar clustering approach to cluster sparse TF-IDF context vectors to create a fixed number of senses for each word. Then they relabel each word token with the clustered sense before learning embeddings. SenseGram [121] and AdaGram [10] are among the most successful word sense embedding methods. AdaGram is a Bayesian extension of the Skip-Gram model and provides Word Sense Disambiguation (WSD) functionality based on the induced sense inventory. SenseGram transforms word embeddings to sense embeddings via graph clustering and uses them for WSD.

In this work, we present a fast unsupervised multi-prototype sense representation method. Instead of using traditional clustering approaches which assign each context word to a specific sense, we use an efficient soft clustering approach where contexts are assigned to different senses according to their membership values. For any ambiguous word w, most of its context words are common across all of its senses. Therefore, learning the distribution of context words across senses is an intuitive and natural choice. Moreover, our algorithm outputs both sparse and dense representations for senses. The sparse representations are interpretable and can be used for word sense induction. The correct sense of the word can be induced by comparing the usage of the word in a given sentence with the sparse sense vectors and picking the most similar sense. The dense representations can be used as word features in training machine learning models for any desired NLP application such as sentiment analysis, translation, etc.

5.2 Proposed Method

Similar to word embeddings, sense embeddings should also be learned from a large text corpus such as Wikipedia or Common Crawl data. In our method, we first calculate the global symmetric word-word co-occurrence counts matrix X by moving a t-sized context window over the corpus. Assuming that the corpus contains n unique words (i.e. vocabulary $V = \{w_1, w_2, \dots, w_n\}, |V| = n$), then the co-occurrence counts X is a highly sparse $n \times n$ matrix. This co-occurrence matrix contains global statistics about word relations and is the primary source of information for many word embedding algorithms such as GloVe [122]. Each row i of this matrix is the global context vector $\overrightarrow{x}_i$ of a particular word w_i which represents all possible context words that are co-occurred with w_i in the corpus. Please note that each global context vector $\overrightarrow{x}_i$ is the algebraic sum of all context vectors of all the meanings of w_i. In section 5.2.2 we will break these vectors down into multiple context vectors, one for each sense.

5.2.1 Refining the co-occurrences and extracting word relations

Many of the co-occurrences in X are not informative. For instance, the words "also" and "their" co-occur thousands of times in Wikipedia but this should not be considered as a high degree of association. Pointwise Mutual Information (PMI) is an information theoretic measure that can be used for finding collocations or associations between words [31] and is widely used in count-based and matrix factorization based word embeddings. For any word pair (w_i, w_j), PMI is defined as the log ratio between their joint probability and product of their marginal probabilities:

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \tag{5.1}$$

If two words co-occur more often than being independent then their PMI will be positive, and if they co-occur less frequently than being independent then their PMI will be negative. Hence, a commonly accepted approach is to use Positive PMI (PPMI) matrix by replacing all the negative values with 0, $PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0)$. Therefore, unlike many sense disambiguation methods that use TF-IDF approach which is more suitable for document-term matrices, we take the PPMI of the co-occurrence matrix X that results in a more refined and accurate word relation matrix P.

5.2.2 Word Sense Disambiguation: Obtaining Sparse Sense Vectors

For any ambiguous word w_i we break its refined context vector $\overrightarrow{p}_i$ (i.e. i-th row of the PPMI matrix P) into multiple context vectors, one for each sense of the word. Assuming that word w_i has k_i different senses $\{s_{i1}, s_{i2}, \dots, s_{ik_i}\}$, we learn a sparse context vector $\overrightarrow{s}_{ij}$ for each sense s_{ij} in such a way that $\sum_{j=1}^{k_i} \overrightarrow{s}_{ij} = \overrightarrow{p}_i$. This will ensure that sum of all context vectors from different senses equals to the global context vector of the word.

We break the context vector $\overrightarrow{p}_i$ of the word w_i by clustering the context vectors of its context words. Let $L_i = \{l_1, l_2, \dots, l_m\}$ be the set of all non-zero entries in $\overrightarrow{p}_i$ which is the set of indices of all context words of w_i. Then, we select a slice C_i of the matrix P where only rows corresponding to L_i are selected. C_i will be a $m \times n$ matrix representing the context vectors of the context words of w_i. Our aim is to learn the distribution of context words over senses and we achieve this by applying a soft clustering. We need to do this in an efficient way since the clustering has to be done for each ambiguous word separately.

Fuzzy C-means clustering is not applicable in this case since the input matrix is high dimensional and therefore, all the centroids in fuzzy clustering will converge to the center of gravity of the entire data distribution [170]. We propose to use Non-negative Matrix Factorization (NMF) [86] in order to cluster the columns of C_i. NMF decomposes the matrix $C_i = WH$ in the form of multiplication of two matrices $W_{m\times k}$ and $H_{k\times n}$ by minimizing the reconstruction error of the original matrix constraining all elements of all three matrices to be non-negative. NMF can be used to cluster the columns of the input matrix and $H_{k\times n}$ is known to produce membership values of n

points to k clusters [79]. We use WordNet lexical database to obtain the number of senses k_i for each ambiguous word w_i and apply NMF with k_i components. Finally, we normalize membership values for each context word to sum 1 and multiply the membership matrix by the global context vector $\overrightarrow{p}_i$ to obtain the sparse sense vectors $\overrightarrow{s}_{ij} = \overrightarrow{h}_j \odot \overrightarrow{p}_i$. Normalizing the columns of H ensures $\sum_{j=1}^{k_i} \overrightarrow{s}_{ij} = \overrightarrow{p}_i$.

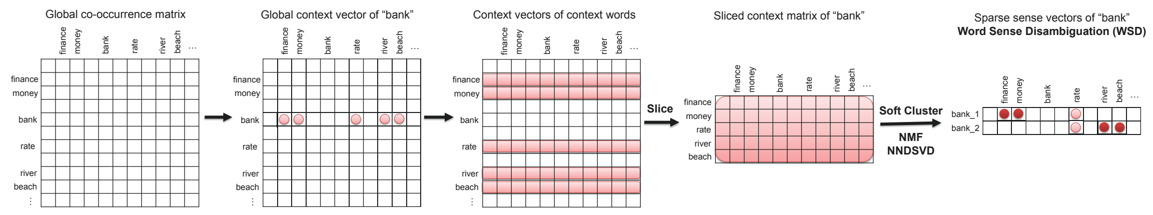
Latent Dirichlet Allocation (LDA) and topic modeling [18] also seem to be a good choice for the soft clustering of contexts as it can learn the distribution of context words in different topics (or senses in this case). However, LDA is resource intensive and it was computationally intractable to disambiguate all words in our experiments using LDA. For instance, disambiguating a single word with around 50,000 context words (i.e. LDA on a matrix of 50,000 rows) takes more than 1.5 hours while NMF on the same matrix ends in 2 minutes.

Further improving the efficiency Although NMF using coordinate descent optimization is pretty fast, we take it a step further and make our algorithm faster by only taking the initialized value of H. We use the Non-Negative Double Singular Value Decomposition (NNDSVD) method [21] which is often used to initialize W and H matrices for NMF and is shown to provide a very good starting point [21]. Therefore, we just use the output of NNDSVD without optimizing it further to reduce the runtime to the minimum. Our experiments did not show a significant difference between the accuracy of the two methods, thus, we pick the fastest and report the results of NNDSVD in section 5.3.

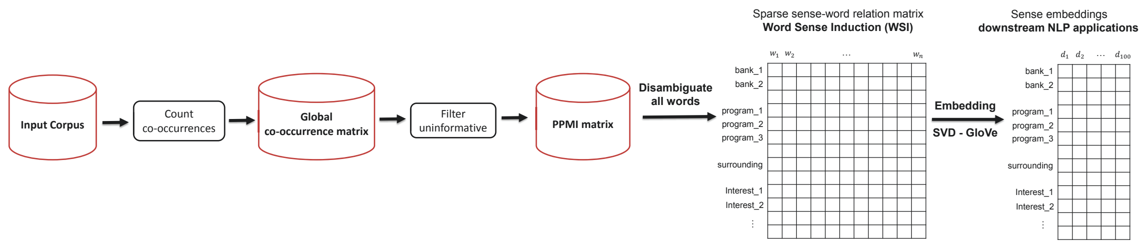
5.2.3 Word Sense Induction Using the Sparse Sense Representation

The sparse sense vectors can be used to infer the correct sense of a word in a given sentence based on its context words. Then we can select the corresponding sense embedding and use it for the underlying NLP task. For the purpose of word sense induction, we build the context vector of the query sentence $\overrightarrow{q}_i$ and use Cosine similarity to find the most similar sense vector $\{\overrightarrow{s}_{i1}, \ldots, \overrightarrow{s}_{ik_i}\}$:

$$j^* = \arg\max_j \frac{\overrightarrow{s}_{ij}.\overrightarrow{q}_i}{|\overrightarrow{s}_{ij}| \times |\overrightarrow{q}_i|} \tag{5.2}$$


Global co-occurrence matrix
Global context vector of "bank"
Context vectors of context words
Slice
Sliced context matrix of "bank"
Soft Cluster
NMF
NNDSVD
Sparse sense vectors of "bank"
Word Sense Disambiguation (WSD)
finance
money
bank
rate
river
beach
bank_1
bank_2

(a) Disambiguation procedure for a particular word, "bank" in this example.



(b) Overall flowchart for disambiguating all words and multi-sense embedding.

Figure 5.1: Flowchart of the proposed Word Sense Disambiguation (WSD) and multi-sense embedding technique.

5.2.4 Word Sense Embedding: Obtaining Dense Representations

We augment all the sparse sense vectors of all ambiguous words to the PPMI matrix P, build a global sense-word co-occurrence matrix S, and use it for the embedding. After constructing the sense-word relation matrix, one can use GloVe [122] optimization or any type of matrix factorization approaches [89, 141] in order to obtain the sense embeddings. For the purpose of efficiency we use a sparse Singular Value Decomposition (SVD) on the sense-word relation matrix $S = U\Sigma V^T$ and take the first d singular vectors of U as our dense representations. This sparse factorization uses the efficient Lanczos iterative sparse eigensolver to obtain the top d singular vectors corresponding to the largest singular values and is shown to capture semantic similarities pretty well when applied on co-occurrence type relational matrices [141].

Figure 5.1 shows the flowchart of our proposed Word Sense Disambiguation (WSD) and multi-sense embedding technique. Figure 5.1a illustrates the disambiguation procedure for a particular word, "bank" in this example, and Figure 5.1b depicts the overall flowchart for disambiguating all words and multi-sense embedding.

Table 5.1: Nearest neighbors of sample words in our sense embedding vector space

Word	Nearest neighbors
program_1	mentoring, internships, scholarships, educational, volunteering, trainings
program_2	software, implementations, computing, data, protocol, algorithms, automated
program_3	aired, hosted, televised, telecast, christmas, filmed, wrestling, announcer
square_1	plaza, street, apartments, palace, town, cafe, landmark, mall
square_2	rectangular, arched, octagonal, cornice, triangular, hexagonal, gable
square_3	km, meters, households, kilometers, hectares, ft, sq, islander, wingspan
interest_1	interested, attention, fascination, curiosity, attraction, hobby, keen, enthusiasm
interest_2	ventures, assets, acquisitions, investments, contracts, subsidiaries, leases, holdings
interest_3	romantically, girlfriend, friend, relationship, friendship, actress, partner

5.3 Evaluation

We used Wikipedia dump of April 2018 as our training corpus which has about 2.2 billion tokens. After a few preprocessing steps we applied our algorithm to learn both sparse and dense representations for all the senses of all words.

Qualitative Evaluation Table 5.1 shows the nearest neighbors of the different senses of words “program”, “square”, and “interest” in our sense embedding vector space. As we can see it can disambiguate different senses very well. For instance, the three senses of word “program” correspond to training/educational programs, software programs, and television programs which are quite distinct.

Quantitative Evaluation We evaluated our learned sense vectors on the SemEval 2013 Task 13 word sense disambiguation competition data. This dataset contains 50 ambiguous words and about 100 example sentences per each word for a total of 4664 instances. The algorithms have to infer the correct sense for each of the instances. For many test sentences, multiple senses are applicable with different levels of appropriateness and this enables various scoring systems. We have used all 5 official scoring functions released by the competition organizers (Jaccard Index, Positional Tau, Weighted NDCG, Fuzzy NMI, and Fuzzy B-Cubed) and compared our algorithm with the top winning teams as well as the current state-of-the-art approaches AdaGram and SenseGram in Table 5.2. Looking at the performances of different algorithms in the table we can see that our method has achieved the best overall accuracy, and that it is very close to the current state-of-the-art.

Table 5.2: Comparison of our method's performance to participants of the SemEval 2013 Task 13 and two systems based on word sense embeddings (AdaGram and SenseGram)

Model	Jaccard	Tau	WNDCG	Fuz. NMI	Fuz. B-Cub.	Average
AI-KU	0.197	0.620	**0.387**	0.065	0.390	0.3318
AI-KU (rem5-add1000)	0.245	0.642	0.332	0.039	0.451	0.3418
Unimelb (50k)	0.213	0.620	0.371	0.060	0.483	0.3494
UoS (top-3)	0.232	0.625	0.374	0.045	0.448	0.3448
SenseGram, p=2, d=100	0.197	0.615	0.291	0.011	**0.615**	0.3458
AdaGram, α=0.05, d=100	**0.274**	**0.644**	0.318	0.058	0.470	0.3528
Our Method	0.215	0.610	0.335	**0.072**	0.560	**0.3584**

5.4 Conclusion

In this chapter, we analyzed word sense disambiguation and embedding methods in NLP and proposed an efficient alternative that achieves state-of-the-art accuracy. Our method uses a soft clustering approach to group context words, which has a clear advantage over conventional context clustering methods by learning a more accurate distribution of words across different meanings. In fact, it splits the global co-occurrence distribution into multiple distributions over the senses of words. We provide both sparse and dense representations for each sense that can accommodate most NLP needs from word sense induction (i.e. inferring the correct sense of a word given a query sentence) to training machine learning models using dense vector space representations. Qualitative and quantitative evaluations show the effectiveness of the proposed method.

