Text Embeddings in Riemannian Manifolds

Thesis submitted in partial fulfillment
of the requirements for the degree of

*Master of Science in **Computational Linguistics** by Research*

by

Souvik Banerjee
20171094
souvik.banerjee@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA
July 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Text embeddings in Riemannian Manifolds"** by Souvik Banerjee, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Manish Shrivastava

To Ma and Baba.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr Manish Shrivastava, who has guided me throughout my years as a research student in NLP. A special thanks to the rest of the LTRC faculty members, especially, Prof Radhika Mamidi, Prof Dipti Mishra and Prof Vasudev Verma for being amazing professors and sparking my interest in Linguistics in the first two years of my college life.

I am sincerely grateful to Bamdev Mishra and Pratik Jawanpuria whose guidance has led to me publishing a first author paper. Their guidance has been absolutely necessary for me to understand the maths behind my work and gain so much insight into the niche of work I have pursued for writing this manuscript.

I could not have undertaken this journey without the help of my fellow researchers Siddarth Bhatt and Alok Debnath. This project was their brain-child which inspired my interest and research in NLP. The long discussions, meetings and mathematical discourse I have had with them over the last three years is invaluable. They have also taught me to how to systematically approach a research problem - a fundamental aspect of academic research and writing. I would also like to thank Saujas Vaduguru whose knowledge in NLP goes beyond anything that I ever hope to have. His advices and suggestions have helped me a lot in the last two years.

I am deeply indebted to my parents whose support has brought me this far in life. I am forever grateful to two of my closest hometown friends - Aritra and Sayantan for they have taught me how to live life and still teach me to this day. A big personal thanks to my college friends - Shantanu Das, Abhinav Vaishya, Zubair Abid, Priyank Modi, Ujwal Narayan, Sriven Reddy, Debojit Das, Meher Shashwat Nigam, Aditya Morolia, Shivam Bansal, Aashna Jena, Suryansh Srivastava, Sarthak Singh Rajput, Athreya Chandramouli. All of them have contributed to my well being and beautiful college life. If not for their extensive support, I would have been a sad soul wandering aimlessly in this maze of a life. For this and more, I am immensely thankful.

Abstract

Unsupervised text embedding models are ubiquitous in the field of Natural language Processing. These models embed words, sentences or documents as vectors in an Euclidean space with the principle that textual information that are semantically similar have similar representations i.e they lie close to each other in the semantic space. `Word2vec` and `GLoVe` are the most popular examples of such models. Both these models provide efficient training of word embeddings and provide evaluation methods like word similarity and word analogy tasks that proves the effectiveness of these models. However, they have their fair share of drawbacks. For instance, there is little to no explanation on why word vector summation solves word analogy. These models also suffer from meaning conflation deficiency where multiple senses of a word are represented by one single vector and thus lead to inaccurate semantic modelling. This is also an issue with document and sentence embeddings which span multiple words, phrases and topics.

Existing unsupervised models which tackle meaning conflation deficiency perform sense representation where each senses of a word are represented by an individual vector. This is done by modifying the `Word2vec` skip-gram algorithm and adding some extra constraints on the vector embedding space. Instead of adding extra constraints on the `Word2vec` algorithm, we aim to address this issue of by exploiting the geometry of the embedding space. We propose three unsupervised text embedding models that embed texts in Riemannian manifolds by integrating the linguistic principles of `Word2vec` and `GLoVe` with optimization tools from differential geometry. The first and the second model uses the same joint modelling framework but embeds both words and documents in the Grassmannian manifold and a custom product manifold respectively. Both models generate quality document embeddings given by their evaluation results in document clustering and document classification tasks. The third model embed words in Spectahedron manifold where each word is a matrix whose eigenvectors correspond with the multiple senses of that word.

Finally, we provide a Lie-group theoretic understanding of linear substructures present in `Word2vec` and `GLoVe` that solves word analogy. Lie groups are differentiable manifolds with a group structure. Some basic properties of these groups show that the linear substructures are present due to the tangent space of the Identity element of the group.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Representational learning of textual information is very pivotal in modern Natural Language Processing(NLP). As bigger and more complicated neural network(NN) models enter the field of NLP research, it becomes increasingly important to model textual information as some form of a mathematical object embedded in a latent space that can then be seamlessly integrated into complex NN models. Unsupervised word representation learning models have proven to be remarkably effective in various downstream tasks of NLP which range from named entity recognition(NER), lexical entailment, document classification, document clustering, machine translation, and more. Typically, such word representation learning models encode words into embeddings in a latent space which fall into either of the following two categories - contextualized embeddings where each instance of a word appearing in the given corpus is encoded with a different representation based on its local context and global/non-contextualized embeddings where each unique word in the corpus is represented with one representation irrespective of its context. In recent years, contextualized embedding models like `BERT` [21], `GPT-3` [15], `RoBERTa` [48] have shown state of the art results in most NLP downstream tasks and thus global embeddings have taken a bit of a backseat in recent years.

Popular architectures of such global embeddings include `Word2vec` [55] [56], `GLoVe` [63], where the words are encoded as embeddings in a vector space. Most of these models devise their training with the principle that words appearing in the same context will have a higher dot product. Such a principle offers a distributional interpretation of lexical semantics, which goes by the adage - "A word is known by the company it keeps" [26]. `Word2vec` and `GLoVe` like models are an integral part of Neural Network-based NLP research. They are conveniently available in software packages and therefore allow very efficient training and evaluation of the learned embeddings. To this date, the NLP community has not managed to find ways to pre-train word embedding models that supersede `Word2vec` and `GLoVe` methods. However, these models have an inherent lack of interpretability [31] and have their fair share of deficiencies. One workaround for that has been to move away from vectors to other paradigms of representations like density-based representations - Gaussian embeddings [75], tuples of probability representations like fuzzy set embeddings [13] and manifold-based embeddings like `Poincaré GLoVe` [72].

1.1 Some criticisms that pave the way for more research

There is a lack of theoretical explanation about the effectiveness of global embeddings in literature. Furthermore, the symmetric and linear nature of the vector space operations makes it hard to correspond similarity and analogy results with human cognition theory and behaviour [74]. They remain quite puzzling and mysterious even with the growing popularity of contextualized embeddings. A widely accepted explanation and theory is proposed by [46] which states that `Word2vec` Skip-Gram Negative Sampling(SGNS) based models are implicitly factorizing a Pointwise Mutual Information (PMI) word-context matrix. They also use the same theory of PMI to explain the linear substructure present in these embeddings - why vector summation solves analogy and cosine similarity solves word similarity. However, the PMI method is not well understood in itself [4] and thus no concrete mathematical framework exists to explain neural word embeddings with similar architecture. Moreover, there seems to be a gap between the actual implementation of `Word2vec` in its C code and the theory present in the paper. For instance, `Word2vec` trains two sets of embedding matrices in the C code. Each word is associated with two vectors - context vector, which is used when a word appears in the context window of the focus word(the word which the cursor is currently at in the corpus), and focus vector, which is used when the current training iteration is at that specific word in the corpus. However, no mention or reasoning is provided in the paper as to why that is the case. `GLoVe` also uses the same two sets of embedding matrices although they mention it in their paper. It seems to be crucial for proper training of these embeddings. [57] provides more insight into the strange geometry of these two sets of embeddings. They experimentally verify that in the case of `Word2vec` , the focus vector positions are not simply determined by semantic similarity, but rather occupy a narrow cone and are diametrically opposed to the context vectors in the latent space. They also show that this geometric concentration of the narrow cone depends on the ratio of positive to negative examples and that it is neither theoretically nor empirically inherent in related embedding algorithms like `GLoVe` . [47] show that much of the performance gains of word embeddings are due to certain system design choices and hyperparameter optimizations, rather than the embedding algorithms themselves. The single vector representation of textual information also leads to meaning conflation deficiency for polysemous words, sentences and documents. Moreover, word similarity evaluations using cosine similarity and word analogy evaluations using vector summation have no concrete theoretical justification. Clearly, there exists a linear substructure in the vector space which leads to word embeddings providing solutions for word analogy but there is no concrete mathematical framework to justify that.

We aim to address the meaning conflation deficiency of textual embeddings and provide a new perspective on word analogy through the lens of differential geometry. The next two sections describe the two issues and provide motivation for our approaches to addressing them.

1.2 Meaning conflation deficiency

`GLoVe` and `Word2vec` like embeddings map each word to a single point in the vector space. As a result of that, all possible meanings of the word present in the corpus are conflated into a single representation, which is termed as meaning conflation deficiency [1]. This problem of meaning conflation becomes even more of an issue for sentence and document vectors. Sentences generally span multiple words and phrases while documents span multiple sentences that have their own local context. Every document also has a different number of distinct topics. Practical applications in NLP generally consider the concept of "topics" to be a cluster of similar words. Such nuances cannot be captured by a single fixed-dimensional vector representation.