Chapter 6

Intrinsic Evaluation of Word Embeddings

In this chapter, we present the results of intrinsic evaluation of different word embedding algorithms. In particular, we use word similarity datasets to compare the accuracy of embeddings. These datasets contain word pairs with their corresponding human-assigned similarity score. The idea is that the embedding space has to reflect these similarity scores to some degree. In other words, similar words should be spatially close together while dissimilar words should be further away, which is basically the main objective of word embeddings.

This type of evaluation is called intrinsic evaluation since the word vectors are evaluated in an isolated way and are not used in any downstream NLP task. Nevertheless, it is a fundamental manner of evaluating how embeddings preserve semantics and word similarity, and it is being used as a standard benchmark by many researchers.

6.1 Corpus for training word embeddings

For the training of models, we have used English Wikipedia dump of April 20, 2018. The set of articles in Wikipedia is considered as a good representative of the English language since it contains a broad range of topics and vocabularies. Moreover, it has diverse and accurate grammar and contains a wide variety of sentence structures. For these reasons, it is often used as a generic corpus for pre-training models. We also used this huge corpus for training different word embeddings.

Table 6.1 shows some statistics about the Wikipedia text corpus. The number of paragraphs, sentences, tokens, and vocabulary are calculated after the preprocessing. Even after extensive preprocessing, there are about 7.8 million unique words in the vocabulary even though the entire English dictionary has no more than 200,000 words. Most of these 7.8 million words are non-English words, pseudo codes or code snippets, markup text, and many other forms of non-textual content in Wikipedia

Table 6.1: Statistics of the English Wikipedia dump (April 2018)

Element	**Value**
Size of the original XML formatted data	$\approx$ 65GB
Size of the clean text data	$\approx$ 12GB
Articles	5,838,382
Paragraphs	36,525,012
Sentences	93,993,014
Tokens	2,222,963,243
Unique words (vocabulary size)	7,775,664
Common words with IMDB movie review dataset	87,486
Common words with IMDB with minimum 5 occurrence	75,914

pages. Removing these rare vocabularies does not affect the corpus in any way and it will still be as good a representative of the language as it was before.

In Chapter 7 we evaluate the embeddings on downstream NLP tasks and we use the same corpus introduced here. The same Wikipedia dump is used for both intrinsic evaluation and also for the sentiment analysis task on IMDB movie reviews in Section 7.2. Therefore, we limited the vocabulary of the embeddings to the common vocabulary between Wikipedia and IMDB, which contains nearly all English words.

6.2 Cleaning and Preprocessing of the Wikipedia Corpus

The Wikipedia dump comes in XML format where all the articles are in a single large XML file. There are many components in every page including the header, sidebars, table of contents, notes, references, external links and the main body of content. These parts are nicely formatted in XML and is pretty straightforward to remove the nonessential components and extract the clean text from the articles. Our preprocessing and cleaning steps are listed below:

1. Strip the HTML tags and extract the clean text of Wikipedia articles using the WikiExtractor tool[1] [7].

2. Remove any remaining tags between “<” and “>” characters. This removes some of the remaining markup text from the articles, though not all, for instance from `https://en.wikipedia.org/wiki/HTML`.

[1]`https://github.com/attardi/wikiextractor`

Table 6.2: Word similarity datasets used for comparison.

Dataset Name	**Number of Word Pairs**
WordSim Similarity (WS-SIM) [2]	203
WordSim Relatedness (WS-REL) [2]	252
WordSim-353 (WS-ALL) [48]	353
MEN [24]	3000
MC [104]	30
RG [131]	65
SimLex [58]	999
YP Verbs [171]	130
MTurk-771 [56]	771
Stanford Rare Words (RW-STN) [96]	2034

3. Remove any content between parentheses. Most of the content enclosed in parentheses in Wikipedia are either non-English roots of words or equivalents, non-English named entities including peoples' names, pronunciation tips, etc. Removing this information does not affect the content of the sentences.

4. Remove any content between braces. This removes most of the content from code snippets, for example from `https://en.wikipedia.org/wiki/C_Sharp_(programming_language)`.

5. Remove links and URLs from all articles. This is done by using a simple regular expression.

6. Use spaCy library[2] [62] to detect sentences and break paragraphs into sentences.

7. Tokenize text using spaCy library and lowercase all tokens.

6.3 Results on Word Similarity Datasets

We have used 10 different word similarity datasets to compare the quality of word embeddings. These datasets and their corresponding number of word pairs are shown in Table 6.2.

As for the word embedding algorithms, we used a variety of algorithms in the comparisons. For all the matrix factorization based algorithms we used first d columns in

[2]`https://spacy.io/`

the U matrix as the word vectors without any eigenvector weighting techniques. The original eigenvectors in U without weighting are shown to be good in general [89]. We tuned all word embedding algorithms as described below. The list of word embedding algorithms and their parameters are as follows:

1. **SVD**: Truncated Singular Value Decomposition on the co-occurrence matrix.

2. **SVD-Sqrt**: Truncated Singular Value Decomposition on the square root of the co-occurrence matrix.

3. **SVD-Log**: Truncated Singular Value Decomposition on the natural logarithm of the co-occurrence matrix.

4. **SVD-PPMI** [89]: Singular Value Decomposition on the Positive Point-wise Mutual Information (PPMI) matrix.

5. **SVD-SPPMI** [89]: Singular Value Decomposition on the Shifted Positive Point-wise Mutual Information (SPPMI) matrix where the shift value is $-\log 5$.

6. **GloVe** [122]: For GloVe we tried 4 different values for the x_{max} parameter, $x_{max} \in \{10, 20, 50, 100\}$. We only report the results for the best setting in the comparisons, $x_{max} = 20$.

7. **Word2Vec** [101]: For Word2Vec we tried both Continuous Bag-Of-Words (CBOW) and Skip-Gram. In both cases, we tried $k \in \{5, 10\}$ as the negative sampling parameter, resulting in four different settings and reported the best results which was for Skip-Gram and $k = 10$.

8. **FastText** [19]: For FastText we tried $k \in \{5, 10\}$ as the negative sampling parameter and used the default character n-gram lengths between three and six. The best result is reported in the comparisons which is obtained by using $k = 10$.

9. **SVD-NS**: For our proposed algorithm, SVD-NS (described in chapter 4) we used PMI thresholds in $\{-6, -5, \ldots, 0\}$ and reported $\alpha = -3$ in the comparisons.

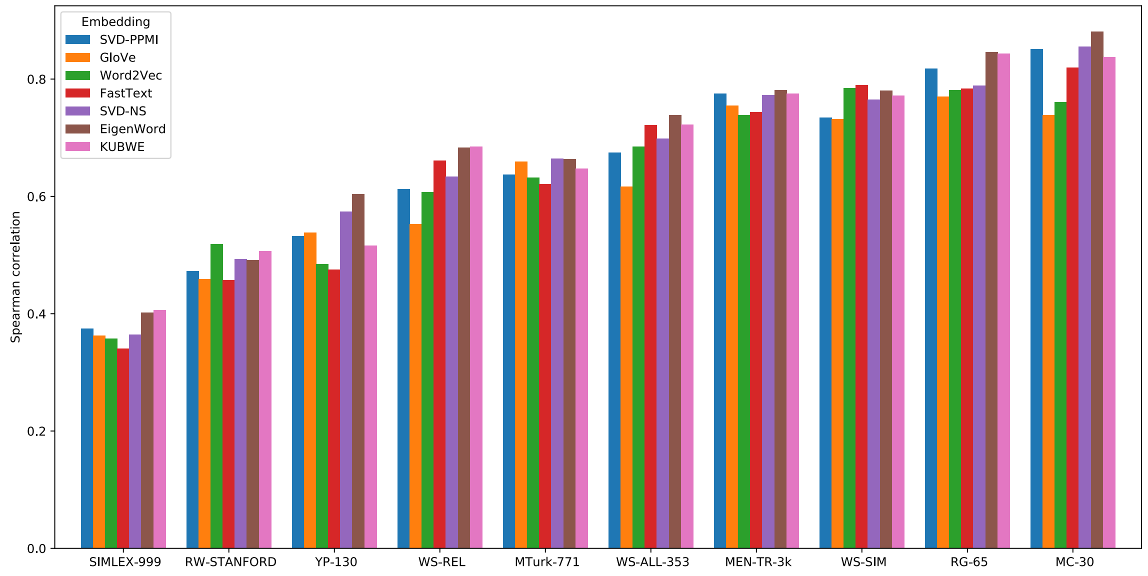


Figure 6.1: Comparison of different word embedding algorithms on 10 word similarity datasets. Y axis shows the Spearman correlation between the ground truth similarities and the similarities in the embedding space.

10. **EigenWord**: For our proposed algorithm, EigenWord (described in chapter 4), we used PMI thresholds in $\{-6, -5, \ldots, 0\}$ and reported $\alpha = -3$ in the comparisons.

11. **KUBWE**: For our proposed algorithm, KUBWE (described in chapter 3), we used kernel degrees in $\{1, 3, 5, \ldots, 21\}$ and reported $p = 13$ in the comparisons.

In our experiments, all the algorithms are trained on the exact same preprocessed input text data as explained in previous sections 6.1 and 6.2. We have also used the exact same vocabulary for all the algorithms. This will ensure a fair comparison as the size of the training data has a great influence on the quality of embeddings. The dimensionality of embeddings is 100 in all the results in this chapter.

Figure 6.1 shows the comparison of different word embedding algorithms on 10 word similarity datasets. The Spearman correlation between the human-assigned similarities and the similarities in the embedded space is used as the metric to evaluate the quality of embeddings.

Tables 6.3 and 6.4 show the evaluation of various word embedding algorithms on 10 different word similarity datasets. Numbers in the table represent the Spearman correlation between the human-assigned similarity scores and the Cosine similarity

Table 6.3: Evaluation of different word embedding algorithms on five word similarity datasets (part 1). The dimensionality of the embeddings is 100. Numbers in the table are Spearman correlations between the human-assigned similarity scores and Cosine similarities calculated in the embedding spaces.

	MC-30	**RG-65**	**WS-SIM**	**WS-REL**	**WS-ALL**
SVD	0.2839	0.4418	0.5132	0.2962	0.4099
SVD-Sqrt	0.7003	0.6672	0.7417	0.5637	0.6459
SVD-Log	0.7682	0.7153	0.7332	0.589	0.6589
SVD-PPMI	0.851	0.8179	0.7343	0.6128	0.675
SVD-SPPMI	0.8487	0.817	0.7079	0.6022	0.6595
GloVe	0.7387	0.7704	0.7316	0.5528	0.617
Word2Vec	0.7606	0.7812	0.7845	0.6073	0.6851
FastText	0.8192	0.7841	**0.7893**	0.6611	0.7216
SVD-NS	0.855	0.789	0.7649	0.6339	0.6987
EigenWord	**0.8809**	**0.8458**	0.78	0.6832	**0.7389**
KUBWE	0.8375	0.8439	0.7721	**0.6847**	0.7224

scores calculated in the embedding spaces. As we can see from the tables, our proposed algorithms perform the best in eight out of ten datasets. EigenWord algorithm proposed in chapter 4 is the best performing embedding in MC, RG, WS-ALL, MEN, and YP datasets, KUBWE proposed in chapter 3 works best in WS-REL and SimLex datasets, SVD-NS explained in chapter 4 performs the best in MTurk-771 dataset, while Word2Vec and FastText perform the best in RW-STN and WS-SIM datasets, respectively.

Table 6.5 shows the average accuracy of different word embedding algorithms over 10 word similarity datasets. The numbers in the table show the average of Spearman correlations over 10 datasets. As we can see from the table, our prosed algorithm, EigenWord, performs the best by an average Spearman of 0.6871 followed by KUBWE at 0.6712 and followed by SVD-NS at 0.6611, while the top competitor is SVD-PPMI at 0.6484. It is noteworthy to mention that FastText performs better than Word2Vec and GloVe on most datasets with an average score of 0.6414.

Our proposed algorithm, EigenWord, proved the best algorithm on word similarity task and it is better than the state-of-the-art by a margin of about 4% which is very significant.

Table 6.4: Evaluation of different word embedding algorithms on five word similarity datasets (part 2). The dimensionality of the embeddings is 100. Numbers in the table are Spearman correlations between the ground truth similarities and Cosine similarities in the embedding spaces obtained from algorithms.

	MEN	**MTurk-771**	**SimLex**	**YP Verbs**	**RW-STN**
SVD	0.3989	0.2608	0.2154	0.213	0.2876
SVD-Sqrt	0.6616	0.5575	0.2999	0.3569	0.4797
SVD-Log	0.7232	0.6394	0.3519	0.4415	0.4831
SVD-PPMI	0.7751	0.6375	0.3746	0.5324	0.473
SVD-SPPMI	0.7646	0.62	0.3577	0.5356	0.45
GloVe	0.7551	0.6594	0.3628	0.5385	0.4587
Word2Vec	0.7384	0.6318	0.358	0.485	**0.519**
FastText	0.744	0.6214	0.3408	0.4751	0.457
SVD-NS	0.7729	**0.6648**	0.3644	0.5744	0.4928
EigenWord	**0.7811**	0.6633	0.4017	**0.6041**	0.4917
KUBWE	0.7749	0.6471	**0.4059**	0.5165	0.5071

Table 6.5: Overall evaluation of different word embedding algorithms on 10 word similarity datasets. The dimensionality of the embeddings is 100. Numbers in the table show the average Spearman correlations over 10 datasets between the human-assigned similarity scores and Cosine similarities calculated in the embedding spaces.

Embedding	**Average Accuracy**
SVD	0.3321
SVD-Sqrt	0.5674
SVD-Log	0.6104
SVD-PPMI	0.6484
SVD-SPPMI	0.6363
GloVe	0.6185
Word2Vec	0.6351
FastText	0.6414
SVD-NS	0.6611
EigenWord	**0.6871**
KUBWE	0.6712

Chapter 7

Extrinsic Evaluation: Experiments with Downstream NLP Applications

In this chapter, we present the results of applying our proposed algorithms on downstream Natural Language Processing (NLP) tasks and compare them with existing approaches. We have picked two standard tasks for this purpose: 1) Emotion recognition from tweets, and 2) Sentiment analysis on IMDB movie review dataset.

This type of evaluation is often referred to as extrinsic evaluation of embeddings since it is an indirect way of measuring the quality of word embeddings. In fact, we associate the accuracy that we get on the underlying NLP task to the quality of embedding, so, the higher the accuracy we obtain, the better the embedding. We compare the quality of different word embeddings on the task by employing the same classifier and fixing all other experimental settings.

7.1 Emotion Intensity Recognition from Tweets

Emotion recognition is one of the challenging tasks in natural language processing which has attracted a lot of attention recently. The dataset we use in this section was released as part of SemEval 2018 competition, task 1: Affect In Tweets (AIT-2018) [106]. The AIT-2018 considers four fundamental emotions for the most parts including: anger, fear, joy, and sadness. This task consists of four subtasks:

1. EI-reg (emotion intensity regression): Given a tweet and an emotion E, determine the intensity of emotion E that best represents the mental state of the tweeter. The target value is a continuous number between 0 (lowest E) and 1 (highest E).

2. EI-oc (emotion intensity ordinal classification): Given a tweet and an emotion E, classify the tweet into one of four ordinal classes of intensity of emotion E

that best represents the mental state of the tweeter. The four ordinal classes are:

- 0: no E can be inferred
- 1: low amount of E can be inferred
- 2: moderate amount of E can be inferred
- 3: high amount of E can be inferred

3. V-reg (sentiment intensity regression): Given a tweet, determine the intensity of sentiment that best represents the mental state of the tweeter. The target value is a continuous number between 0 (most negative) and 1 (most positive).

4. V-oc (sentiment ordinal classification): Given a tweet, classify it into one of seven ordinal classes of intensity that best represents the mental state of the tweeter:

   - 3: very positive emotional state can be inferred
   - 2: moderately positive emotional state can be inferred
   - 1: slightly positive emotional state can be inferred
   - 0: neutral or mixed emotional state can be inferred
   - -1: slightly negative emotional state can be inferred
   - -2: moderately negative emotional state can be inferred
   - -3: very negative emotional state can be inferred

5. E-c (emotion classification): Given a tweet, classify it as 'neutral or no emotion' or as one, or more, of eleven emotions that best represent the mental state of the tweeter. The 11 emotions include anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, and trust.

All the subtasks are somewhat related both in terms of their data and the prediction task itself. They are all based on Twitter data which comes with its own challenges including inaccurate grammar, typos and shortened words, emojis and hashtags, sarcastic opinions, etc. Considering the similarity of subtasks we only report the results for the main subtask EI-reg which is emotion intensity recognition.

Table 7.1: SemEval 2018 — Task 1 — EI-reg dataset information

Emotion	Number of train tweets	Number of validation tweets	Number of test tweets
Anger	1701	388	17939
Fear	2252	389	17923
Joy	1616	290	18042
Sadness	1533	397	17912

Table 7.2: DIstant Supervision Corpus (DISC) information

Element	Count
Tweets	21,466,106
Tokens	301,221,506
Unique words (vocabulary size)	1,489,453
Common words with EI-reg	15,484
Common words with EI-reg with minimum 5 occurrence	14,213

7.1.1 Emotion Intensity Recognition (EI-reg) Dataset

As mentioned before, this task considers four emotions: anger, fear, joy, and sadness. The data for each emotion is split into train, validation, and test sets by the competition organizers. Table 7.1 shows the number of tweets for the train, validation and test sets of each emotion.

7.1.2 Corpus for training word and tweet embeddings

The competition data itself is too small to train any kind of meaningful embedding on it either for word-level or document-level embeddings. For this reason, competition organizers also released a DIstant Supervision Corpus (DISC) containing approximately 100 million tweet IDs. This distant supervision corpus was collected by querying the Twitter API for tweets that included emotion-related words such as “angry”, “annoyed”, “panic”, “happy”, “scared”, “surprised”, etc. However, we could not collect all the 100 million tweets because of the limitations on the number of tweets we could pull using the API. We managed to get a subset of the dataset containing 21,466,106 tweets and used it to train various embeddings. All word embedding algorithms as well as Doc2Vec [82] and Doc2VecC [30] algorithms in this section are trained on this corpus. Table 7.2 shows some statistics about the DISC dataset after cleaning and preprocessing.

7.1.3 Cleaning and Preprocessing of EI-reg Dataset and DISC Corpus

In order to achieve the highest similarity, consistency, and common vocabulary, we applied the exact same cleaning and preprocessing steps to both datasets. This ensures that we obtain the most meaningful embeddings and representations from the large corpus that helps solving the task the most.

Our preprocessing steps are listed below:

1. Lowercase all tweets.

2. Convert emojis to a canonical form (i.e. demojize) using Emoji library[1] [74]. Examples of standard emojis include "emoji_face_with_tears_of_joy" or "emoji-_loudly_crying_face". This is particularly useful for separating emojis in a tweet since users often use multiple emojis consecutively with no space in between.

3. Remove hashtags. Although hashtags often convey a lot of meaning about the topic in a tweet, they are not parts of the sentences and can come in any arbitrary order or position in text. Moreover, they are mostly made-up words or phrases and are often misused/overused. Therefore, we removed hashtags for simplicity.

4. Remove HTML character codes such as "&amp;", "&quot;", "&apos;", "&pound;", "&lt;", "&gt;", etc.

5. Remove user IDs and mentions from tweets (i.e. starting with @).

6. Remove links and URLs from tweets.

7. Find and fix elongated words. If any character is repeated more than two times, we replace the repeated character with exactly two occurrences of the same character. For instance, we replace "gooooood" with "good".

8. Generate a list all possible ways that emotion-related words can be written by repeating any of their characters and then standardize them. For instance, the acronym "rofl" (rolling on floor laughing) can be written as "roofl", "rofll",

[1] `https://github.com/carpedm20/emoji/`

“rrofl”, etc. and we standardize them as “rofl”. Or the word “love” can be written as “luv”, “lovee”, “lovve”, ’lovvee”, etc. Therefore, we compiled a list of approximately 50 emotion related words and generated all possible combinations of mistakes and standardized them.

9. Replace abbreviated words with their corresponding complete word. For instance, we replace “sry” with “sorry”, “bday” with “birthday”, “xmas” with “christmass”, “thx” with “thanks”, “tbh” with “to be honest”, “there’s been” with “there has been”, “we’d” with “we would”, “doesn’t” with “does not”, and so on. We compiled a list of 104 such abbreviations and standardized them.

10. Remove all special characters.

After applying all the preprocessing steps above, we lemmatized tweets using *WordNetLemmatizer* in NLTK library[2] [94] and tokenized tweets using spaCy library[3] [62].

7.1.4 Results of Traditional Methods and Document Embeddings

In this section, we present the results of methods that encode a document, tweet in this case, into a single vector. This includes traditional algorithms such as Bag-Of-Words (BOW) [133] and Term Frequency Inverse Document Frequency (TF-IDF) [145] and also some modern document embedding approaches such as Doc2Vec [82]. In particular, we have used bag of unigrams, bag of unigrams and bigrams, TFIDF of unigrams, TFIDF of unigrams and bigrams, Doc2Vec, and Doc2VecC [30]. For the Doc2Vec algorithm we used the implementation available in GenSim library[4] [126]. For the Doc2VecC we used the original implementation by the authors[5]. Doc2Vec and Doc2VecC algorithms in this section are trained on the DISC corpus explained in section 7.1.2 with embedding dimensionality of 100.

Each of the aforementioned methods provides a single vector for each tweet which we use as input features for those tweets. We then use a classification/regression algorithm to learn the task (i.e. mapping between the input features and the ground-truth

[2]`https://www.nltk.org/`
[3]`https://spacy.io/`
[4]`https://github.com/RaRe-Technologies/gensim`
[5]`https://github.com/mchen24/iclr2017`

Table 7.3: Comparison of different feature extraction methods and different regression algorithms on "anger" emotion of EI-reg dataset. For each feature extraction method, the first row shows the mean and standard deviation of Pearson correlation (%) between the predicted emotion intensity values and the ground-truth intensity values, and the second row shows the p-value of t-test against the best performing feature extraction using the same regression algorithm.

	RF	**SVR**	**KNN**	**LASSO**
BOW (unigram)	60.6 ± 2.99	52.68 ± 3.91	30.83 ± 5.44	43.58 ± 18.0
	0.0884	1e-18	1e-53	1e-9
BOW (uni + bi)	60.77 ± 2.95	59.51 ± 2.89	29.2 ± 4.87	43.72 ± 18.08
	0.1447	0.244	1e-59	1e-9
TFIDF (unigram)	58.77 ± 3.21	**60.2 ± 3.05**	50.51 ± 3.21	**60.05 ± 3.26**
	1e-5	**-**	1e-21	**-**
TFIDF (uni + bi)	59.42 ± 3.08	59.76 ± 2.59	49.03 ± 3.04	59.5 ± 3.45
	0.0006	0.4385	1e-27	0.4165
Doc2Vec	32.69 ± 4.0	52.9 ± 2.56	19.57 ± 4.81	54.94 ± 3.25
	1e-62	1e-23	1e-71	1e-12
Doc2VecC	**61.69 ± 3.28**	57.41 ± 3.05	**57.4 ± 2.51**	58.13 ± 2.89
	-	1e-5	**-**	0.0024

label). For the final regression task, we used Random Forests (RF) regression [22], Support Vector Regression (SVR) [41], K-Nearest Neighbors (KNN) regression [3], and Least Absolute Shrinkage and Selection Operator (LASSO) regression [152]. We used 50 times repeated 5-fold cross-validation for all the experiments below and report the mean and standard deviation of accuracies to makes sure that results are fair and unbiased. Moreover, we use t-test as a statistical significance test to compare different embedding algorithms.

Tables 7.3, 7.4, 7.5, and 7.6 show the results of different regression algorithms applied on different feature spaces for emotions anger, fear, joy, and sadness, respectively. Numbers in the tables show the mean and standard deviation of Pearson correlation between the predicted emotion intensity and the ground-truth emotion intensity, as well as the p-value of t-test against the best performing feature extraction method. T-test in done separately for each column (i.e. column-wise), in order to compare the statistical significance for the feature extraction methods. Mean and standard deviations of Pearson correlations are converted into percentages (i.e. multiplied by 100) to save space.

As we can see from Table 7.3, random forests and k-nearest neighbors regression

Table 7.4: Comparison of different feature extraction methods and different regression algorithms on "fear" emotion of EI-reg dataset. For each feature extraction method, the first row shows the mean and standard deviation of Pearson correlation (%) between the predicted emotion intensity values and the ground-truth intensity values, and the second row shows the p-value of t-test against the best performing feature extraction using the same regression algorithm.

	RF	**SVR**	**KNN**	**LASSO**
BOW (unigram)	61.05 ± 3.24	55.73 ± 3.46	34.21 ± 4.69	56.73 ± 4.02
	1e-5	1e-34	1e-53	1e-24
BOW (uni + bi)	61.15 ± 3.21	62.1 ± 3.4	30.18 ± 4.22	56.61 ± 4.0
	0.0001	1e-13	1e-61	1e-25
TFIDF (unigram)	59.89 ± 3.1	66.26 ± 2.69	53.1 ± 2.82	**66.34 ± 2.92**
	1e-9	0.094	1e-20	**-**
TFIDF (uni + bi)	59.86 ± 3.04	**67.14 ± 2.52**	49.09 ± 4.05	65.54 ± 2.93
	1e-9	**-**	1e-27	0.1739
Doc2Vec	29.2 ± 4.66	47.01 ± 4.09	16.52 ± 4.33	48.31 ± 3.32
	1e-67	1e-50	1e-76	1e-49
Doc2VecC	**63.61 ± 2.76**	59.02 ± 3.33	**60.41 ± 3.5**	60.28 ± 3.12
	-	1e-24	**-**	1e-16

methods work best with features extracted from Doc2VecC for anger emotion, while support vector regression and LASSO work best with unigram TF-IDF features. The maximum achieved correlation for the anger emotion is by using random forests on Doc2VecC features which is 61.69%. We can say that Doc2VecC has the best overall performance on anger emotion.

From Table 7.4 we can observe that similar to anger, random forests and k-nearest neighbors regression methods work best with features extracted from Doc2VecC on fear emotion, while support vector regression and LASSO work best with TF-IDF features. And the maximum achieved correlation for the fear emotion is by using support vector regression on unigram and bigram TF-IDF features which is 67.14%. For the fear emotion, TF-IDF is the winner and has the best overall performance.

As we can see from 7.5, random forest works best with Doc2VecC features on joy emotion while support vector regression, k-nearest neighbors, and LASSO work best with TF-IDF features. For the joy emotion, TF-IDF is the clear winner with best overall performance and the maximum achieved correlation is 65.87%.

Lastly, from Table 7.6 we can observe that for the sadness emotion Doc2VecC either has the best performance or is very close to the best performance (statistically

Table 7.5: Comparison of different feature extraction methods and different regression algorithms on "joy" emotion of EI-reg dataset. For each feature extraction method, the first row shows the mean and standard deviation of Pearson correlation (%) between the predicted emotion intensity values and the ground-truth intensity values, and the second row shows the p-value of t-test against the best performing feature extraction using the same regression algorithm.