[68] is one of the earliest works to identify the meaning conflation deficiency of word vectors. Having different (possibly unrelated) meanings conflated into a single representation can hamper the semantic understanding of a language model that utilizes these embeddings are pre-trained vector inputs. [77] has shown that word embeddings are unable to capture the various senses of a word even if those senses appear in the corpus we are training on. Meaning conflation deficiency also causes some unwanted semantic modelling behaviour ([61], [64]) like pulling two seemingly unrelated words close together in the latent space. We take the example of two words: "rat" and "screen" to demonstrate that. They are pulled towards each other in the embedding space because both words are synonymous with the word "mouse" under two different senses. If a distinction is not made between the senses, they have negative effects on the accuracy of semantic modelling. One popular solution to this problem is to use sense representation i.e. each individual sense of a word present in a corpus is encoded with an independent vector embedding. There are two main paradigms of sense representation -

- Unsupervised models identify the different senses of a word through word sense induction, i.e., automatic identification of a word's senses by analyzing the contexts in which it appears in a text corpora
- Knowledge-based models use an external sense inventory like `WordNet`(which keeps a list of all the senses of a word) to identify the senses after single word vectors are obtained from the usual training

We are more interested in unsupervised models since no extra data is required other than the text corpora. There are quite a few well-known models that put extra sets of constraints in the `Word2vec` Skip-gram negative sampling(SGNS) algorithm to encode different senses ([61], [10]). Backed by quantitative and qualitative analysis, such models have been shown to produce high-quality sense vector representations.

We argue that the conflation deficiency issue in textual embeddings is better tackled by "exploiting" the structure of the embedding space instead of putting extra constraints on the SGNS algorithm like [61] and [10]. Since the linear nature of vector space does not provide much room for such exploitation, we shift from learning embeddings on vector spaces to Riemannian Manifolds - a family of continuous spaces which are non-linear and also allow relatively simple usage of differential calculus to carry out

optimization [2]. Learning textual representations on Riemannian manifolds allows more robust modes of representation because some innate property of the space might naturally model a linguistic property of the word. For instance, it has been observed that hierarchical data is far better represented in the hyperbolic space because of the innate continuous tree-like structure of the space [62]. So, hyperbolic word representation models hypernymy-hyponymy relations better. To add to that, even sentences and paragraphs might also benefit from these structures. Therefore, we propose unsupervised text embedding models that learn textual representations in Riemannian manifolds and train them using optimization tools in differential geometry.

1.3 Theoretical Understanding of Word Analogy

A word analogy is a statement of the form "a is to b as x is to y", which asserts that a and x can be transformed in the same way to get b and y, and vice-versa. A somewhat surprising property of `Word2vec` and `GLoVe` embeddings is that they exhibit some form of linear structures such that word analogies can be solved by vector arithmetic [55] [56]. For instance, in Figure 1.1, we see the vector connecting woman to man and the vector connecting queen to king are almost parallel. Indeed, the vector for queen can be retrieved by performing $w_{\text{king}} - w_{\text{man}} + w_{\text{woman}}$. So, if there exists a word analogy $a : b :: c : d$, the vector space embedding of these words adheres to the equation $w_y = w_b - w_a + w_c$. In practice though, it is not true for all analogies. Nevertheless, it is still quite relevant given the 75% accuracy scores on word analogy task by `GLoVe` embeddings ([63] has listed the accuracy score for `Word2vec` and other unsupervised vector space embeddings as well). Figure 1.1 shows the word analogy pattern in `GLoVe` embeddings.

As mentioned previously, the reason for the existence of such additive structures is not fully understood. Since these word embeddings inherently lack interpretability [45], there is a need for a more theoretical understanding of these models. However, comparatively little work exists that provides an elaborate mathematical framework that can explain why vector addition captures analogy relations. There are a handful of works that have tackled the issue based on the theory of PMI-matrix factorization [46]. As explained by their similar algorithm structure, the rationale underpinning analogical structure in both `GLoVe` and `Word2vec` like models should also be theoretically similar. [4] proposes a generative model which treats corpus generation as a random walk of a discourse vector and provides theoretical motivations for word analogy behavior using that. [29] and [3] use a "paraphrasing" concept to build on top of the PMI-matrix to explain word analogy. As explained in [23], both these papers make strong assumptions about the distributions of word frequencies and also do not have any empirical evidence to back up their claims. [23] instead attempts to provide an information-theoretic explanation of Euclidean distance in SGNS and `GLoVe` and build on top of that along with PMI-matrix to explain word analogy.

We provide a Lie-Group theoretic understanding of these "linear substructures providing solutions to word analogy" phenomena that exist in `Word2vec` and `GLoVe` like architecture embedding models without making any strong assumptions about the distribution of word frequencies in the corpus. In

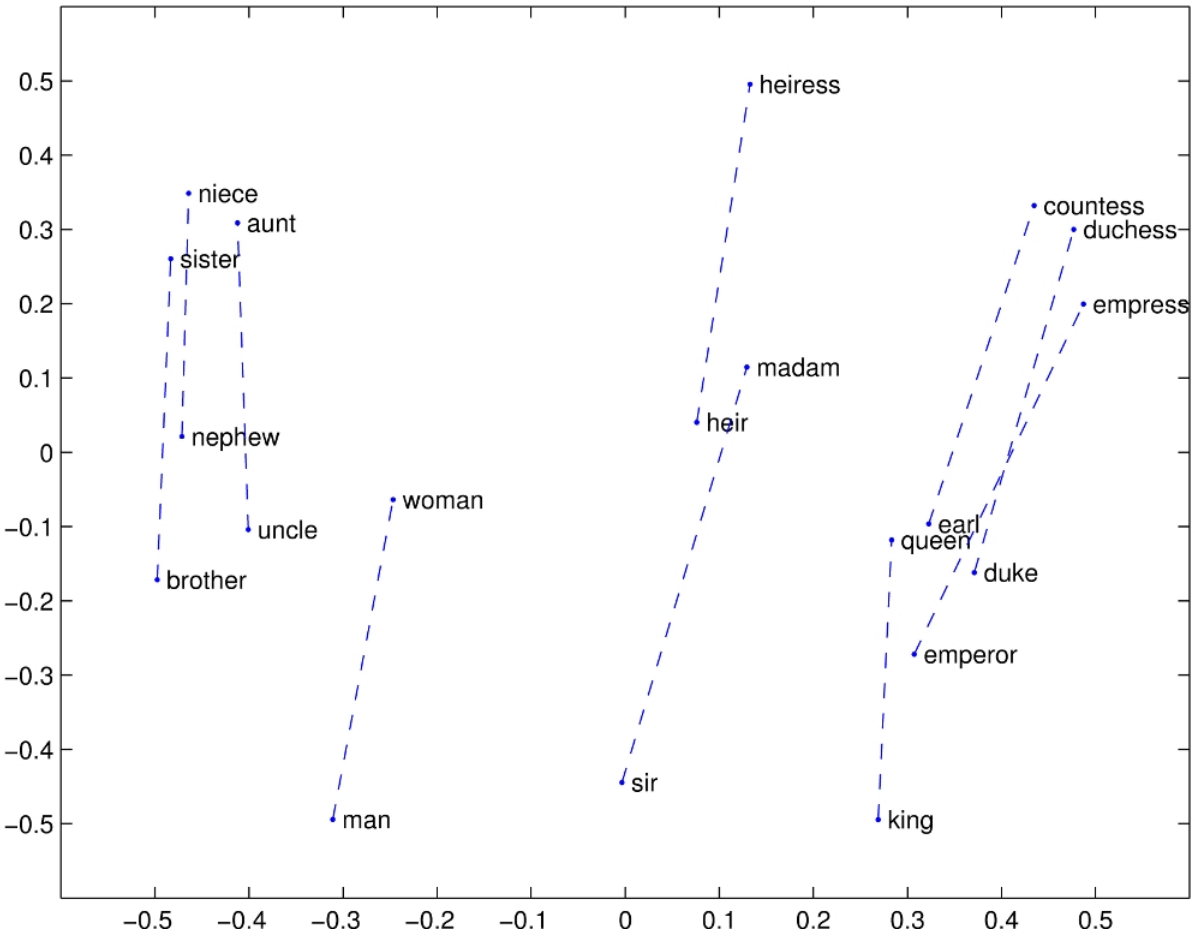

Figure 1.1: Linear substructure of `GLoVe` trained word embeddings that solve word analogy. The points are plotted after PCA from 300 dimensions to 2 dimensions. We can see a parallelogram-like pattern between "woman, man, queen, king".

differential geometry, lie groups are differentiable manifolds that have a group structure. We show that the `Word2vec` SGNS can be modified and extended such that words are embedded in any arbitrary Lie Group, preferably a matrix group. This is done by exploiting a few basic properties of Lie theory, mapping each word uniquely to a point in the tangent space of the identity element of the group.

The contributions of this thesis are as follows:

- Applying optimization techniques in Spectahedron and Grassmannian manifold to build an unsupervised textual embedding model that tackles the issue of meaning conflation deficiency

- Introduce a novel similarity metric to represent text embeddings with matrices such that each column captures a latent topic/variable associated with a word or a document

- Exploiting the intrinsic geometric properties of the above embedding space to perform better in qualitative tasks like document clustering, document classification, word, and sentence similarity
- Provide a Lie-Group theoretic understanding of linear word analogy

1.4 Thesis Organization

Chapter 2 We provide the relevant linguistic background on distributional semantics. We cite the landmark papers in vector space global textual embeddings and delve deeper into the two most important embedding models relevant to our study - `Word2vec` and `GLoVe` . We also look at notable works in representation learning in non-euclidean spaces that serve as a jumping point for our research. We modify the basic framework of these models, suit it for the manifold we want to train our embeddings in, and use some innate property of the space to tackle the meaning conflation deficiency and analogy interpretability problem.

Chapter 3 We provide a primer of the mathematical concepts that one would require to understand the framework behind the proposed models in this manuscript. These mathematical concepts form the basis of a highly involved branch of topology - differential geometry. A thorough reading of these concepts with the supplementary study materials (the papers are cited in the chapter) should give the reader a full mathematical grasp of the embedding models along with their training and analysis.

Chapter 4 and 5 provides two different forms of text representation using matrices. They both have the same model framework - the generative model idea proposed in `JoSE` [52], which jointly learns words and paragraph embeddings as points in the spherical manifold. JoSE itself is based on `Word2vec` SGNS algorithm and ultimately modifies it.

Chapter 4 We generalize the generative model idea proposed in `JoSE` to learn words and paragraph embeddings as some fixed dimensional subspace. In differential geometry, a manifold space that would have fixed dimensional subspaces as points is called a Grassmannian manifold. We show that the choice of metric allows us to be robust and model words and paragraph subspaces to have different dimensions. Qualitative results on various evaluation tasks are provided to show the benefits.

Chapter 5 The framework of the model proposed in this chapter is similar to the model used in Chapter 4. This time around, instead of a subspace representation, we provide a custom matrix representation with a novel metric that exploits the matrix structure of the embeddings. This robust metric takes a word or document matrices of an arbitrary number of columns and calculates the similarity between them. The resulting document embeddings perform better than our baseline model.

Chapter 6 We propose an unsupervised model to learn words as points in the Spectahedron manifold. The Spectahedron Manifold is mathematically represented by a symmetric positive semi-definite matrix and allows the choice of two metrics of similarity. The basic framework is that of a skip-gram negative sampling model with a custom loss function. The mode of representation used in the Spectahedron manifold is a matrix that allows for a unique geometric interpretation where the most dominant senses of the word can be obtained by the eigenvalue decomposition of the word matrix in question.

Chapter 7 We introduce a few key concepts from Lie Group theory that allows the extension of the Skip-Gram negative sampling algorithm to train words in the manifold of symmetric positive definite manifold(SPD) with the Log-Euclidean metric (a Lie group). We then proceed to generalize the algorithm for arbitrary matrix Lie groups.

Chapter 2

Background

2.1 Vector Space Embedding Models

The simplest and the most ubiquitous mode of textual representation is the vector space representation where they are represented as dense fixed-length (also called the dimension) vectors. The intrinsic linear geometry of vector space allows the usage of extremely computationally efficient functions like vector addition, subtraction, cosine similarity metric, etc, and thus allows these representations to be easily integrated into more complicated machine learning and deep learning algorithms as encoded textual information. The novelty of the vector space models (VSM) is generally attributed to [67] which considered a document as a vector whose dimensions were the whole vocabulary. The theory was then refined by the Information Retrieval (IR) community and was backed by research in human cognition [44], [28]. The aforementioned document-based VSM has now been extended to other lexical items like words. In this case, words are generally represented as points embedded in a vector space. From the standpoint of semantics, word vector embeddings are especially important as they encode basic semantic information like the meaning of a word which in turn contributes to understanding the semantics of every other textual information like phrases, sentences, documents, and ultimately the whole corpus.

Keeping the above points in mind, we give a brief overview of three popular vector space embedding models relevant to our novel study.

2.1.1 `Word2vec`

`Word2vec` - an unsupervised word embedding model [55] [56], is arguably the gold standard of non-contextualized vector space models. `Word2vec` embeds words as points in a vector space such that words that have similar contexts have higher dot products between them. This principle is based on the distributional view of lexical semantics [26]. While definitely not the first to propose such an idea [9], [16], `Word2vec` provides an efficient training regime to train the embeddings, easily store and evaluate them for other downstream tasks. The vanilla `Word2vec` model has two main architectures - continuous Bag of Words (CBOW) and continuous Skip-Gram model.

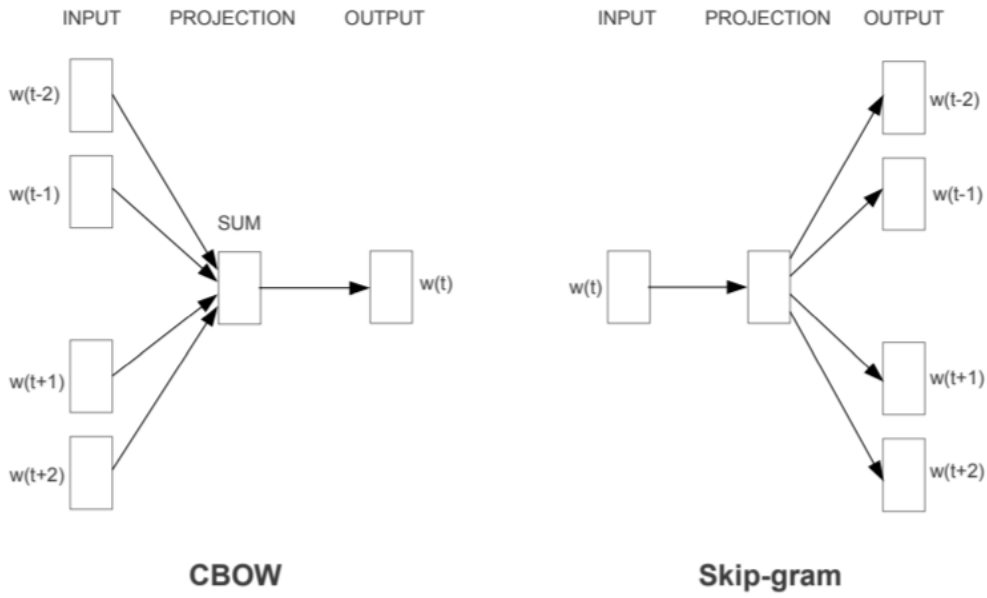

Figure 2.1: The two main architectures of `Word2vec` . CBOW takes as inputs the sequence of words left and right of the focus word within a local context window and outputs out the focus word while skip-gram does the opposite

CBOW. In Figure 2.1, the illustration on the left shows the CBOW input - a set of words $[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$. This set consists of the context words surrounding the focus(center) word w_t based on a "context window". The context window size in the illustration above is 2. Thus, CBOW is seen as a precognitive model which takes in a certain number of words around the focus word to predict the focus word itself. The objective loss function looks like this -

$$J = \frac{1}{T}\sum_{t=1}^{T} \log p(w_{t-c}, \ldots, w_t, w_{t+1}, \ldots, w_{t+c}|w_t) \tag{2.1}$$

where c is the local context window and T is the number of words in corpus and t is the time step

Skip-Gram. Skip-gram, on the other hand, turns the CBOW objective on its head i.e. it uses the focus word to predict its surrounding context words. The illustration on the right in Figure 2.1 shows w_t predicting the context words which was the set of words used as input in CBOW. The skip-gram objective is thus shown to be the sum of log probabilities of the surrounding words to the left and right of the focus words.

$$J = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c \le j \le c, \neq 0} \log p(w_{t+j}|w_t) \tag{2.2}$$

where the notations are exactly the same as the ones in CBOW with j iterating over all the surrounding words in the context window. In the above loss function, the conditional probability $p(w_{t+j}|w_t)$ is calculated using softmax over the dot products of all the words in the vocabulary with center word w_t

$$p(w_{t+j}|w_t) = \frac{\exp(v'^{\top}_{w_{t+j}} v_{w_t})}{\sum_{i=1}^{V} \exp(v'^{\top}_{w_i} v_{w_t})} \tag{2.3}$$

where v_x represents the vector embedding for word x.

While it is feasible to calculate the softmax for each word at every time step, it is computationally expensive when dealing with large vocabularies. Thus, it becomes necessary to take an approximation of the denominator in Equation 2.3. `Word2vec` proposes two methods - hierarchical softmax and negative sampling, to do the same. From this point on, we use the skip-gram negative sampling algorithm as the default algorithm for `Word2vec` . Unless otherwise stated, `Word2vec` algorithm means the skip-gram negative sampling algorithm in this manuscript. After a few simplifications and notation changes, the skip-gram loss function takes the form

$$J = \sum_{(w_c, w_f) \in D} \log \sigma(v'^{\top}_{w_c} v_{w_f}) + \sum_{(w_n, w_f) \in D'} \log \sigma(-v'^{\top}_{w_n} v_{w_f}) \tag{2.4}$$

Here, set D and D' are two sets of tuples such that D contains a list of word pairs occurring in the context window of each other and D' contains a list of negatively sampled word pairs where the two words do not appear in the context window of each other. [45] takes a deeper dive into the inner workings of `Word2vec` and discusses these concepts in detail. We highly recommend a thorough reading of the mentioned manuscript.

2.1.2 `GLoVe`

`GLoVe` short for Global Vector embeddings [63], is another really popular and influential word embedding model. `GLoVe` claims that an optimal vector space embedding model should make efficient use of statistics incorporating global word-word co-occurrence counts as they do in matrix factorization methods like LSA [20] and also incorporates local context windows like `Word2vec` . They propose a log bi-linear regression model that produces embeddings in a vector space with meaningful substructures as is evident by their word analogy evaluation results. Further qualitative results are provided on word similarity and named entity recognition where they outperform their contemporary state-of-the-art models. Specifically, the loss function is a weighted least squares model that trains on global word-word co-occurrence counts like this -

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \bar{w}_j + b_i + \bar{b}_j - \log X_{ij})^2 \tag{2.5}$$

where V is the total number of words in the vocabulary, $f(X)$ is a weighing function with a few special properties(for more details, please refer to the paper), X_{ij} is a ratio of word-word co-occurrence counts

between i-th and j-th word. $b_i, \bar{b}_j$ are bias vectors of focus word vector w_i and context word vector $\bar{w}_j$ respectively.