	RF	**SVR**	**KNN**	**LASSO**
BOW (unigram)	57.05 ± 3.6	55.96 ± 4.01	34.32 ± 4.71	52.1 ± 3.69
	0.1145	1e-25	1e-49	1e-27
BOW (uni + bi)	56.51 ± 3.61	62.41 ± 3.79	32.23 ± 5.43	52.55 ± 3.79
	0.0211	1e-6	1e-48	1e-25
TFIDF (unigram)	56.29 ± 3.59	64.57 ± 3.21	**57.03 ± 3.09**	**61.77 ± 2.76**
	0.0092	0.0344	**-**	**-**
TFIDF (uni + bi)	56.07 ± 3.71	**65.87 ± 2.83**	54.98 ± 3.23	60.89 ± 2.9
	0.0046	**-**	0.0016	0.1241
Doc2Vec	23.91 ± 4.63	39.88 ± 4.64	15.06 ± 6.36	42.78 ± 4.26
	1e-64	1e-56	1e-64	1e-46
Doc2VecC	**58.2 ± 3.62**	57.73 ± 3.23	56.07 ± 3.76	58.21 ± 3.64
	-	1e-24	0.1636	1e-7

not significant) regardless of the regression method. The maximum correlation in this emotion is 63.03% and is achieved by LASSO on Doc2VecC features.

From all the experiments on document-level feature extraction, we can conclude that TF-IDF is a very powerful technique and for that reason it is always used as a strong baseline in many natural language processing tasks. From the modern document embedding approaches, Doc2VecC has the best accuracy and its performance is comparable to TF-IDF. We should mention that Doc2Vec performed poorly compared to other approaches in all emotions.

7.1.5 Results of Word Embedding Based Algorithms

In this section, we present the results of different word embeddings being used in a recurrent neural network. We have used a Long Short-Term Memory (LSTM) [61] network as the learning method. We have used the same neural network with the exact same architecture and hyper-parameters for all the embeddings to make sure we have a fair comparison.

We did not do exhaustive architecture and hyper-parameter search for the LSTM as it was not the intent of the experiment. However, we followed common best

Table 7.6: Comparison of different feature extraction methods and different regression algorithms on “sadness” emotion of EI-reg dataset. For each feature extraction method, the first row shows the mean and standard deviation of Pearson correlation (%) between the predicted emotion intensity values and the ground-truth intensity values, and the second row shows the p-value of t-test against the best performing feature extraction using the same regression algorithm.

	RF	**SVR**	**KNN**	**LASSO**
BOW (unigram)	61.02 ± 2.69	51.55 ± 3.62	34.44 ± 5.35	60.56 ± 3.1
	0.51	1e-30	1e-42	0.0001
BOW (uni + bi)	**61.39 ± 2.87**	62.12 ± 2.9	25.01 ± 5.91	60.19 ± 3.07
	-	0.6157	1e-54	1e-6
TFIDF (unigram)	60.6 ± 3.0	59.3 ± 3.12	53.54 ± 2.87	62.26 ± 2.89
	0.1831	1e-6	0.006	0.1761
TFIDF (uni + bi)	60.21 ± 2.92	**62.42 ± 2.96**	48.38 ± 3.55	61.24 ± 3.11
	0.0435	**-**	1e-17	0.0032
Doc2Vec	34.45 ± 4.27	50.89 ± 3.07	23.33 ± 4.66	50.81 ± 2.81
	1e-59	1e-35	1e-63	1e-39
Doc2VecC	61.02 ± 3.06	62.4 ± 2.37	**55.2 ± 3.05**	**63.03 ± 2.82**
	0.5388	0.9684	**-**	**-**

practices in applying recurrent neural networks and after trying tens of different settings, we used the architecture described in Table 7.7. We also tried different regularization methods, dropout [147, 164], batch normalization [66, 33], bidirectional LSTM [54] and attention-based recurrent neural networks [166], but they did not show a significant difference on this task.

As for the word embedding algorithms, similar to the intrinsic evaluation we used a variety of algorithms in the comparisons. We used the exact same algorithms as listed in section 6.3. We tuned all word embedding algorithms with parameter ranges as described in section 6.3 and used the best settings for each algorithms. The best settings for the algorithms were almost identical to the ones in chapter 6 with the exception of FastText where $k = 5$ for negative sampling was better for tweet data.

Table 7.8 shows the results of different word embedding algorithms being used in an LSTM neural network to predict emotion intensity for the EI-reg task. Numbers in the table show the mean and standard deviation of Pearson correlation between the predicted emotion intensity and the ground-truth emotion intensity, as well as the p-value of t-test against the best performing word embedding. T-test in done separately for each emotion (i.e. column-wise), in order to compare the statistical

Table 7.7: The architecture and hyper-parameters of the LSTM neural network used for the EI-reg task.

Element	**Size/Info**
Layers	
LSTM	64
Fully Connected	32
Fully Connected	32
Fully Connected (prediction layer)	1
Optimizer	Adam [75]
Loss	Mean Squared Error (MSE)
Batch size	128
Epochs	80
Validation ratio	15%
Early stopping patience	10

significance for the embedding methods. Mean and standard deviations of Pearson correlations are converted into percentages (i.e. multiplied by 100) to save space. The size of embedding vectors is $d = 100$ in all cases. Experimenting with higher dimensionality up to $d \approx 300$ improves the accuracy consistently for all methods. However, due to the limited number of datapoints in EI-reg dataset and in order to avoid over-fitting of the LSTM, we used $d = 100$ and only report the results of that setting.

As we can see from the Table 7.8, our proposed algorithms work best in all emotions and are significantly better than Word2Vec and GloVe and others. For the anger emotion, KUBWE is the best performing one with 69.34% Pearson correlation. For the fear and sadness emotions, SVD-NS is the best performing one with 72.51% and 71.8% Pearson correlations, respectively. Finally, for the joy emotion, EigenWord is the best performing method with 68.19% Pearson correlation.

Please note that FastText also works better than Word2Vec and GloVe since it considers the character n-grams as well when constructing the vector space. Another important point to mention is that word embedding based methods improve the accuracy on the underlying NLP task significantly. For instance, the average of best accuracies over all emotions achieved by traditional methods (e.g. BOW and TF-IDF) and document embedding methods (e.g. Doc2Vec and Doc2VecC) is approximately 64.43% while using the word embedding features in LSTM we achieved an average of

Table 7.8: Comparison of different word embedding algorithms used in the LSTM neural network for emotion intensity recognition. For each embedding method, the first row shows the mean and standard deviation of Pearson correlation (%) between the predicted emotion intensity values and the ground-truth intensity values, and the second row shows the p-value of t-test against the best performing embedding on the same emotion.

	Anger	**Fear**	**Joy**	**Sadness**
SVD	37.97 ± 4.3	40.54 ± 5.32	50.39 ± 4.64	37.57 ± 5.31
	1e-67	1e-61	1e-42	1e-65
SVD-Sqrt	53.73 ± 3.66	60.75 ± 3.48	57.7 ± 3.92	57.05 ± 2.78
	1e-45	1e-36	1e-29	1e-51
SVD-Log	65.03 ± 3.04	69.4 ± 2.52	65.35 ± 3.51	69.33 ± 2.41
	1e-12	1e-9	1e-6	1e-7
SVD-PPMI	65.69 ± 2.48	71.31 ± 2.35	64.21 ± 3.76	70.51 ± 2.06
	1e-11	0.0137	1e-8	0.0021
SVD-SPPMI	60.74 ± 2.61	67.39 ± 2.77	57.89 ± 3.82	65.58 ± 2.55
	1e-31	1e-16	1e-29	1e-24
GloVe	66.64 ± 3.1	70.2 ± 2.66	65.63 ± 3.4	69.85 ± 2.75
	1e-6	1e-5	1e-5	0.0001
Word2Vec	67.32 ± 2.66	71.4 ± 2.11	65.69 ± 3.53	70.49 ± 2.29
	0.0001	0.0157	0.0001	0.0032
FastText	67.96 ± 2.46	71.53 ± 2.47	66.49 ± 3.48	70.68 ± 2.07
	0.005	0.0468	0.006	0.0077
SVD-NS	68.31 ± 2.64	**72.51 ± 2.42**	67.94 ± 2.77	**71.8 ± 2.04**
	0.0421	**-**	0.645	**-**
EigenWord	68.18 ± 2.82	71.65 ± 2.49	**68.19 ± 2.48**	71.07 ± 2.06
	0.0279	0.0819	**-**	0.0789
KUBWE	**69.34 ± 2.35**	71.73 ± 2.42	67.21 ± 3.39	71.46 ± 2.42
	-	0.1105	0.1037	0.4527

70.46% over all emotions which is about 6% better.

We should also mention that 70.46% on average is not state-of-the-art on the EI-reg task and the winning teams have achieved accuracies as high as 76%. However, here we did not optimize everything in order to achieve the highest possible accuracy since the intention was just to compare the word embeddings. For example, for the joy emotion Convolutional Neural Networks (CNN) work better than LSTM, but we used the same LSTM network for consistency. Moreover, the winning teams have done a lot of preprocessing and feature extraction from emojis, hashtags, etc. Additionally, a lot of teams have used other external tools such as AffectiveTweets[6] in order to extract lexical features from tweets. Furthermore, concatenating different types of features and performing ensemble learning as well as using higher dimensionality embeddings improve the accuracy on this task. However, we only used the word embedding features since the main point of the experiment was to compare the quality of word embeddings in a downstream NLP task.

7.2 Sentiment Analysis of IMDB Movie Reviews

Sentiment analysis is one of the core and fundamental tasks in natural language processing. It is being used as a standard task to compare various classifiers, neural networks, attention-based methods, embeddings, etc. Internet Movie DataBase (IMDB) movie reviews [98] is the most well-known dataset for sentiment analysis and has been used as a standard benchmark by many researchers. In this binary classification task, the algorithm has to classify movie reviews into either positive or negative category based on its sentiment.

In this section, we use IMDB movie reviews dataset and train a neural network using different pre-trained embeddings to do the sentiment analysis. We use the same neural network and associate the accuracy of sentiment analysis with the word embeddings being used.

7.2.1 IMDB Movie Reviews Dataset

The IMDB dataset [98] contains 100,000 movie reviews, 50,000 of which are labeled into positive/negative. The other 50,000 unlabeled instances can be used for unsupervised training of algorithms. From the 50,000 labeled reviews, 25,000 are for training and 25,000 for testing. Both training and testing sets are perfectly balanced with 12,500 positive and 12,500 negative reviews. Table 7.9 shows the basic information about this dataset and the distribution of labels in the train and test sets.

There may be more than one review per movie with a maximum limitation of 30 reviews per movie. Since multiple reviews for the same movie can be correlated, the dataset is prepared in such a way that all the reviews for the same movie are either

[6]`https://github.com/felipebravom/AffectiveTweets`

Table 7.9: IMDB movie reviews dataset information

	Positive	**Negative**	**Unlabeled**	**Total**
Train	12,500	12,500	-	25,000
Test	12,500	12,500	-	25,000
Unsupervised (Train)	-	-	50,000	50,000
Total	25,000	25,000	50,000	100,000

in the training or the test set with no overlap. Consequently, the train and test sets cannot be mixed together and used for cross-validation. In our experiments, we also made sure that we leave the test data untouched at all times during training.

7.2.2 Results of Traditional Methods and Document Embeddings

In this section, we present the results of methods that encode a document, movie review in this case, into a single vector. This includes traditional algorithms such as Bag-Of-Words (BOW) [133] and Term Frequency Inverse Document Frequency (TF-IDF) [145] and also some modern document embedding approaches such as Doc2Vec [82]. We have used the same algorithms as described in section 7.1.4 that include bag of unigrams, bag of unigrams and bigrams, TFIDF of unigrams, TFIDF of unigrams and bigrams, Doc2Vec, and Doc2VecC [30]. For the Doc2Vec algorithm we used the implementation available in GenSim library[7] [126]. For the Doc2VecC we used the original implementation by the authors[8]. Doc2Vec and Doc2VecC algorithms in this section are trained on all the 100,000 reviews of the IMDB dataset with the embedding dimensionality of 100.

Each of the aforementioned methods provides a single vector for each movie review which is then used as input features for those reviews. We then train a classifier to classify the sentiment and learn the mapping between the input features and the ground-truth label. As for the classifiers, we used Random Forests (RF) classification [22], Support Vector Machines (SVM) [35] with linear kernel, K-Nearest Neighbors (KNN) classification [34], and Naive Bayes (NB) classification [128]. Naive Bayes classifiers have different variants suitable for different types of input. We used Gaussian Naive Bayes algorithm [68] for the Doc2Vec and Doc2VecC embeddings

[7]`https://github.com/RaRe-Technologies/gensim`

[8]`https://github.com/mchen24/iclr2017`

which consist of continuous-valued features and contain both positive and negative numbers. For the BOW and TF-IDF feature vectors we used Multinomial Naive Bayes algorithm [73] which has been proven to work well on count vectors and for text classification tasks.

Since this is a binary classification task, we use probabilistic output predictions from different classifiers and take into account various evaluation metrics including accuracy (i.e. Correct Classification Rate), Area Under the Curve (AUC), Precision, Recall, and F1 measure. Some algorithms, such as KNN or Naive Bayes can inherently provide probabilistic output while some others cannot. For instance, the linear SVM from LIBLINEAR library that we used [46] does not support probabilistic predictions. However, certain probability calibration techniques can be used to turn the output of any classifier into a probabilistic output [176, 177, 116] such as the Platt's algorithm for SVMs [125].

We did not use cross-validation on this task due to having multiple reviews for many movies which are most often correlated. Instead, we used the train-test approach where the test data is always the same, but we randomly sampled 80% of the training data in each run. We repeated this train-test validation 25 times for all the experiments below and report the mean and standard deviation of accuracies to makes sure that results are fair and unbiased. Moreover, we use t-test as a statistical significance test to compare different embedding algorithms.

Tables 7.10, 7.11, and 7.12 show the results of different classification algorithms applied on different feature spaces evaluated by accuracy, Area Under the Curve (AUC), and F1 measure, respectively. Numbers in the tables show the mean and standard deviation of evaluation metric, as well as the p-value of t-test against the best performing feature extraction method. T-test in done separately for each column (i.e. column-wise), in order to compare the statistical significance for the feature extraction methods. Mean and standard deviations of evaluation metrics are converted into percentages (i.e. multiplied by 100) to save space.