The paper further argues that all unsupervised word embedding models exhibit same kind of properties and are ultimately based on the co-occurrence statistics of a corpus. It is only logical that all these model will have commonalities or more strongly, a basic theoretical framework. While it is hard to arrive at mathematical connections between all possible models, they go onto show that the skip-gram softmax loss function can be reduced to take the form of Equation 2.5. Though we are skipping the details, readers are highly advised to check out section 3.1 in [63] for further clarity.

Relationship between Word2Vec and GLoVe. [46] states that the `Word2vec` skip-gram negative sampling algorithm is implicitly factorizing a PMI-matrix of the word-context pairs shifted by a factor determined the number of negative samples. Mathematically, `Word2vec` - SGNS is optimised if,

$$w_i^\top c_j = \mathbf{PMI}(w_i, c_j) - \log k, \tag{2.6}$$

where $\mathbf{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i)p(c_j)}$ which is the log ratio between w_i and c_j's joint probability and product of their marginal probabilities. k is the number of negative samples, w_i and c_j are the i-th focus word vector and j-th context word vector in the focus and context matrix respectively. A keen observer will note that Equation 2.6 has a similar form to the `GLoVe` loss function in Equation 2.5. Indeed, when loss is minimised to 0 for `GLoVe` embeddings, w_i, c_j should satisfy the property

$$w_i^\top c_j = \log X_{ij} - b_i - \bar{b_j} \tag{2.7}$$

Since both `Word2vec` SGNS and `GLoVe` have the same architecture, the theoretical framework that underlies `Word2vec` will also apply, albeit a little loosely (due to the inclusion of bias vectors) to `GLoVe` as well. The same goes for their cons as well.

2.1.3 `Doc2vec`

`Doc2vec` [45] is an extension of `Word2vec` for encoding documents as points in a vector space. Figure 2.2 illustrates the two model architectures used in `Doc2vec` - Distributed Memory Model Of Paragraph Vectors (DM) and Paragraph Vector With A Distributed Bag Of Words(DBOW). It is quite apparent that Distributed memory model closely resembles the CBOW architecture of `Word2vec` where alongside the surrounding context words, the paragraph id(which is the vector encoding of the paragraph) of which the words are a part, is passed as an input to predict the focus word. On the other hand, Distributed memory resembles the skip-gram model with the paragraph id being the input that predicts the focus word and its surrounding context words based on a local window. A combination of both algorithms is recommended for usage, though the PV-DM model is superior and usually produce superior results by itself.

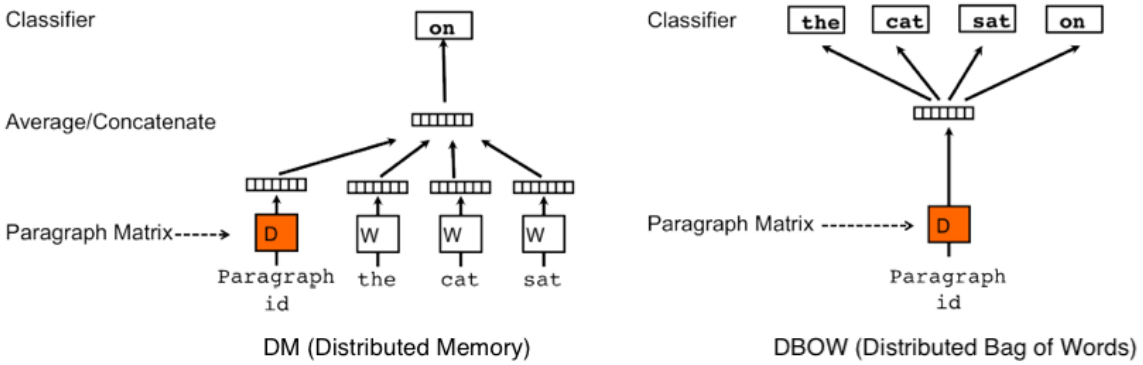

Figure 2.2: The two main architectures of Doc2Vec. DM takes as inputs the sequence of words left and right of the focus word within a local context window and outputs out the focus word while DBOW does the opposite

2.2 Non-euclidean Embedding Models

This study is certainly not the first to propose unsupervised text embedding models that train in non-Euclidean spaces. We look at a few notable works which move away from the usual vector-based representation of text and try different modes of representation.

2.2.1 Joint Spherical Text Embedding(`JoSE`)

[52] proposes a generative model which learns word and document embeddings jointly in a spherical manifold, called `JoSE` . They argue that words are not only identified by their context but also by the paragraph they are part of.

$$p(v, u|\, d) \propto p(v|\, u).p(u|\, d) \tag{2.8}$$

where $p(.)$ calculates conditional probability , u, v are unit length word vectors, d is the document vector. It is observed that `Word2vec` and `GLoVe` normalizes their embeddings before performing any form of evaluation on them. So, there is a gap between training and evaluations of these embeddings. `JoSE` addresses this issue by training on the unit-spherical manifold using Riemannian optimization tools. The metric of similarity used is cosine similarity metric (which is used to calculate the geodesic distance in the spherical manifold) and uses a max-margin loss function for optimization

$$L = \max(0, m - \cos(v, u) - \cos(u, d) + \cos(v, u') + \cos(u', d)) \tag{2.9}$$

where $m > 0$ is the margin, $\cos(.)$ is the cosine similarity metric function and u' is a negatively sampled word vector with respect to focus word u. The choice of similarity metric shows that it is indeed the directional aspect of the vectors which decides the word similarity results. Both the word and document

embeddings are also shown to be of higher quality than contemporary unsupervised embedding models as given by their results in document classification, document clustering, and word similarity. The models we propose in the next two chapters build on top of this joint embedding model by extending the framework to Grassmannian manifolds(chapter 4) and a custom product manifold(chapter 5).

2.2.2 Word2Gauss

[75] represents words as Gaussian embeddings on the probability space of Gaussian distributions, instead of the usual point-vector representations. They also propose the use of an asymmetric measure of similarity like KL-divergence along with a symmetric measure like the expected likelihood kernel to train the embeddings. This allows better modeling of word similarity since linguistically speaking word pairs can have an unequal relationship like "mammals" and "humans". The density-based nature of the distribution also models lexical entailment which was missing in the point representations. However, there are certain bold assumptions made in the model. For instance, it is assumed that the covariance matrix associated with a word is either diagonal or spherical. The range of values of the variance is also limited by a few hyperparameters. Such restriction aid in faster training of the model. This Gaussian density-based representation is also mathematically linked with Hyperbolic embedding - `Poincaré GLoVe`.

2.2.3 Poincaré GLoVe

[62] was the first to propose neural network models that embed symbolic data like text, graphs, and multi-relational data on a Hyperbolic manifold - Poincaré ball. The model observes that the trained embeddings capture both hierarchical and synonymous relations in the data much better than vector space embeddings. Inspired by that, [72] takes the training regime of `GLoVe` and modifies the architecture to train words as points in the Hyperbolic spaces - Poincaré Disk and Poincaré ball. They provide a new metric of similarity well-defined in the space, closed-form gradient expression, and retraction operation. Naturally, hierarchical relations like hypernymy-hyponymy relations of words are captured better by these embeddings owing to the nature of the space(continuous tree-like structure) itself. They also extend the word analogy solution to Poincaré embeddings through the theory of Gyrocalculus which provides the Hyperbolic manifold equivalent operations of vector addition, subtraction, and scaling. Moreover, they also provide a mathematical theory to link Poincare embeddings with 1D-Gaussian word embeddings produced by `Word2Gauss`.

2.2.4 Embeddings as Elliptical probability distribution

[60] embeds objects as elliptical probability distributions - arbitrary families of elliptical distributions, which subsume Gaussian distributions. The 2-Wasserstein Bures distance is used to calculate the distance between these embeddings and is a direct generalization of `Word2Gauss` representations as it can handle

the degenerate cases by having closed-form gradient expressions for them. So, instead of training on only diagonal or spherical covariance matrices like `Word2Gauss`, they can train on the entire set of covariance matrices allowing for a more robust representation. It is also noticed that the BW distance is a geodesic distance on the space of covariance matrices, which is the manifold of positive semi-definite matrices. The modeling proposed in chapter 6 exploits the geometry of the aforementioned manifold and provides a single vector representation for each sense of a word.

Chapter 3

Learning Representations on Matrix Manifolds

[24] proposed the "Manifold hypothesis" which states that high dimensional data tend to lie in the vicinity of a low dimensional manifold. Inspired by this hypothesis, we conjecture that a more accurate modeling of word embeddings would be to train them in a low dimensional manifold space which itself is embedded in a vector space. Such a manifold is called a Riemannian submanifold. We now provide a summary of differential geometry concepts that are integral for getting a solid grasp on the manifold text embedding models proposes in the next few chapters. In order to design an unsupervised training regime for the issue we have decided to tackle, it is very crucial that we propose a loss function that takes in as argument some form of a "similarity" metric that is geometrically analogous to what cosine similarity metric does in vector space embeddings. We also recommend [2] as it provides quite a comprehensive understanding of the tools required to perform optimization on Riemannian submanifolds. Here are the primers for that -

3.1 Differentiable Manifolds

A p-dimensional manifold $\mathcal{M}$ can be informally defined as a set covered with an appropriate collection of coordinate subsets (which are referred to as charts), that identify certain subsets of $\mathcal{M}$ with open subsets of $\mathbb{R}^d$. For example, the Earth is considered to be a spherical manifold. We, the inhabitants(the "points" on the surface of Earth) locally perceive the land around us as flat(the "flat" land can be roughly considered to be the "coordinate chart" for that area of the Earth). These coordinate charts are an essential and necessary tool for addressing fundamental notions such as the differentiability of a function on the manifold. As a result of that, smooth line search algorithms like gradient descent algorithms relevant to a machine learning problem can be defined on manifold $\mathcal{M}$ just like it is defined in the Euclidean space $\mathbb{R}^d$. Indeed, the Euclidean space $\mathbb{R}^d$ happens to be a manifold itself, where the local coordinate charts come in a natural way (by identity maps). However, it is often cumbersome or impractical to use coordinate subsets to locally turn computational problems on $\mathcal{M}$ into computational problems on $\mathbb{R}^d$. This brings us to the notion of a matrix manifold.