As we can see from these tables, random forests and k-nearest neighbors classification methods work best with features extracted from Doc2VecC algorithm, while support vector machines and naive Bayes work best with TF-IDF features. Adding

Table 7.10: Comparison of accuracies for different feature extraction methods and different classification algorithms on the IMDB dataset. For each feature extraction method, the first row shows the mean and standard deviation of accuracy, and the second row shows the p-value of t-test against the best performing feature extraction using the same classification algorithm.

	RF	**SVM**	**NB**	**KNN**
BOW (unigram)	81.39 ± 0.25	85.35 ± 0.11	81.17 ± 0.13	64.54 ± 0.17
	1e-30	1e-62	1e-50	1e-84
BOW (uni + bi)	81.91 ± 0.23	88.24 ± 0.08	85.41 ± 0.12	62.74 ± 0.41
	1e-22	1e-29	1e-17	1e-72
TFIDF (unigram)	80.41 ± 0.22	87.11 ± 0.09	82.74 ± 0.26	69.49 ± 0.21
	1e-42	1e-50	1e-39	1e-73
TFIDF (uni + bi)	80.97 ± 0.34	**88.77 ± 0.08**	**86.35 ± 0.35**	69.7 ± 0.82
	1e-30	**-**	**-**	1e-48
Doc2Vec	76.17 ± 0.13	81.68 ± 0.06	62.0 ± 0.54	70.07 ± 0.23
	1e-70	1e-84	1e-70	1e-71
Doc2VecC	**82.83 ± 0.12**	84.8 ± 0.05	75.2 ± 0.08	**80.6 ± 0.15**
	-	1e-73	1e-66	**-**

bigrams generally improves the results since it helps in identifying phrases, collocations, and sometimes disambiguation. The maximum accuracy and AUC are achieved by using support vector machines on TF-IDF of unigrams and bigrams features which are 88.77% (Table 7.10) and 95.14% (Table 7.11), respectively. The results of SVM on TF-IDF features is significantly better than all other methods.

From all the experiments on document-level feature extraction, we can conclude that TF-IDF is a very powerful technique and for that reason it is often used as a strong baseline in many natural language processing tasks. Specially for the sentiment analysis, linear SVM on TF-IDF is the best approach to try first. From the modern document embedding approaches, Doc2VecC has the best accuracy, though its performance is not comparable to that of TF-IDF.

Figure 7.1 illustrates the Receiver Operating Characteristic (ROC) diagram of the four classifiers on all six input feature spaces. For the support vector machines and random forests where the performances on some feature spaces are similar, a portion of the diagram is zoomed to make the comparison clearer. The ROC diagrams also confirm our findings explained before. By comparing the four ROC plots we can clearly see that SVM and Naive Bayes work better than random forests and KNN for sentiment analysis task. Also, we can see that TF-IDF and Doc2VecC are the best

Table 7.11: Comparison of AUC values for different feature extraction methods and different classification algorithms on the IMDB dataset. For each feature extraction method, the first row shows the mean and standard deviation of AUC, and the second row shows the p-value of t-test against the best performing feature extraction using the same classification algorithm.

	RF	**SVM**	**NB**	**KNN**
BOW (unigram)	89.61 ± 0.16	92.5 ± 0.09	89.04 ± 0.08	70.47 ± 0.25
	1e-34	1e-65	1e-78	1e-83
BOW (uni + bi)	89.97 ± 0.22	94.59 ± 0.04	92.4 ± 0.07	68.25 ± 0.53
	1e-21	1e-48	1e-58	1e-70
TFIDF (unigram)	88.59 ± 0.17	94.15 ± 0.04	91.58 ± 0.07	76.27 ± 0.24
	1e-45	1e-58	1e-66	1e-75
TFIDF (uni + bi)	89.09 ± 0.28	**95.14 ± 0.02**	**94.33 ± 0.06**	77.24 ± 0.5
	1e-32	**-**	**-**	1e-59
Doc2Vec	84.19 ± 0.11	89.01 ± 0.03	74.08 ± 0.77	77.52 ± 0.27
	1e-75	1e-98	1e-63	1e-71
Doc2VecC	**90.72 ± 0.07**	92.1 ± 0.02	83.0 ± 0.08	**88.28 ± 0.06**
	-	1e-89	1e-93	**-**

representations for the input data in this task.

7.2.3 Results of Word Embedding Based Algorithms

In this section, we present the results of different word embeddings being used in a neural network for sentiment analysis. We have used a complex architecture which consists of a mixture of Convolutional Neural Networks (CNN) [78] and Long Short-Term Memory (LSTM) [61] networks. We have used the same neural network with the exact same architecture and hyper-parameters for all the embeddings to make sure we have a fair comparison.

We did not do exhaustive architecture and hyper-parameter search for the network as it was not the intent of the experiment. However, we followed common best practices in applying convolutional and recurrent neural networks and after trying tens of different settings, we used the architecture described in Table 7.13. We also tried bidirectional LSTM [54] networks, but it did not show a significant difference on this task. However, unlike the emotion recognition task explained in section 7.1, attention-based recurrent neural networks [166] improved the results on sentiment analysis and therefore we used the attention-based network in the experiments of this section.

Table 7.12: Comparison of F1 measures for different feature extraction methods and different classification algorithms on the IMDB dataset. For each feature extraction method, the first row shows the mean and standard deviation of F1 measure, and the second row shows the p-value of t-test against the best performing feature extraction using the same classification algorithm.

	RF	**SVM**	**NB**	**KNN**
BOW (unigram)	81.38 ± 0.25	85.35 ± 0.11	81.09 ± 0.13	64.34 ± 0.19
	1e-30	1e-62	1e-49	1e-82
BOW (uni + bi)	81.91 ± 0.23	88.24 ± 0.08	85.39 ± 0.13	62.41 ± 0.52
	1e-22	1e-29	1e-16	1e-68
TFIDF (unigram)	80.4 ± 0.22	87.11 ± 0.09	82.66 ± 0.28	69.45 ± 0.23
	1e-42	1e-50	1e-38	1e-72
TFIDF (uni + bi)	80.97 ± 0.34	**88.77 ± 0.08**	**86.31 ± 0.36**	69.27 ± 1.03
	1e-30	**-**	**-**	1e-44
Doc2Vec	76.16 ± 0.13	81.68 ± 0.06	60.33 ± 0.61	69.95 ± 0.25
	1e-70	1e-84	1e-70	1e-70
Doc2VecC	**82.83 ± 0.12**	84.8 ± 0.05	75.19 ± 0.08	**80.6 ± 0.15**
	-	1e-73	1e-65	**-**

During the experiments, we found out that LSTM networks alone do not work well on very long sequences. In the IMDB dataset, the length of the reviews varies from 8 to 2371 words with an average of 263 words per review. One of the reasons that we added the convolutional and pooling layers was to shorten the sequence lengths before feeding them to the recurrent network. Convolutional layers have limited field-of-view and analyze the sentences in short sections and therefore learn n-gram-like features from the input sequence. And the max pooling layers identify and select the most important phrases or n-grams to be processed. We also tried dilated convolutional layers [174, 95] and convolutional layer with different strides instead of pooling, but the standard convolution with max pooling turned out to be the most effective in this task.

For the word embedding based experiments in this section, we used the English Wikipedia dump of April 2018 as described in section 6.1 with the same preprocessing steps explained in section 6.2. In fact, we used the exact same pre-trained embeddings that we obtained in chapter 6 and did not tune them for the IMDB task. Therefore, the list of algorithms and their best parameter settings are the same as the ones listed in section 6.3.

Table 7.14 shows the results of different word embedding algorithms being used

Table 7.13: The architecture and hyper-parameters of the neural network used for the IMDB sentiment analysis task.

Element	**Size/Info**
Layers	
Convolution 1D (kernel=3)	64
Max pooling 1D	pool size=2
Convolution 1D (kernel=3)	64
Max pooling 1D	pool size=2
Attention-based LSTM [166]	64
Fully Connected	64
Fully Connected	32
Fully Connected (prediction layer)	1
Optimizer	Adam [75]
Loss	Mean Squared Error (MSE)
Batch size	Dynamic between 256 and 512
Epochs	100
Validation ratio	15%
Early stopping patience	10

in our CNN-LSTM neural network (described in Table 7.13) to predict the sentiment for the IMDB movie reviews task. We used the same 25 times repeated train-test evaluation strategy explained in section 7.2.2. Numbers in the table show the mean and standard deviation of accuracy, AUC, and F1 measure, as well as the p-value of t-test against the best performing word embedding. T-test in done separately for each evaluation metric (i.e. column-wise), in order to compare the statistical significance for the embedding methods. Mean and standard deviations of evaluation metrics are converted into percentages (i.e. multiplied by 100) to save space. The size of embedding vectors is $d = 100$ in all cases.

As we can see from the Table 7.14, our proposed algorithms work best with regard to all evaluation metrics and are significantly better than Word2Vec, GloVe, FastText and others. The proposed algorithm KUBWE with 88.75% accuracy is the best performing one, followed by SVD-NS with 88.66% accuracy followed by EigenWord with 88.53% accuracy.

Please note that FastText also works better than Word2Vec and GloVe since it considers the character n-grams as well when constructing the vector space. Another point to mention is that unlike the EI-Reg task where the embedding based methods were significantly better than traditional methods, word embedding based methods

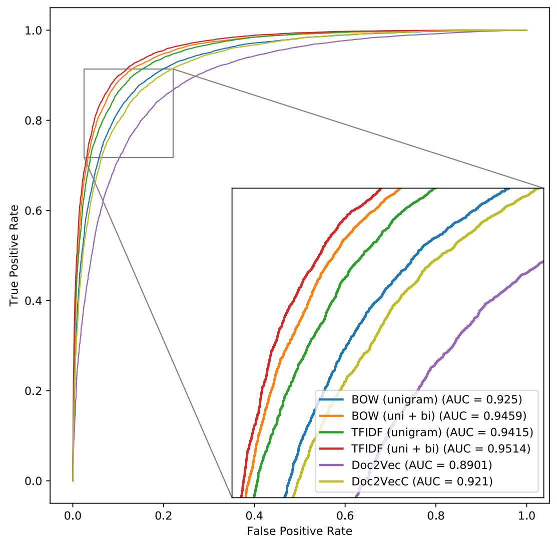
Table 7.14: Comparison of different word embedding algorithms used in our CNN-LSTM neural network for sentiment analysis. For each embedding method, the first row shows the mean and standard deviation of evaluation metrics (%), and the second row shows the p-value of t-test against the best performing embedding with respect to the same evaluation metric.

	Accuracy	**AUC**	**F1**
SVD	75.75 ± 0.44	83.61 ± 0.37	75.72 ± 0.46
	1e-59	1e-64	1e-59
SVD-Sqrt	78.89 ± 0.34	86.85 ± 0.3	78.86 ± 0.35
	1e-56	1e-61	1e-56
SVD-Log	87.01 ± 0.33	94.08 ± 0.19	87.01 ± 0.34
	1e-22	1e-28	1e-22
SVD-PPMI	88.19 ± 0.38	94.81 ± 0.24	88.19 ± 0.38
	1e-6	1e-11	1e-6
SVD-SPPMI	87.44 ± 0.39	94.42 ± 0.27	87.44 ± 0.4
	1e-16	1e-18	1e-16
GloVe	87.66 ± 0.29	94.75 ± 0.16	87.66 ± 0.29
	1e-15	1e-16	1e-15
Word2Vec	87.29 ± 0.33	94.43 ± 0.14	87.28 ± 0.34
	1e-19	1e-25	1e-19
FastText	87.71 ± 0.31	94.74 ± 0.18	87.71 ± 0.31
	1e-14	1e-15	1e-14
SVD-NS	88.66 ± 0.36	95.23 ± 0.24	88.66 ± 0.37
	0.4325	0.1369	0.4314
EigenWord	88.53 ± 0.32	95.18 ± 0.17	88.52 ± 0.32
	0.032	0.0039	0.0331
KUBWE	$\mathbf{88.75 \pm 0.38}$	$\mathbf{95.32 \pm 0.17}$	$\mathbf{88.74 \pm 0.38}$
	-	**-**	**-**

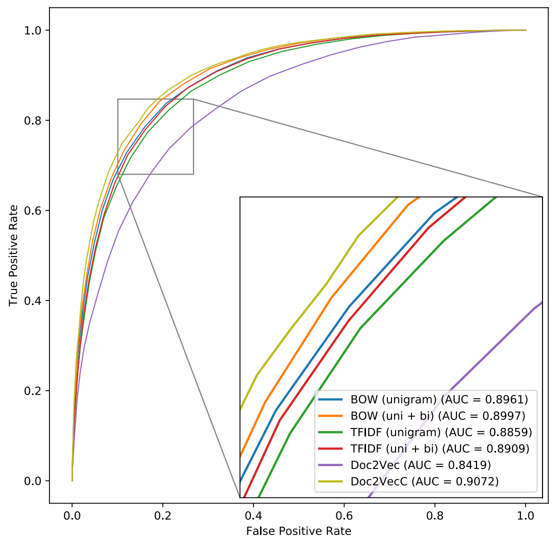
did not work significantly better than traditional methods on this sentiment analysis task. In fact, with the embedding based approach and using neural networks we achieved almost the same result as the SVM with TF-IDF features. Perhaps, further tuning of the neural network or using higher dimensional embeddings could help. However, we did not tune them further as the main point of this experiment was to compare the quality of different word embedding methods on a real world NLP application, and not to obtain the best possible accuracy on the task.