3.2 Matrices - convenient representations of certain abstract manifolds

The training models proposed in the subsequent chapters rely on exploiting the natural matrix structure of the manifolds associated with the examples of interest, rather than imposing a local $\mathbb{R}^d$ structure. Such a matrix structure can be naturally derived from the definition in most of these cases. For example, Grassmannian Manifold Grass(d, r) is the Riemannian manifold space of all r-dimensional subspaces of $\mathbb{R}^d$. Each point in the manifold can be represented by a full-rank orthonormal matrix U of dimension $\mathbb{R}^{d\times r}$. A detailed discussion of that can be found in chapter 4. Moreover, the manifolds we train on are all either Riemannian quotient manifolds or Riemannian submanifolds - both of which are subsets of a base manifold. In our cases, all the base manifolds are $\mathbb{R}^d$.

3.2.1 Tangent Space and Riemannian Metric

The tangent space of a differentiable manifold $\mathcal{M}$ is a vector space at a point k on the manifold whose elements are the tangent vectors (or "velocities") to all the curves passing through that point k. The tangent space at this point k is usually denoted by $T_k\mathcal{M}$. The tangent space at a point has the same dimension as the manifold itself. For a simple example, the tangent space of any point in the 3-d sphere is the the vector space of velocity vectors tangential to that point in any given direction. The dimension of those velocity vectors are the same as the dimension of the sphere - 2.

Now that the tangent space has been provided with a notion of velocity or a "sense of direction" for each point in a manifold, we also need a notion of "length" to calculate the magnitude of these velocities so that the information about steepest ascent(gradient) can be calculated. A Riemannian metric bestows a manifold with exactly that - norm. A Riemannian metric is an inner product $g : T_k\mathcal{M} \times T_k\mathcal{M} \to \mathbb{R}$. This function has a bi-linear, symmetric positive-definite form and thus it can be seen that the inner product induces a norm $||V_k|| = \sqrt{g(V_k, V_k)}$ for some element V_k in the tangent space of $T_k\mathcal{M}$. It can also be inferred easily that the gradient of a function defined on the manifold $\mathcal{M}$ at the point k is nothing but an element in the tangent space of $T_k\mathcal{M}$ with the steepest ascent special property.

When a differentiable manifold $\mathcal{M}$ is endowed with a Riemannian metric g, we call it a Riemannian manifold $(\mathcal{M}, g)$. We only deal with Riemannian manifolds in our proposed models.

3.2.2 Gradient

For Riemannian submanifold and Riemannian Quotient manifold, the gradient gradF_k at a point k in the manifold is calculated by obtaining the gradient on the base manifold which is the Euclidean gradient in our case. It is then "projected" back to the quotient manifold or submanifold in question. The projection operation is decided by the properties of the tangent and normal space.

3.2.3 Geodesic Distance Induced from a Riemannian Metric

Once the Riemannian metric is defined on a manifold, it is possible to define the structure of a metric space on $\mathcal{M}$. Specifically, curves on $\mathcal{M}$ which locally yield the shortest distance between two points are of great interest. Given two points on the manifold, a geodesic refers to the shortest curve on the manifold connecting the points. The length of the geodesic curve is called the geodesic distance induced from the Riemannian metric norm.

3.3 Metric of similarity

Similar to the `Word2vec` training regime, the goal is to minimize a loss function which attempts to align word embeddings of words occurring in similar contexts in the same spatial locality, as per the distribution hypothesis. As mentioned earlier, these loss functions require a similarity function as input for their calculation. Finding a mathematically sound similarity metric which is functionally analogous to what cosine similarity metric does in the case of vector spaces i.e give a notion of angularity between two points in a manifold forms the most important concept when approaching the problem from this differential geometric perspective. Note that the metric or "notion" of similarity between two word embeddings in a manifold space may or may not be same as the Riemannian metric of that manifold. For instance, in the case of vector spaces, the notion of similarity happens to be the dot product which is also the Riemannian metric for Euclidean spaces. As it will be evident later, the similarity metric proposed for any of the manifolds in the subsequent chapters does not coincide with their corresponding Riemannian metric.

3.4 Retraction

We need to perform the gradient descent update operation now by finding the tangent matrix corresponding to the descent direction and computing a step forward on the manifold aligned in this direction. Recall that euclidean gradient descent has an addition operation in the update rule but such linear addition is not applicable in the case of general Riemannian manifolds. Thus, we need a "retraction" operation which will carry out something analogous to the aforementioned step forward but in a Riemannian manifold. Loosely speaking, a retraction operation is just a mapping from a point in the tangent space of the manifold to a point in the manifold itself. Generally, closed form equations of the retraction for our optimization problem is derived from its equation of geodesic.

3.5 Optimization on Riemannian submanifolds

Classical `Word2vec` and `GLoVe` uses gradient descent algorithm to minimise their loss function. Most of their derivative unsupervised word embedding models also use gradient descent algorithm.

Since the principle and basic framework of our model is borrowed from `Word2vec` and its derivative work Spherical Text Embedding - `JoSE` , we use a Riemannian manifold analogue of the gradient descent algorithm. The following paragraph defines an arbitrary function over a Riemannian manifold, defines its tangent space, and lays down the step to calculate the gradient at each step and minimise the loss function over a certain number of iterations.

We define a function F on a Riemannian manifold $\mathcal{M}$ which is embedded in $\mathbb{R}^d$ such that $F : \mathcal{M} \to \mathbb{R}$. The tangent space at any point k in the manifold is denoted by $T_k\mathcal{M}$. For every k in $\mathcal{M}$, the Riemannian metric g provides an inner product on $T_k\mathcal{M}$ given by the non-degenerate symmetric bi-linear form $g_k : T_k \times T_k \to \mathbb{R}$. The notation $\langle X_k, Y_k \rangle = g_k(X_k, Y_k)$ and $||X_k|| = \sqrt{g_k(X_k, X_k)}$, where $X_k, Y_k \in T_k\mathcal{M}$ is often used. The geodesic distance induced by the Riemannian metric between two points k_1 and k_2 in $\mathcal{M}$ is denoted by $d(k_1, k_2)$. The gradient of F on $\mathcal{M}$ at k, denoted by gradF_k, is the unique vector in $T_k\mathcal{M}$ such that $df_k(X) = \langle \text{grad}_k F, X \rangle$ for all X in $T_k\mathcal{M}$. Finally, the learning rate is η and the retraction operation at point k is $\text{Retr}_k(.)$. The optimization problem would be to compute

$$\min_{k \in \mathcal{M}} F(k)$$

Select initial point $k_0 \in \mathcal{M}$

1. At any give step $t (t \geq 0)$, compute euclidean gradient G_t = - ∇F_{k_t}.
2. Project the euclidean gradient, G_t into the tangent space of k_t to get Riemannian gradient, $\text{grad}_{k_t} F$ using a projection operator appropriate for that manifold
3. A motion along the tangent vector, to add the direction of steepest descent, $(k_t + \eta \, \text{grad}_{k_t} F)$ in the linear embedding space
4. Perform retraction operation to get new point $k_{t+1} = \text{Retr}_{k_t}(k_t + \eta \, \text{grad}_{k_t} F)$

3.6 Connection to our models

All three models proposed in the manuscript use the max-margin loss function as F. The loss function has the form $\max(0, m - \text{sc}_+ + \text{sc}_-)$ where $m > 0$ and sc_+, sc_- are positive and negative word tuple sample metric of similarity. Each word or document to be embedded can be considered a point in the manifold. Thus, the loss function is a function of these similarity functions which can take in as an argument a focus word, a context word, a negatively sampled word, and a document(for chapters 4 and 5). We take the partial Euclidean gradient of the loss function w.r.t to these points separately and perform the appropriate projection to get the Riemannian gradient. Finally, the retraction operation is performed to update these points(embeddings).

Chapter 4

Grassmannian Manifold Text Embeddings

4.1 Introduction

Most unsupervised text embedding models are trained by encoding the words or paragraphs acquired from the training data in a feature-length vector, with the assumption that they reside in a Euclidean space. Such models are ubiquitous for good reason. Aside from their efficiency, they have also proven to be very effective providing us with state-of-the-art results in various intrinsic and extrinsic embedding evaluation tasks. `Word2vec` [55], [56], and `GLoVe` [63] are two notable examples where word embeddings are learned in the Euclidean space and are trained to be oriented such that vectors with higher similarities have higher dot products when normalized. Some of the most common methods of intrinsic evaluation of word embeddings include similarity, analogy, and compositionality. `Doc2vec` [45], a document embedding model generalizes the training method introduced in `Word2vec` to documents and achieves improved results in various downstream tasks like sentiment analysis, information retrieval, and multi-class classification.