We should also mention that 88.75% accuracy is less than the state-of-the-art accuracy on the IMDB task. The state-of-the-art accuracy is about 95% by ULM-FiT [63] and block-sparse LSTM [55] followed by onehot LSTM (i.e. oh-LSTM) [69] with 94.1% accuracy. However, here we did not optimize all the hyper-parameters in order to achieve the highest possible accuracy, since the intention was to compare the word embeddings. For instance, using a larger and more appropriate corpus instead of Wikipedia for pre-training the embeddings, as well as using higher dimensional embeddings would have improved the results significantly. Furthermore, following best practices in training word embeddings [90, 81], as well as using various tricks and techniques for improving the quality of embeddings [102] can result in higher accuracies on the task. One such technique is the repeated identification and replacement of Multi-Word Expressions (MWE) [132] as described in [102]. Nevertheless, we did not use those types of techniques and kept the experimental setting simple in order to compare the quality of word embeddings on a real world NLP task.

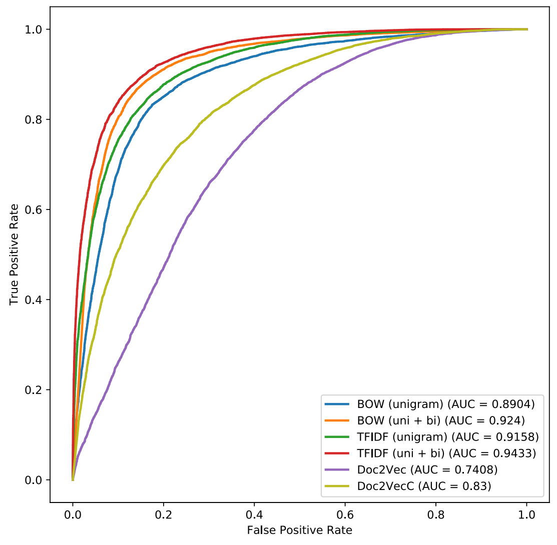
Figure 7.2 illustrates the Receiver Operating Characteristic (ROC) diagram of our CNN-LSTM neural network using various pre-trained word embeddings as input features. Part of the plot is zoomed for better clarification. As we can see from the figure, the three proposed algorithms, KUBWE, SVD-NS, and EigenWord have the best ROC curve and therefore, the highest Area Under the Curve (AUC).



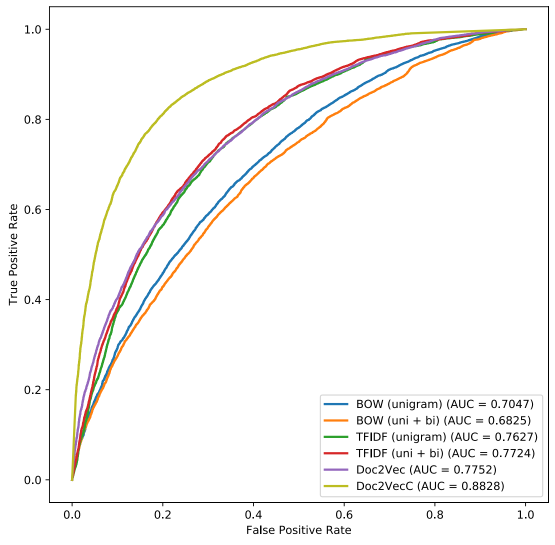
(a) Support Vector Machines (SVM)



(b) Random Forests (RF)



(c) Naive Bayes (NB)



(d) K Nearest Neighbors (KNN)

Figure 7.1: Receiver Operating Characteristic (ROC) diagram of (a) support vector machines, (b) random forests, (c) naive Bayes, and (d) k nearest neighbors classification methods on all six feature spaces.

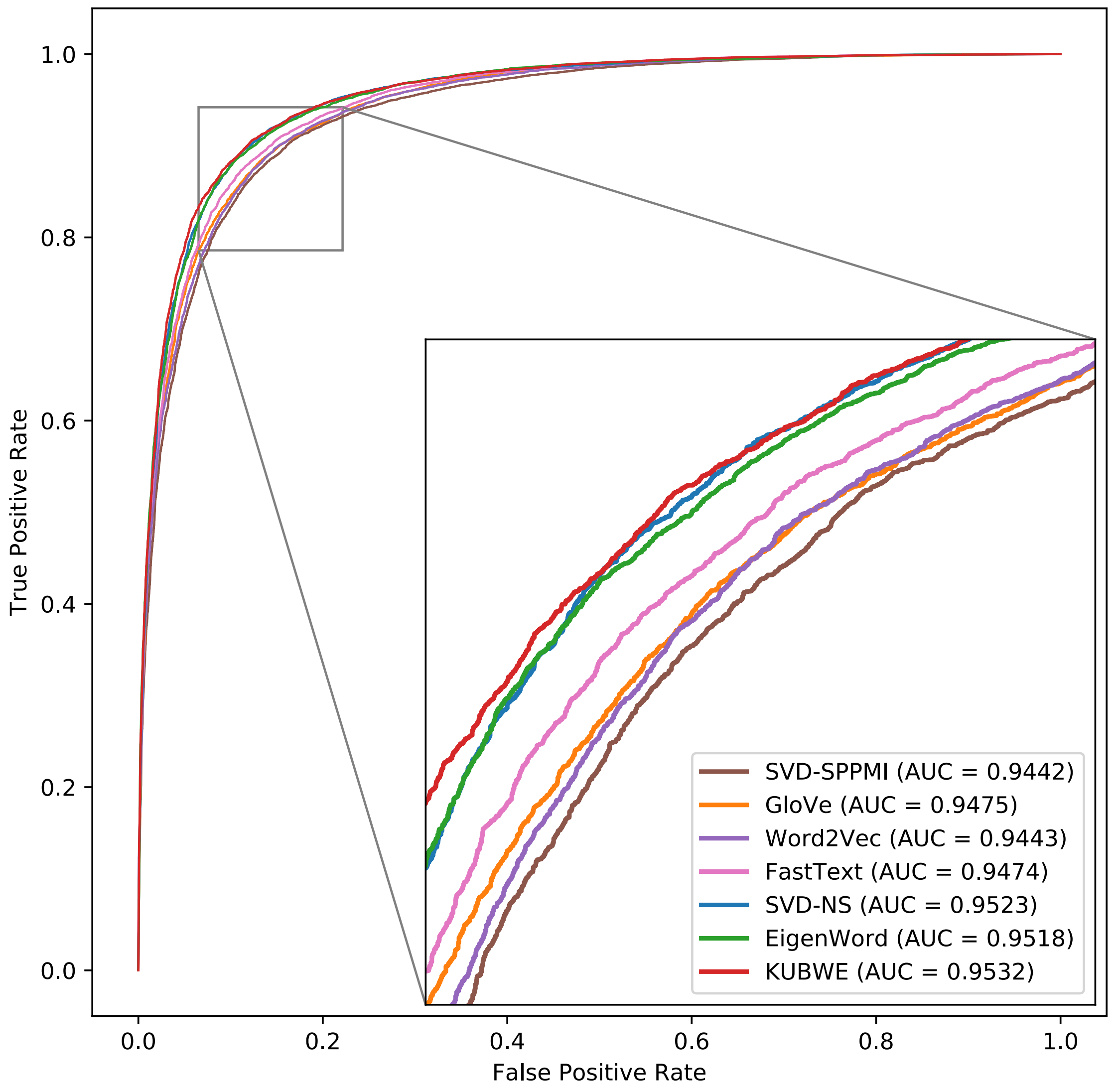


Figure 7.2: Receiver Operating Characteristic (ROC) diagram of our CNN-LSTM neural network using different word embeddings.

Chapter 8

Conclusion and Future Research

8.1 Conclusion

In this thesis, we studied embedding techniques for learning representations from text and images. We first reviewed dimensionality reduction and manifold learning methods and proposed a spherical representation learning algorithm with an intuitive objective function that learns the structure of manifolds and maximizes their separability in the embedding. This makes it suitable particularly for clustering applications. Our experiments on face, digit, and object recognition datasets showed that our proposed algorithm significantly improves the clustering quality in the embedding space.

We then extended our approach to the text domain to learn word embeddings from a corpus. In this case, we calculate the global co-occurrence matrix and refine it using Pointwise Mutual Information (PMI) matrix and use it as input to the optimization. The main advantage of our algorithm compared to other word embedding algorithms is that our approach makes use of the negative context to improve the distribution of words in the embedding.

We also analyzed frequency-based or count-based approaches and proposed a spectral word embedding method that takes into account negative examples. This information was previously ignored by almost all algorithms and we showed both theoretically and empirically that it improves the embedding quality in the frequency-based algorithms. Our proposed solutions, EigenWord and SVD-NS, have intuitive formulation, can consider co-occurrence based similarity and context-based similarity, and they have an optimal closed-form solution through eigenanalysis.

We also proposed an efficient Word Sense Disambiguation (WSD) and multi-sense embedding technique to tackle the ambiguity in natural language. Our approach uses a soft-clustering of contexts which is a natural choice since a lot of context words are common among different senses of words. We showed the effectiveness of our approach on a WSD competition data.

Finally, we evaluated our proposed word embedding techniques on two NLP tasks: emotion recognition from tweets and sentiment analysis from movie reviews. Our proposed algorithms achieved significantly higher accuracy than other embedding algorithms in both tasks.

Based on the research in this thesis and our experiments we have the following conclusions for word embeddings:

- The co-occurrence counts and consequently, the PMI matrix are great ways to represent word associations. They contain global statistics about the contextual information in the corpus and are good measures for semantic similarity.

- Calculating global context is a useful strategy since it is a more accurate way of representing context vectors. Embeddings that are based on global context can work at least as well as prediction-based embeddings that use local context.

- Negative examples are indeed important in the embeddings and appropriate use of negative context can significantly improve the quality of embedding. Pushing unrelated words away from each other is as important as placing semantically similar words close to each other. Calculating the global co-occurrence statistics enables us to accurately make use of negative examples as we did in our proposed algorithms. Prediction-based algorithms such as Word2Vec [101] that use local context simply do not have access to this valuable knowledge and therefore, have no other choice than random sampling.

- Factorization-based methods and spectral word embeddings are computationally fast compared to prediction-based algorithms. Their computational advantage comes from the fact that they work on count-aggregated data, as opposed to Word2Vec's training procedure which requires each observation to be presented separately [89]. They are also exact and have almost no hyper-parameter to tune. In this thesis, we proposed EigenWord, a spectral word embedding technique that is extremely fast and has all the advantages of frequency-based embeddings and in addition, it makes use of negative examples to improve the distribution of words in the embedding, and it has an optimal closed-form solution.

- The size of the corpus and the size of vocabulary have great a effect on the quality of embedding. Careful preprocessing of the corpus can significantly decrease the amount of unwanted words. This helps in reducing the computational cost as well as improving the embedding for the remaining words.
- Embeddings improve the accuracy on the downstream NLP tasks by a great margin. Based on our experiments, embedding-based learning of the underlying task improves the accuracy by about 5%.

8.2 Future Research

GloVe extension We showed that PMI is a very effective way to measure word associations, and that PMI-based embeddings can capture the semantic similarity of words. GloVe originally works on the log co-occurrence matrix and a natural extension would be to use PMI matrix in the optimization. Although there exist PMI-based embeddings that use objective functions similar to Glove [5], the usage of negative examples in GloVe is still unexplored. GloVe provides the best results in the word analogy task compared to all other methods, and we believe extending that to take into account the negative context can significantly improve its quality of embedding.

Negative examples in context-dependent embeddings Another possible extension is to explore the usage of negative examples in the context-based embeddings. Recently, context-dependent embedding methods such as BERT [39], ELMo [123], and CoVe [99] have caught a lot of attention and have been successfully applied in many NLP tasks. It has been shown that they improve the accuracy on almost any NLP task by a significant margin. Therefore, it would be interesting to explore the possibility of using negative context in these embeddings and improve them further.

Kernel theory In KUBWE algorithm, we proposed a kernel similarity measure for the embedding space and showed its effectiveness compared to the Cosine similarity. We used polynomial kernel, without an extensive investigation of the choice of the kernel. It would be interesting to study the kernel and its effect from a theoretical perspective and also examine various kernels. Moreover, the Cosine similarity is the main similarity function in NLP and embeddings. Therefore, should the kernel similarity be shown more effective than Cosine, then due to its easy generalization to other embeddings it would be beneficial to other algorithms.

Unobserved instead of insignificant In our spectral word embedding solution, EigenWord, we use the negative PMI values as our negative context. This might seem counter-intuitive as the negative PMI correspond to word pairs that have co-occurred in the corpus, though insignificantly. However, the zeros are the ones that are unobserved and it makes sense to sample from zeros for negative context. We have done some experiments to sample from zeros and replace them by random negative values and did not observe a significant differencein the quality of embedding. It would be insightful to study the effect of random sampling in spectral methods.

WSD extension In our multi-sense embedding approach, we analyze and disambiguate each ambiguous word separately and expand their context vector into a context matrix of senses. This creates a lot of redundancy as there many words that have a common meaning. In fact, there are a lot of highly correlated senses in our sense-word co-occurrence matrix. One possible extension to the approach is to learn the latent senses simultaneously. In other words, one can learn the distribution of words over the global pool of senses/concepts at once. This will improve the computational time and reduce the redundancy.

Out-Of-Vocabulary (OOV) extension Currently our proposed embedding algorithms cannot handle out-of-vocabulary words. A valuable extension to our work is to generalize it to OOV words. Right now, there are very limited number of algorithms with OOV capability such as FastText [19] and Char2Vec [27]. Let's assume the embedding is already learned from the corpus. For a new OOV word, if we calculate its similarity to the existing words (possibly few words), then one can formulate the OOV extension as a Least Squares (LS) minimization problem to the current vector space and obtain the embedding for the new word. This is a vital extension to almost any embedding method and provides practical benefits to all pre-trained embeddings.

More complicated NLP tasks In this thesis, we applied the pre-trained embeddings on word similarity, word analogy, emotion recognition from tweets and sentiment analysis from movie reviews. It would be interesting to see its application on more complicated NLP tasks such as Natural Language Inference (NLI) and also compare them with context-dependent embeddings.

Bibliography

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer, 2001.

[2] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

[3] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[4] LNF Ana and Anil K Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003.

[5] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to PMI-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[6] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association of Computational Linguistics*, 6:483–495, 2018.

[7] Giuseppe Attardi. Wikiextractor: A tool for extracting plain text from wikipedia dumps, 2009.

[8] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[9] Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010.

[10] Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial Intelligence and Statistics*, pages 130–138, 2016.

[11] Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):711–720, 1997.

[12] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[13] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[15] Yoshua Bengio and Martin Monperrus. Non-local manifold tangent learning. *Advances in Neural Information Processing Systems*, 17(1):129–136, 2005.

[16] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. *Neural Probabilistic Language Models*, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[17] CM Bishop. Pattern recognition and machine learning. *Springer, New York*, 2006.

[18] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[20] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[21] Christos Boutsidis and Efstratios Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[22] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[23] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[24] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics, 2012.

[25] John A Bullinaria and Joseph P Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, 2007.

[26] Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

[27] Kris Cao and Marek Rei. A joint model for word embedding and word morphology. *arXiv preprint arXiv:1606.02601*, 2016.

[28] Miguel A Carreira-Perpinán. The elastic embedding algorithm for dimensionality reduction. In *ICML*, volume 10, pages 167–174, 2010.

[29] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[30] Minmin Chen. Efficient vector representation for documents through corruption. In *5th International Conference on Learning Representations (ICLR)*, 2017.

[31] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[32] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[33] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

[34] Danny Coomans and Désiré Luc Massart. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules. *Analytica Chimica Acta*, 136:15–27, 1982.

[35] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[36] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.

[37] Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Technical report, Stanford University, 2004.

[38] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[40] David L Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[41] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[42] Dheeru Dua and Casey Graff. UCI machine learning repository, 2019.

[43] Harry Dym. *Linear algebra in action*, volume 78. American Mathematical Soc., 2013.

[44] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, 2009.

[45] Samuel G Fadel, Francisco M Fatore, Felipe SLG Duarte, and Fernando V Paulovich. Loch: A neighborhood-based multidimensional projection technique for high-dimensional sparse spaces. *Neurocomputing*, 150:546–556, 2015.

[46] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[47] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. pages 1606–1615, 2015.

[48] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[49] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic time. *ACM Trans. Math. Software*, 3(SLAC-PUB-1549-REV. 2):209–226, 1976.

[50] Johannes Fürnkranz. A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, 3(1998):1–10, 1998.

[51] Athinodoros S Georghiades, Peter N Belhumeur, and David J Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):643–660, 2001.

[52] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *Advances in neural information processing systems*, pages 451–458, 2005.

[53] Daniel B Graham and Nigel M Allinson. Characterising virtual eigensignatures for general purpose face recognition. In *Face Recognition: From Theory to Applications*, pages 446–456. Springer, 1998.

[54] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[55] Scott Gray, Alec Radford, and Diederik P Kingma. GPU kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 2017.

[56] Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM, 2012.

[57] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[58] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2016.

[59] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 833–840, 2002.

[60] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[61] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[62] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2017.

[63] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[64] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[65] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[66] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[67] Xiang Jiang, Erico N de Souza, Xuan Liu, Behrouz Haji Soleimani, Xiaoguang Wang, Daniel L Silver, and Stan Matwin. Partition-wise recurrent neural networks for point-based ais trajectory classification. In *25th European Symposium on Artificial Neural Networks (ESANN), Computtional Intelligence and Machine Learning*, volume 6, pages 529–534, 2017.

[68] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[69] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373*, 2016.

[70] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[71] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[72] Yi-Hao Kao and Benjamin Van Roy. Learning a factor model via regularized pca. *Machine learning*, 91(3):279–303, 2013.

[73] Ashraf M Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence*, pages 488–499. Springer, 2004.

[74] Taehoon Kim and Kevin Wurster. Emoji python library.

[75] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[76] Effrosini Kokiopoulou, Jie Chen, and Yousef Saad. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18(3):565–602, 2011.

[77] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[79] Da Kuang. *Nonnegative matrix factorization for clustering*. PhD thesis, Georgia Institute of Technology, 2014.

[80] Stephane Lafon and Ann B Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1393–1403, 2006.

[81] Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.

[82] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[83] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[84] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[85] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[86] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[87] Kuang-Chih Lee, Jeffrey Ho, and David J Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):684–698, 2005.

[88] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308, 2014.

[89] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[90] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[91] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[92] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust region Newton method for logistic regression. *Journal of Machine Learning Research*, 9(Apr):627–650, 2008.

[93] Tong Lin and Hongbin Zha. Riemannian manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):796–809, 2008.

[94] Edward Loper and Steven Bird. NLTK: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[95] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pages 4898–4906, 2016.

[96] Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.

[97] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[98] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[99] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[100] S. Mika, G. Ratsch, J. Weston, B. Scholkopft, and K. R. Mullert. Fisher discriminant analysis with kernels. In *Proceedings of IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[101] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[102] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

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

[104] George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.

[105] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[106] Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, 2018.

[107] Robson Motta, Rosane Minghim, Alneu de Andrade Lopes, and Maria Cristina F Oliveira. Graph-based measures to assist user assessment of multidimensional projections. *Neurocomputing*, 150:583–598, 2015.

[108] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 142–148. Association for Computational Linguistics, 2016.

[109] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2227–2240, 2014.

[110] Habibeh Naderi, Behrouz Haji Soleimani, and Stan Matwin. Manifold learning of overcomplete feature spaces in a multimodal biometric recognition system of iris and palmprint. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 191–196. IEEE, 2017.

[111] Habibeh Naderi, Behrouz Haji Soleimani, Stan Matwin, Babak Nadjar Araabi, and Hamid Soltanian-Zadeh. Fusing iris, palmprint and fingerprint in a multi-biometric recognition system. In *2016 13th Conference on Computer and Robot Vision (CRV)*, pages 327–334. IEEE, 2016.

[112] Habibeh Naderi, Behrouz Haji Soleimani, Saif Mohammad, Svetlana Kiritchenko, and Stan Matwin. DeepMiner at semeval-2018 task 1: Emotion intensity recognition using deep representation learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 305–312. Association for Computational Linguistics (ACL), 2018.

[113] Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia object image library (coil-100). 1996.

[114] Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia object image library (coil-20). 1996.

[115] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[116] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005.

[117] X Niyogi. Locality preserving projections. In *Neural information processing systems*, volume 16, page 153. MIT, 2004.

[118] Victor Onclinx, John A Lee, Vincent Wertz, and Michel Verleysen. Dimensionality reduction by rank preservation. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[119] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM, 2002.

[120] Beresford N Parlett. *The symmetric eigenvalue problem*. SIAM, 1998.

[121] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183. Association for Computational Linguistics, 2017.

[122] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[123] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics, 2018.

[124] Matthew Peters, Mark Neumann, Wen-tau Yih, and Luke Zettlemoyer. Dissecting contextual word embeddings: Architecture and representation. In *Empirical Methods in Natural Language Processing, EMNLP*, 2018.

[125] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[126] R Rehurek and P Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[127] Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics, 2010.

[128] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[129] Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood component analysis. *Advances in Neural Information Processing Systems (NIPS)*, 17:513–520, 2004.

[130] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[131] Herbert Rubenstein and John B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[132] Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for NLP. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer, 2002.

[133] Gerard Salton. The smart retrieval system–experiments in automatic document processing. 1971.

[134] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142. IEEE, 1994.

[135] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, (5):401–409, 1969.

[136] Hinrich Schütze and Jan O Pedersen. Information retrieval based on word senses. 1995.

[137] Behdad Soleimani, Mohammad-Hassan Zokaei Ashtiani, Behrouz Haji Soleimani, and Hadi Moradi. A disaster invariant feature for localization. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1096–1101. IEEE, 2010.

[138] Behrouz Haji Soleimani, Erico N De Souza, Casey Hilliard, and Stan Matwin. Anomaly detection in maritime data based on geometrical analysis of trajectories. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 1100–1105. IEEE, 2015.

[139] Behrouz Haji Soleimani and Stan Matwin. Nonlinear dimensionality reduction by unit ball embedding (UBE) and its application to image clustering. In *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 983–988. IEEE, 2016.

[140] Behrouz Haji Soleimani and Stan Matwin. Dimensionality reduction and visualization by doubly kernelized unit ball embedding. In *Advances in Artificial Intelligence: 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018*, pages 224–230. Springer, 2018.

[141] Behrouz Haji Soleimani and Stan Matwin. Spectral word embedding with negative sampling. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5481–5487. Association for the Advancement of Artificial Intelligence (AAAI), 2018.

[142] Behrouz Haji Soleimani and Stan Matwin. Fast PMI-based word embedding with efficient use of unobserved patterns. In *Thirty-Third AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence (AAAI), 2019.

[143] Behrouz Haji Soleimani, Stan Matwin, and Erico N De Souza. A density-penalized distance measure for clustering. In *Advances in Artificial Intelligence: 28th Canadian Conference on Artificial Intelligence, Canadian AI 2015*, pages 238–249. Springer, 2015.

[144] Behrouz Haji Soleimani, Habibeh Naderi, and Stan Matwin. Efficient unsupervised word sense induction, disambiguation and embedding. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Workshop on Relational Representation Learning*, 2018.

[145] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[146] Marcus Spruill et al. Asymptotic distribution of coordinates on high dimensional spheres. *Electronic communications in probability*, 12:234–247, 2007.

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

[148] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. Spine: Sparse interpretable neural embeddings. In *AAAI Conference on Aritificial Intelligence, AAAI*, 2018.

[149] Ching Y Suen. N-gram statistics for natural language understanding and text processing. *IEEE transactions on pattern analysis and machine intelligence*, (2):164–172, 1979.

[150] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Sparse word embeddings using l1 regularized online learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI*, 2016.

[151] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[152] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[153] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[154] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry.* Chapman and Hall/CRC, 2017.

[155] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[156] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[157] Peter D Turney and Michael L Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346, 2003.

[158] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[159] L. J. P. Van Der Maaten, E. O. Postma, and H. J. Van den Herik. Dimensionality reduction: a comparative review. *Journal of Machine Learning Research*, 10:66–71, 2009.

[160] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[161] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[162] Farshid Varno, Behrouz Haji Soleimani, Marzie Saghayi, Lisa Di Jorio, and Stan Matwin. Efficient neural task adaptation by maximum entropy initialization. *arXiv preprint arXiv:1905.10698*, 2019.

[163] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[164] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[165] Jing Wang, Zhenyue Zhang, and Hongyuan Zha. Adaptive manifold learning. In *Advances in neural information processing systems*, pages 1473–1480, 2005.

[166] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[167] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[168] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

[169] Max Welling, Richard S Zemel, and Geoffrey E Hinton. Probabilistic sequential independent components analysis. *Neural Networks, IEEE Transactions on*, 15(4):838–849, 2004.

[170] Roland Winkler, Frank Klawonn, and Rudolf Kruse. Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications (IJFSA)*, 1(1):1–16, 2011.

[171] Dongqiang Yang and David MW Powers. *Verb similarity on the taxonomy of WordNet*. Masaryk University, 2006.

[172] Zhirong Yang, Irwin King, Zenglin Xu, and Erkki Oja. Heavy-tailed symmetric stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 2169–2177, 2009.

[173] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, volume 93, pages 311–321, 1993.

[174] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[175] Jin Yu, SVN Vishwanathan, Simon Günter, and Nicol N Schraudolph. A quasi-newton approach to nonsmooth convex optimization problems in machine learning. *Journal of Machine Learning Research*, 11(Mar):1145–1200, 2010.

[176] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.

[177] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM, 2002.

[178] Tianhao Zhang, Jie Yang, Deli Zhao, and Xinliang Ge. Linear local tangent space alignment and application to face recognition. *Neurocomputing*, 70(7):1547–1553, 2007.

[179] Zhen-yue Zhang and Hong-yuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *Journal of Shanghai University (English Edition)*, 8(4):406–424, 2004.

Appendix A

Copyright Permissions

This appendix includes the copyright forms for our publications in:

1. 15th IEEE International Conference on Machine Learning and Applications (ICMLA 2016) [139]
2. 31st Canadian Conference on Artificial Intelligence (Canadian AI 2018) [140]
3. 32nd AAAI Conference on Artificial Intelligence (AAAI 2018) [141]
4. 33rd AAAI Conference on Artificial Intelligence (AAAI 2019) [142]

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

Nonlinear Dimensionality Reduction by Unit Ball Embedding (UBE) and its Application to Image Clustering
Behrouz Haji Soleimani and Stan Matwin
2016 15th IEEE International Conference on Machine Learning and Applications

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Behrouz Haji Soleimani	07-10-2016
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the

Consent to Publish
Lecture Notes in Computer Science

Springer

Title of the Book or Conference Name: 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018
Volume Editor(s): Ebrahim Bagheri and Jackie Chi Kit Cheung
Title of the Contribution: Dimensionality Reduction and Visualization by Doubly Kernelized Unit Ball Embedding
Author(s) Name(s): Behrouz Haji Soleimani and Stan Matwin
Corresponding Author's Name, Address, Affiliation and Email: Behrouz Haji Soleimani, 6050 University Avenue, Halifax, Nova Scotia, Canada B3H 4R2
Faculty of Computer Science, Dalhousie University, behrouz.hajisoleimani@dal.ca

When Author is more than one person the expression "Author" as used in this agreement will apply collectively unless otherwise indicated.