However, most of these representational models of words and paragraphs over vector spaces are inherently limited due to a single vector being attributed to each word/paragraph. For instance, a paragraph is represented by a single vector when in essence, they have a lot of information to convey as they are built from a sequence of word vectors. Such limitations prevent those embeddings from performing better in document classification and clustering tasks.

This paper attempts to address this limitation by suggesting a more robust representation of words and paragraphs. The intuition for our idea comes from the fact that we should expect the text embedding vectors to lie within a lower-dimensional manifold. To what extent this hypothesis is true in this context is a subject of ongoing research [24]. [72] and [52] propose models to train embeddings on a hyperbolic space and n-hypersphere respectively, showing improved results in tasks like lexical entailment, word similarity, and analogy.

In this paper, we represent each word and paragraph as a l-dimensional and p-dimensional subspaces of its ambient vector space $\mathbb{R}^n$ respectively (l and p are integers less than n which is the dimension of the vector space) and the corresponding manifold formed from the set of all such x-dimensional subspace

is called a Grassmannian manifold. We approach the Grassmannian manifold from a matrix-analytic perspective and provide optimization methods to train word and paragraph embeddings modeled as subspaces. Thus our paper contributes the following -

1. We extend the two-step generative model idea proposed in `JoSE` [52] to Grassmannian manifolds which model the relationship between words and their belonging paragraphs to obtain words and paragraph embeddings directly during the training phase

2. Our model achieves better or comparable results with JoSE and other text embedding models in word similarity, document clustering, document classification, and sentiment analysis

4.2 Related Work

Combining optimization methods in the Grassmannian manifold to produce results in a wide range of ML tasks has certainly been attempted before. In this section, we look at a few of the most noteworthy works in detail.

[59] represents sentences as low-dimensional subspaces. Each sentence consists of a sequence of words which are in turn represented by a vector. The paper proposed the idea that a sentence can be represented by the underlying low-rank subspace in which the word vectors that are a part of the sentence lie in and which can be obtained through principal component analysis of this set of vectors. Semantic similarity tasks are performed to check the quality of these embeddings using a similarity metric based on principal angles.

It has been seen that geodesic flow methods are really well suited for domain adaptation. [34] and [32] use geodesic flow kernel to model domain shift on classes of the image under the assumption that the same class in two different domains may be modeled as separate points(subspaces) on a low-dimensional Grassmannian manifold. While the former uses a sampling-based method to generate the correct subspace, the latter does it by integrating an infinite number of subspaces that characterize changes in geometric and statistical properties from the source to the target domain. [32] also introduces a metric based on principal angles that reliably measure the adaptability between a pair of source and target domains. For a given target domain and several source domains, the metric can be used to automatically select the optimal source domain to adapt and avoid less desirable ones.

[49] introduces a new approach to capture analogies in continuous word representations, based on modeling not just individual word vectors, but rather the subspaces spanned by groups of words. They develop a modified cosine distance model based on the idea of geodesic kernels that captures relation-specific distances across word categories. The evaluation results show they perform significantly better than previous approaches in the word analogy task.

[40] provides a neural network architecture specifically built for datasets modeled as points in Grassmannian manifolds. The new network architecture is designed to accept Grassmannian data directly as input and learns new favorable Grassmannian representations that are able to provide better

results in visual recognition tasks. In other words, the new network aims to deeply learn Grassmannian features on their underlying Riemannian manifolds in an end-to-end learning architecture.

4.3 Differential Geometry Background

4.3.1 Notations

Let d and r be integers with $d \geq r$. We denote the Grassmannian manifold with $\text{Grass}(d, r)$ and define it as the set of all r-dimensional linear subspaces of $\mathbb{R}^d$. Stiefel manifold is denoted by $\text{St}(d, r)$, as the set of all $d \times r$ orthonormal matrices and non-compact Stiefel manifold is denoted by $\mathbb{R}_*^{d\times r}$, as the set of all $d \times r$ matrices of full column rank r [2]. We also have the Orthogonal group $\text{O}(d)$, a lie group which consists of all $d \times d$ orthogonal matrices, and General Linear Group $\text{GL}(d)$, another lie group consisting of all $d \times d$ invertible matrices. $\text{rk}(X)$ and $\text{tr}(X)$ refers to the rank and trace of a matrix X respectively.

4.3.2 Representation for our problem

The most prominent approaches to representing points on Grassmannian manifolds with matrices in computational algorithms are -

1. As a Quotient space of $\mathbb{R}_*^{d\times r}$. Points are represented as equivalence classes of $d \times r$ full-rank matrices where two matrices are equivalent if the subspaces spanned by the columns of their matrices are the same. Mathematically, we can write this as:

$$\text{Grass}(d, r) \simeq \mathbb{R}_*^{d\times r}/\text{GL}(r) \tag{4.1}$$

   The survey [2] provides an overview of this approach. [37] also provides a brief introduction.

2. As a Quotient space of $\text{St}(d, r)$. Points in the Grassmannian manifold are equivalence classes of $d \times r$ orthogonal matrices(as every subspace can be represented by an orthonormal matrix), where two matrices are equivalent if their columns span the same r-dimensional subspace. Equivalently, two matrices are equivalent if they are related by the right multiplication of an orthogonal $r \times r$ matrix :

$$\text{Grass}(d, r) \simeq \text{St}(d, r)/\text{O}(r) \tag{4.2}$$

   [22] provides a detailed survey on this approach.

3. From the perspective of orthogonal lie groups. The Stiefel manifold may also be defined as a quotient space but arising from the orthogonal group. $\text{St}(d, r) \simeq \text{O}(d)/\text{O}(d - r)$. So, we have:

$$\begin{aligned} &\text{Grass}(d, r) \\ &\simeq \text{St}(d, r)/(\text{O}(r) \times \text{O}(d - r) \end{aligned} \tag{4.3}$$

   This perspective was taken in [27] and [70].

4. As the set of all orthogonal projection matrices of rank p:

$$\text{Grass}(d,r) \simeq \{X \in \mathbb{R}^{d\times r} | X^T = X, X^2 = X, \text{rk}(X) = r\} \simeq \{X = YY^T | Y \in \text{St}(d,r)\} \tag{4.4}$$

[12], [37] and [38] are a few examples that use this representation and approach it from the matrix-analytic perspective.

For the rest of the paper, we use the representation given in (4.2) as the definition of $\text{Grass}(d,r)$.

4.3.3 Optimization over Grassmannian Manifold

Our choice of representation gives it a Riemannian quotient manifold structure and thus computationally optimizing a loss function over such a Riemannian manifold would require us to define an easily computable representation of the tangent space in a quotient manifold.

[22] provides detailed definitions, proofs, and theorems for Grassmannian represented as the quotient of a Stiefel manifold. The tangent space $T_X\text{Grass}(d,r)$ at $X \in \text{Grass}(d,r)$ is expressed as

$$\{Z \in \mathbb{R}^{d\times r} | Z^T X = 0\} \tag{4.5}$$

Although each element in the above set is represented using a matrix, the tangent space is also recognized as a vector space (so they are matrices with a "sense" of direction) and endowed with a Riemannian metric. As discussed before, Riemannian metric is a function g that takes two elements from the tangent space and returns a real number i.e. $g : T_X\text{Grass}(d,r) \times T_X\text{Grass}(d,r) \to \mathbb{R}$. The Riemannian metric for $\text{Grass}(d,r)$ is

$$g(\eta_x, \eta_y) = \text{tr}(\eta_X^T \eta_y) \tag{4.6}$$

where $\eta_x, \eta_y \in T_X\text{Grass}(d,r)$ and tr is the matrix trace operation.

A computationally feasible representation of the gradient plays a pivotal role in the gradient descent optimization we will use for training our model. The gradient for some function $F : \text{Grass}(d,r) \to \mathbb{R}$ can be computed by projecting the "Euclidean" gradient $F_X = \left[\frac{\partial F}{\partial X_{ij}}\right]$ onto the tangent space of the Grassmannian manifold using the orthogonal projection operator, $\nabla_X : \mathbb{R}^{d\times r} \to T_X\text{Grass}(d,r)$. Thus, the expression for the gradient of a function F at point X is

$$\nabla_X F = (I_k - XX^T)F_X \tag{4.7}$$

where I_k is an identity matrix of order k.

As it has been discussed before, we need only perform the update operation now by finding the tangent matrix corresponding to the descent direction and computing a step forward on the manifold aligned in this direction. Recall that euclidean gradient descent has an addition operation in the update rule but such linear addition is not applicable in the case of Grassmannian manifolds. Thus, we need a "retraction" operation which will carry out something analogous to the aforementioned step forward but in a Grassmannian manifold. Loosely speaking, a retraction operation is just a mapping from a point in

the tangent space of the manifold to a point in the manifold itself. We provide a closed-form equation of the retraction for our optimization problem once we define the concept of a geodesic.