§ 1 Rights Granted

Author hereby grants and assigns to Springer International Publishing AG, Cham (hereinafter called Springer) the exclusive, sole, permanent, world-wide, transferable, sub-licensable and unlimited right to reproduce, publish, distribute, transmit, make available or otherwise communicate to the public, translate, publicly perform, archive, store, lease or lend and sell the Contribution or parts thereof individually or together with other works in any language, in all revisions and versions (including soft cover, book club and collected editions, anthologies, advance printing, reprints or print to order, microfilm editions, audiograms and videograms), in all forms and media of expression including in electronic form (including offline and online use, push or pull technologies, use in databases and networks for display, print and storing on any and all stationary or portable end-user devices, e.g. text readers, audio, video or interactive devices, and for use in multimedia or interactive versions as well as for the display or transmission of the Contribution or parts thereof in data networks or seach engines), in whole, in part or in abridged form, in each case as now known or developed in the future, including the right to grant further time-limited or permanent rights. For the purposes of use in electronic forms, Springer may adjust the Contribution to the respective form of use and include links or otherwise combine it with other works. For the avoidance of doubt, Springer has the right to permit others to use individual illustrations and may use the Contribution for advertising purposes.
The copyright of the Contribution will be held in the name of Springer. Springer may take, either in its own name or in that of copyright holder, any necessary steps to protect these rights against infringement by third parties. It will have the copyright notice inserted into all editions of the Contribution according to the provisions of the Universal Copyright Convention (UCC) and dutifully take care of all formalities in this connection in the name of the copyright holder.

§ 2 Regulations for Authors under Special Copyright Law

The parties acknowledge that there may be no basis for claim of copyright in the United States to a Contribution prepared by an officer or employee of the United States government as part of that person's official duties. If the Contribution was performed under a United States government contract, but Author is not a United States government employee, Springer grants the United States government royalty-free permission to reproduce all or part of the Contribution and to authorize others to do so for United States government purposes.
If the Contribution was prepared or published by or under the direction or control of Her Majesty (i.e., the constitutional monarch of the Commonwealth realm) or any Crown government department, the copyright in the Contribution shall, subject to any agreement with Author, belong to Her Majesty.
If the Contribution was created by an employee of the European Union or the European Atomic Energy Community (EU/Euratom) in the performance of their duties, the regulation 31/EEC, 11/EAEC (Staff Regulations) applies, and copyright in the Contribution shall, subject to the Publication Framework Agreement (EC Plug), belong to the European Union or the European Atomic Energy Community.
If Author is an officer or employee of the United States government, of the Crown, or of EU/Euratom, reference will be made to this status on the signature page.

§ 3 Rights Retained by Author

Author retains, in addition to uses permitted by law, the right to communicate the content of the Contribution to other scientists, to share the Contribution with them in manuscript form, to perform or present the Contribution or to use the content for non-commercial internal and educational purposes, provided the Springer publication is mentioned as the

original source of publication in any printed or electronic materials. Author retains the right to republish the Contribution in any collection consisting solely of Author's own works without charge subject to ensuring that the publication by Springer is properly credited and that the relevant copyright notice is repeated verbatim.
Author may self-archive an author-created version of his/her Contribution on his/her own website and/or the repository of Author's department or faculty. Author may also deposit this version on his/her funder's or funder's designated repository at the funder's request or as a result of a legal obligation. He/she may not use the publisher's PDF version, which is posted on SpringerLink and other Springer websites, for the purpose of self-archiving or deposit. Furthermore, Author may only post his/her own version, provided acknowledgment is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be provided by inserting the DOI number of the article in the following sentence: "The final publication is available at Springer via http://dx.doi.org/[insert DOI]". The DOI (Digital Object Identifier) can be found at the bottom of the first page of the published paper.
Prior versions of the Contribution published on non-commercial pre-print servers like ArXiv/CoRR and HAL can remain on these servers and/or can be updated with Author's accepted version. The final published version (in pdf or html/xml format) cannot be used for this purpose. Acknowledgment needs to be given to the final publication and a link must be inserted to the published Contribution on Springer's website, by inserting the DOI number of the article in the following sentence: "The final publication is available at Springer via http://dx.doi.org/[insert DOI]".
Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgment is given to the original source of publication. Author also retains the right to use, without having to pay a fee and without having to inform the publisher, parts of the Contribution (e.g. illustrations) for inclusion in future work, and to publish a substantially revised version (at least 30% new content) elsewhere, provided that the original Springer Contribution is properly cited.

§ 4 Warranties

Author warrants that the Contribution is original except for such excerpts from copyrighted works (including illustrations, tables, animations and text quotations) as may be included with the permission of the copyright holder thereof, in which case(s) Author is required to obtain written permission to the extent necessary and to indicate the precise sources of the excerpts in the manuscript. Author is also requested to store the signed permission forms and to make them available to Springer if required.
Author warrants that he/she is entitled to grant the rights in accordance with Clause 1 "Rights Granted", that he/she has not assigned such rights to third parties, that the Contribution has not heretofore been published in whole or in part, that the Contribution contains no libelous statements and does not infringe on any copyright, trademark, patent, statutory right or proprietary right of others, including rights obtained through licenses; and that Author will indemnify Springer against any costs, expenses or damages for which Springer may become liable as a result of any breach of this warranty.

§ 5 Delivery of the Work and Publication

Author agrees to deliver to the responsible Volume Editor (for conferences, usually one of the Program Chairs), on a date to be agreed upon, the manuscript created according to the Springer Instructions for Authors. Springer will undertake the reproduction and distribution of the Contribution at its own expense and risk. After submission of the Consent to Publish form Signed by the Corresponding Author, changes of authorship, or in the order of the authors listed, will not be accepted by Springer.

§ 6 Author's Discount

Author is entitled to purchase for his/her personal use (directly from Springer) the Work or other books published by Springer at a discount of 40% off the list price as long as there is a contractual arrangement between Author and Springer and subject to applicable book price regulation. Resale of such copies or of free copies is not permitted.

§ 7 Governing Law and Jurisdiction

This agreement shall be governed by, and shall be construed in accordance with, the laws of Switzerland. The courts of Zug, Switzerland shall have the exclusive jurisdiction.

Corresponding Author signs for and accepts responsibility for releasing this material on behalf of any and all Co-authors.

Signature of Corresponding Author: **Date:**

February 26, 2018

. . . .

I'm an employee of the US Government and transfer the rights to the extent transferable (Title 17 §105 U.S.C. applies)

I'm an employee of the Crown and copyright on the Contribution belongs to Her Majesty

I'm an employee of the EU or Euratom and copyright on the Contribution belongs to EU or Euratom

Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

AAAI COPYRIGHT FORM

Title of Article/Paper: Spectral Word Embedding with Negative Sampling

Publication in Which Article/Paper Is to Appear: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-18)

Author's Name(s): Behrouz Haji Soleimani, Stan Matwin

Please type or print your name(s) as you wish it (them) to appear in print

PART A – COPYRIGHT TRANSFER FORM

The undersigned, desiring to publish the above article/paper in a publication of the Association for the Advancement of Artificial Intelligence, (AAAI), hereby transfer their copyrights in the above article/paper to the Association for the Advancement of Artificial Intelligence (AAAI), in order to deal with future requests for reprints, translations, anthologies, reproductions, excerpts, and other publications.

This grant will include, without limitation, the entire copyright in the article/paper in all countries of the world, including all renewals, extensions, and reversions thereof, whether such rights current exist or hereafter come into effect, and also the exclusive right to create electronic versions of the article/paper, to the extent that such right is not subsumed under copyright.

The undersigned warrants that they are the sole author and owner of the copyright in the above article/paper, except for those portions shown to be in quotations; that the article/paper is original throughout; and that the undersigned right to make the grants set forth above is complete and unencumbered.

If anyone brings any claim or action alleging facts that, if true, constitute a breach of any of the foregoing warranties, the undersigned will hold harmless and indemnify AAAI, their grantees, their licensees, and their distributors against any liability, whether under judgment, decree, or compromise, and any legal fees and expenses arising out of that claim or actions, and the undersigned will cooperate fully in any defense AAAI may make to such claim or action. Moreover, the undersigned agrees to cooperate in any claim or other action seeking to protect or enforce any right the undersigned has granted to AAAI in the article/paper. If any such claim or action fails because of facts that constitute a breach of any of the foregoing warranties, the undersigned agrees to reimburse whomever brings such claim or action for expenses and attorneys' fees incurred therein.

Returned Rights

In return for these rights, AAAI hereby grants to the above author(s), and the employer(s) for whom the work was performed, royalty-free permission to:

1. Retain all proprietary rights other than copyright (such as patent rights).
2. Personal reuse of all or portions of the above article/paper in other works of their own authorship. This does not include granting third-party requests for reprinting, republishing, or other types of reuse. AAAI must handle all such third-party requests.
3. Reproduce, or have reproduced, the above article/paper for the author's personal use, or for company use provided that AAAI copyright and the source are indicated, and that the copies are not used in a way that implies AAAI endorsement of a product or service of an employer, and that the copies per se are not offered for sale. The foregoing right shall not permit the posting of the article/paper in electronic or digital form on any computer network, except by the author or the author's employer, and then only on the author's or the employer's own web page or ftp site. Such web page or ftp site, in addition to the aforementioned requirements of this Paragraph, shall not post other AAAI copyrighted materials not of the author's or the employer's creation (including tables of contents with links to other papers) without AAAI's written permission.
4. Make limited distribution of all or portions of the above article/paper prior to publication.
5. In the case of work performed under a U.S. Government contract or grant, AAAI recognized that the U.S. Government has royalty-free permission to reproduce all or portions of the above Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract or grant so requires.

In the event the above article/paper is not accepted and published by AAAI, or is withdrawn by the author(s) before acceptance by AAAI, this agreement becomes null and void.

(1)

	2017-November-21
Author/Authorized Agent for Joint Author's Signature	Date
Employer for whom work was performed	Title (if not author)

(For jointly authored Works, all joint authors should sign unless one of the authors has been duly authorized to act as agent for the others.)

Association for the Advancement of Artificial Intelligence

2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

PART B – U.S. GOVERNMENT EMPLOYEE CERTIFICATION

This will certify that all authors of the above article/paper are employees of the U.S. Government and performed this work as part of their employment, and that the article/paper is therefore not subject to U.S. copyright protection. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations.

(2)

U.S. Government Employee Authorized Signature	Date
Name of Government Organization	Title (if not author)

(Please read and sign and return Part B ***only*** *if you are a government employee and created your article/paper as part of your employment. If your work was performed under Government contract, but you are not a Government employee, sign only at signature line (1) in Part A and see item 5 under returned rights. Authors who are U.S. government employees should also sign signature line (1) in Part A above to enable AAAI to claim and protect its copyright in international jurisdictions.)*

PART C–CROWN COPYRIGHT CERTIFICATION

This will certify that all authors of the above article/paper are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties , and that the article/paper is therefore subject to Crown Copyright and is not assigned to AAAI as set forth in the first sentence of the Copyright Transfer Section in Part A. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations, and acknowledges that AAAI has the right to publish, distribute, and reprint the Work in all forms and all media.

(3)

British or British Commonwealth Government Employee Authorized Signature	Date
Name of Government Organization	Title (if not author)

(Please read and sign and return Part C ***only*** *if you are a British or British Commonwealth Government employee and the Work is subject to Crown Copyright. Authors who are British or British Commonwealth government employees should also sign signature line (1) in Part A above to indicate their acceptance of all terms other than the copyright transfer.)*

Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

AAAI DISTRIBUTION LICENSE

Title of Article/Paper (Work):	Spectral Word Embedding with Negative Sampling
Publication in Which Article Is to Appear:	Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-18)
Author's Name(s):	Behrouz Haji Soleimani, Stan Matwin

Please type or print your name(s) as you wish it to appear in print

	2017-November-21
Author's Signature	Date
Employer for whom work was performed	Title (if not author)

Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

AAAI COPYRIGHT FORM

Title of Article/Paper: Fast PMI-Based Word Embedding with Efficient Use of Unobserved Patterns

Publication in Which Article/Paper Is to Appear: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-19)

Author's Name(s): Behrouz Haji Soleimani, Stan Matwin

Please type or print your name(s) as you wish it (them) to appear in print

(1)

	2018-November-13
Author/Authorized Agent for Joint Author's Signature	Date
Employer for whom work was performed	Title (if not author)

(For jointly authored Works, all joint authors should sign unless one of the authors has been duly authorized to act as agent for the others.)

Association for the Advancement of Artificial Intelligence

2275 East Bayshore Road, Suite 160

Palo Alto, California 94303 USA

PART B – U.S. GOVERNMENT EMPLOYEE CERTIFICATION

This will certify that all authors of the above article/paper are employees of the U.S. Government and performed this work as part of their employment, and that the article/paper is therefore not subject to U.S. copyright protection. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations.

(2)

U.S. Government Employee Authorized Signature	Date
Name of Government Organization	Title (if not author)

(Please read and sign and return Part B ***only*** *if you are a government employee and created your article/paper as part of your employment. If your work was performed under Government contract, but you are not a Government employee, sign only at signature line (1) in Part A and see item 5 under returned rights. Authors who are U.S. government employees should also sign signature line (1) in Part A above to enable AAAI to claim and protect its copyright in international jurisdictions.)*

PART C–CROWN COPYRIGHT CERTIFICATION

This will certify that all authors of the above article/paper are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties , and that the article/paper is therefore subject to Crown Copyright and is not assigned to AAAI as set forth in the first sentence of the Copyright Transfer Section in Part A. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations, and acknowledges that AAAI has the right to publish, distribute, and reprint the Work in all forms and all media.

(3)

British or British Commonwealth Government Employee Authorized Signature	Date
Name of Government Organization	Title (if not author)

(Please read and sign and return Part C ***only*** *if you are a British or British Commonwealth Government employee and the Work is subject to Crown Copyright. Authors who are British or British Commonwealth government employees should also sign signature line (1) in Part A above to indicate their acceptance of all terms other than the copyright transfer.)*

Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

AAAI DISTRIBUTION LICENSE

Title of Article/Paper (Work): Fast PMI-Based Word Embedding with Efficient Use of Unobserved Patterns

Publication in Which Article Is to Appear: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-19)

Author's Name(s): Behrouz Haji Soleimani, Stan Matwin

Please type or print your name(s) as you wish it to appear in print

	2018-November-13
Author's Signature	Date
Employer for whom work was performed	Title (if not author)