Given two points on a manifold, a geodesic refers to the shortest curve on the manifold connecting the points. In the case of Grassmannian manifolds, two points $\mathbf{X}, \mathbf{Y} \in \text{Grass}(d, r)$ may be parametrized by a function $\gamma(t) : [0, 1] \rightarrow \text{Grass}(d, r)$ where the parameter t controls the location on the geodesic with the two endpoints as $\gamma(0) = \mathbf{X}$ and $\gamma(1) = \mathbf{Y}$. A computation of this geodesic operation γ is done by transporting the point $\mathbf{X}$ in Euclidean space in the direction and distance specified by the tangent vector $\Delta \in T_X\text{Grass}(d, r)$ followed by projecting the displaced point into the manifold giving the target destination $\mathbf{Y}$. This is consistent with the concept of retraction that we mentioned before. Indeed, such a "projection" mapping, also referred to as an exponential function, is a special type of retraction. There exists a well-known closed-form equation of the exponential mapping for the Grassmannian.

Let $X \in \text{Grass}(d, r)$, $\Delta \in T_X\text{Grass}(d, r)$ and $U\Theta V^T$ be the SVD decomposition of Δ i.e. $\Delta = U\Theta V^T$, we have the

$$t \rightarrow \gamma(t) = XV \cos(\Theta t) + U \sin(\Theta t) \tag{4.8}$$

The resulting subspace $\gamma(1)$ is what we require. This closed form, however, suffers from inefficiency for our problem because we largely deal with huge corpora. So, we instead use the simpler QR decomposition which is a first-order approximation of the exponential mapping. The final retraction operation looks like

$$Y = \text{qf}(X + \Delta) \tag{4.9}$$

where qf refers to the q factor acquired from the QR decomposition of the resulting matrix.

4.4 Objective Function for our model

In this section, we provide the objective function used in training our model and how we arrived at it. The intuition behind the objective function for our generative model comes from the following assumption - words and paragraphs that appear in the same context must also appear around the same locality in the space in which it is embedded. Since the cosine similarity of vectors can't be applied to subspaces directly, we need a measure of similarity that takes into account the angular distance between any two subspaces. The notion of principal angles will allow us to do just that.
Intuitively, principal angles are the "minimal" angles between all possible bases of two subspaces. Let $X \in \text{Grass}(d, r_1), Y \in \text{Grass}(d, r_2)$ where r_1, r_2 are integers less than d and $p = \min(r_1, r_2)$. The set of principal angles $\{\theta_i\}_{i=1}^{p}$ between X and Y can be defined recursively by

$$\begin{cases} \cos\theta_i = \max\limits_{\substack{x_i \in X \\ y_i \in Y}} x_i^T y_i \\ x_i^T x_i = 1, y_i^T y_i = 1 \\ x_i^T x_j = 0, y_i^T y_j = 0, \forall j < i \end{cases} \tag{4.10}$$

for all $i = 1, 2, \ldots, p$ and $\theta_i \in [0, \pi/2]$.

An efficient way of calculating the principal angles between X and Y is by the SVD decomposition of $X^T Y$ where the singular values are the cosines of the principal angles. For any matrix, the sum of squares of the singular values equals the Frobenius norm of that matrix. Thus, we have

$$\begin{aligned}\sqrt{\sum_{i=1}^{p} \cos^2 \theta_i} &= \sqrt{\text{trace}(X^T Y Y^T X)} \\ &= \sqrt{\text{trace}(X X^T Y Y^T)}\end{aligned} \tag{4.11}$$

Thus, we have a natural choice for a similarity metric that takes into account the cosine of the principal angles between X and Y.

$$\text{sim}(X, Y) = \sqrt{\text{trace}(X X^T Y Y^T)} \tag{4.12}$$

Using the joint model idea proposed while training JoSE, given a paragraph d represented by a subspace $D \in \text{Grass}(d, r_1)$, focus word u and context word v represented by elements $U, V \in \text{Grass}(d, r_2)$ we have

$$p(v, u|d) \propto p(v|u)p(u|d) \propto \exp(\text{sim}(V, U)) \exp(\text{sim}(U, D)) \tag{4.13}$$

Adding a constant, the expression becomes -

$$p(v, u|d) = \text{c}(D) \exp(\text{sim}(U, V)) \exp(\text{sim}(U, D)) \tag{4.14}$$

where c(D) is some function dependent on paragraph D. Our final max-margin loss function aims to maximize the probability of $p(v, u|d)$ for a positive training tuple (u, v, d) and minimizes the probability $p(v, u'|d)$ for a negative training tuple (u', v, d) is -

$$F(U, V, D) = \max(0, m - \text{sim}(V, U) - \text{sim}(U, D) + \text{sim}(V, U') + \text{sim}(U', D)) \tag{4.15}$$

where $m > 0$ is the margin, word v appears in the context of word u in a paragraph d, and u' is a randomly sampled word from the vocabulary serving as the negative sample.

Given the values of α - the learning rate, objective function F and word/paragraph orthonormal matrix X representing a subspace $\mathbf{X}$, we perform the following retraction operation -

$$X = \text{qr}(X + \alpha(I_k - X X^T) F_X) \tag{4.16}$$

4.5 Evaluations

In this section, we empirically evaluate the quality of our Grassmannian embeddings for four common tasks - word similarity, document classification, document clustering, and sentiment analysis much like `JoSE` . All the models have been trained for 10 iterations on the corpus, context window size was kept at 5, the embedding dimension was kept at 100 and the dimension of paragraph subspaces is denoted by r_1 while for words, it is denoted by r_2, which we choose accordingly as mentioned in each evaluation

subsection. The rest of the hyperparameters including the number of negative samples, initial learning rate, and margin were set to be the default value of the corresponding algorithm. However our model, due to being trained on a large corpus, performs all computations over an equally large set of moderately big matrices instead of vectors. Naturally, it is not as efficient as `JoSE` or any of the other text embedding models used here.

4.5.1 Word Similarity

Similar to `JoSE` , Word similarity evaluation was carried on the following datasets - WordSim353 [25], MEN [17] and Simlex-999 [39]. The training corpus used to train the data has also been kept the same - the 2019 wikipedia dump. The Spearman's rank correlation is reported in Table 4.2. We compared our model with the following baselines: `Word2vec` [55], `GLoVe` [63], `Poincaré GloVe` [72] and `JoSE` to highlight the different results in training in different manifold spaces. Table 4.1 shows the Spearman's rank correlation for the Simlex-999 dataset for the values of hyperparameter $1 \leq r_2 \leq 4$, r_1 has been kept as 1 as increasing the subspace dimension for words results in very poor score in all three datasets. We see that a 1-dimensional subspace combined with paragraphs as a 4-dimensional subspace produces the best results. Table 4.2 shows Spearman's rank correlation for all three benchmark datasets across all the embedding models. `JoSE` slightly outperforms our model suggesting that their directional similarity over Von Mises-Fisher distribution of words is more suitable than our subspace geometry.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.2845	0.2986	0.3012	**0.3082**

Table 4.1: Evaluation results for Word similarity on Wikipedia dataset. Pearson correlation scores for benchmark datasets - WordSim353, MEN, and Simlex-999 are provided in that respective order. Increasing the value of r_2 till 4 increases the score for Grassmannian embeddings.

4.5.2 Document Classification

We evaluate the quality of our document embeddings with the 20newsgroups[1] topic classification task. Each paragraph in the dataset is treated as a document much like in [52]. The dataset also contains around 18,000 newsgroup documents (both training and testing documents are used) partitioned into 20 classes We also keep the original test/train tests split and chose k-NN as the classification algorithm. k-NN algorithm on our Grassmannian manifold is performed simply by replacing the Euclidean metric in a Euclidean k-NN with the projection metric which is defined as $\frac{||XX^T - YY^T||_F}{\sqrt{2}}$ where $X, Y \in \text{Grass}(d, r_2)$. Everything else in the algorithm is kept the same. To compare with JoSE, k was kept at 3. Table 4.3

[1] http://qwone.com/~jason/20Newsgroups/

Embeddings	WordSim353	MEN	SimLex999
`Word2vec`	0.711	0.726	0.311
GloVe	0.598	0.690	0.321
Poincaré GloVe	0.623	0.652	0.321
JoSE	**0.739**	**0.748**	**0.339**
Grassmannian	0.698	0.745	0.331

Table 4.2: Pearson correlation score for benchmark datasets - WordSim353, MEN and Simlex-999 are provided in that respective order. Our best possible score is compared with other standard embedding models.

shows the change in the F1-macro score when the hyperparameters r_1 and r_2 are changed. We can see that when words and documents are both trained as a 3-dimensional subspace, we achieve the best results. Table 4.4 reports the F1-macro and F1-micro score across different text embedding models - Averaged word embedding using `Word2vec` [55], SIF [5], `Doc2vec` [45] and `JoSE` . Our Grassmannian model achieves the highest in both scores demonstrating that for document classification tasks, the topology of Grassmannian is much better suited than any other text embedding model.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.7307	0.7124	0.7210	0.7107
r_1=2	0.7448	0.7390	0.7297	0.7216
r_2=3	0.7380	0.72808	**0.7492**	0.7385
r_2=4	0.7265	0.7340	0.7449	0.7420

Table 4.3: F1 macro score of document classification on 20news dataset. Grassmannian document embeddings give different scores on changing values of r_1 and r_2. The optimal score is achieved at $r_1 = r_2 = 3$ suggesting the benefits of low-dimensional subspace

4.5.3 Document Clustering

We perform document clustering on our document embeddings and compare our result with the baseline as the same text embedding models used in the previous section. The training corpus is also the same as the previous one. As traditional k-means can't be applied to Grassmannian, we perform

Embedding	F1-Macro	F1-Micro
Avg. W2V	0.630	0.631
SIF	0.552	0.549
Doc2Vec	0.648	0.645
JoSE	0.703	0.707
Grassmannian	**0.749**	**0.752**

Table 4.4: F1-macro, F1-micro scores for k-NN (k=3) classification task with 20 Newsgroups dataset. Our best possible score is compared with other standard unsupervised embedding models.

Riemannian k-means [30] for our embeddings and traditional k-means for the others with the number of clusters as 20. The metrics are also chosen to be the standard ones - Adjusted Rand Index(ARI), Mutual Information(MI), Normalized Mutual Information(NMI), and Purity. Tables 4.6 compares results across models while table 4.5 shows the different Mutual Information(MI) scores we get when our Grassmannian model is trained with words as r_1-dimensional subspaces and paragraphs as r_2-dimensional subspaces for hyperparameters $1 \leq r_1 \leq 4$,$1 \leq r_2 \leq 4$. The best MI score is achieved for our model when words are 1-dimensional subspaces and paragraphs are 3-dimensional subspaces. `JoSE` outperforms our model in document clustering suggesting that clustering on a hypersphere is better suited for document clustering tasks than clustering on Grass(d, r_2).

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	1.855	1.801	**1.941**	1.781
r_1=2	1.781	1.834	1.776	1.851
r_1=3	1.797	1.715	1.734	1.566
r_1=4	1.630	1.719	1.613	1.779

Table 4.5: Mutual information score(MI) of document clustering task on the 20 Newsgroup dataset. Grassmannian document embeddings give different results for different values of r_1 and r_2. The best possible score is obtained at $r_1 = 1, r_2 = 3$.

Embedding	MI	NMI	ARI	Purity
Avg. W2V	1.299 ± 0.01	0.445 ± 0.009	0.247 ± 0.008	0.408 ± 0.014
SIF	0.893 ± 0.028	0.308 ± 0.009	0.137 ± 0.006	0.285 ± 0.011
Doc2Vec	1.856 ± 0.020	0.626 ± 0.006	0.469 ± 0.015	0.640 ± 0.016
JoSE	**1.975** ± 0.026	**0.663** ± 0.008	**0.556** ± 0.018	**0.711** ± 0.020
Grassmannian	1.943 ± 0.013	0.655 ± 0.09	0.498 ± 0.003	0.680 ± 0.009

Table 4.6: Mutual Information(MI), Normalized Mutual Information(NMI), Adjusted Rand Index(ARI) and Purity scores for k-NN (k=3) classification task with 20 Newsgroups dataset. Our best possible score is compared with other standard unsupervised embedding models.

4.5.4 Sentiment Analysis

Finally, we performed sentiment analysis on the imdb movie reveiew dataset[2] using kernel SVM. We use our similarity metric as the kernel trace(YY^TXX^T) and compare our results with the models used in the last two subsections. Table 4.7 lists the accuracy scores for the changing hyperparameters r_1, r_2. We see that the best accuracy is achieved when both words and documents are 1-dimensional subspaces. Table 4.8 lists the accuracy score for all the text embedding model trained. We see that our model slightly outperforms the other embeddings model suggesting that grassmannian kernel is effective for sentiment analysis.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	**88.9**	88.18	87.90	87.98
r_1=2	87.90	88.08	88.07	87.57
r_1=3	87.54	88.03	88.03	87.39
r_1=4	85.34	87.25	87.22	86.73

Table 4.7: Accuracy table for sentiment analysis with Kernel SVM of IMDB dataset for differing values of l and p. Best score is obtained at $r_1 = r_2 = 1$ suggesting the linearity of vector embeddings prove more effective in SVM binary classification.

[2] https://ai.stanford.edu/~amaas/data/sentiment/

Embedding	Accuracy
Avg. W2V	88.02
SIF	85.32
Doc2Vec	88.52
JoSE	87.95
Grassmannian	**88.90**

Table 4.8: Sentiment analysis accuracy results with SVM binary classification. Our best score is compared with other unsupervised embedding models.

4.6 Conclusion

In this paper, we extend the joint model idea used in training text embeddings in the n-hypersphere manifold to the Grassmannian manifold. Hence, each word/document has an entire subspace to encode all information learnt during training. Since vector dot product cannot be applied, we provide a new measure that captures the directional similarity between subspaces. We also show how to get the gradient from our objective function in this space and the corresponding update rule that is needed to be performed. This theory is validated by our results which show that Grassmannian text embeddings outperform or produce similar results when compared with other text embeddings in word similarity, document classification, clustering, and sentiment analysis tasks.

This paper is meant to serve as a groundwork for more involved research topics that integrate concepts of Grassmannian manifold learning, differential geometry, and NLP. There is much room for work in the current word embeddings itself. It would be interesting to see how to exploit the vectors that are part of a subspace. For instance, each sense of a polysemous word could be identified with a vector from the overall subspace representation of the word. Instead of training on a fixed dimensional subspace, each word could be represented with a different dimensional subspace that is determined by the number of senses it exhibits in the corpus. This would open up a new avenue for geometric algebraic operations to be integrated like subspace meet and join. The representations could also be used for evaluating word sense disambiguation tasks, lexical entailment, word analogy, and compositionality. To go even further, neural network architectures like bi-LSTM and CNN could be specifically designed with Grassmannian manifolds in mind, which would make the representations more suitable for already established complicated architecture for question-answering tasks, machine translation tasks, and so on.

Chapter 5

Long Matrices Text Embeddings

5.1 Introduction

Most unsupervised text embedding models are trained by encoding the words or paragraphs acquired from the training data as a feature-length vector, with the assumption that they reside in an Euclidean space. Such models are ubiquitous for good reason. Aside from their efficiency, they have also proven to be very effective providing us with state-of-the-art results in various intrinsic and extrinsic embedding evaluation tasks. `Word2vec` [55, 56], and `GLoVe` [63] are two notable examples where word embeddings are learned in the Euclidean space and are trained to be oriented such that word vectors that appear in the same context have higher cosine similarity. Some of the most common methods of intrinsic evaluation of word embeddings include word similarity, word analogy, and compositionality. `Doc2vec` [45], an unsupervised document embedding model generalizes the training method introduced in `Word2vec` to documents and achieves improved results in various downstream tasks like sentiment analysis, information retrieval, and multi-class classification. There are other document embedding models like skip-thought [43] and infersent [19, 58].

The joint spherical embedding model, `JoSE` as proposed in [52], shows that directional similarity is often more effective in tasks such as word similarity and document clustering. They show that when embeddings are trained in the Euclidean space, there is a performance gap between the training stage and usage stage of text embeddings. To bridge that gap, they propose a model which trains both words and paragraphs on a spherical space with tools from Riemannian optimization methods. The resulting embeddings are also shown to give considerably better results in word similarity, document clustering, and document classification tasks when compared with other standard models. Such application of manifold geometry has also been explored in substantial depth in works like [11, 66, 33]. There is also another notable Riemannian optimization-based embedding training model like [62, 72] which trains embeddings on the hyperbolic manifold space and uses its tree-like property for better hierarchical representation of data. Hyperbolic word embeddings are also intrinsically linked with Gaussian word embeddings [75] which gives a lot more insight into the geometry of word embeddings.

However, most of these text embedding models like JoSE , Word2vec, Doc2vec, and fastText [14, 53] are trained with the goal of getting a single dense vector representation per word or document. These models treat both polysemous and monosemous words, in the same way, resulting in the most frequent meaning of the word dominating the others or the meanings getting mixed in the case of the former. It is especially detrimental for documents where we use a single dense vector representation to encode information that spans several sentences, often involving multiple topics.

This chapter proposes a model to address this problem by using matrices as the mode of representation instead of vectors. Our model is the joint word and document training generative model proposed in JoSE where we replace the cosine similarity metric with a novel metric that exploits the matrix structure of the embeddings. This robust metric takes a pair of word/document matrices of an arbitrary number of columns and calculates the similarity between them. We also show that a few reshape operations allow us to reformulate the optimization problem of our model in terms of the spherical manifold optimization problem. Thus, we offer more flexibility in the way of matrix dimensions while retaining efficiency. Our choice of metric also suggests that the word, sentence, and paragraph/document embeddings do not need to have the same number of columns, which has linguistic validation.

5.2 Matrix Representation of Texts and Optimization Problem

The text embeddings are represented as elements of the following set

$$\mathcal{S}(p, r) = \{\mathbf{X} \in \mathbb{R}^{p \times r} : ||\mathbf{X}||_F = 1\},$$

where $r \leq p$ and $\|\cdot\|_F$ denotes the Frobenius norm. The Frobenius norm is the matrix norm of a $p \times r$ matrix $\mathbf{X}$ defined as the square root of the sum of the absolute squares of its elements, i.e.,

$$||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{p} \sum_{j=1}^{r} x_{ij}^2}.$$

Our model design is consistent with JoSE where it is assumed that text generation is a two-step process: a center word is first generated according to the semantics of the paragraph, and then the surrounding words are generated based on the center word's semantics. Consider a positive tuple $(\mathcal{U}, \mathcal{V}, \mathcal{D})$ where word $\mathcal{V}$ appears in the local context window of word $\mathcal{U}$ in paragraph $\mathcal{D}$ and negative tuple $(\mathcal{V}, \mathcal{U}', \mathcal{D})$ where $\mathcal{U}'$ is a randomly sampled word from the vocabulary serving as a negative sample. We represent words $\mathcal{V}, \mathcal{U}, \mathcal{U}'$ as matrices $\mathbf{V}, \mathbf{U}, \mathbf{N}$ which are elements of the set $\mathcal{S}(p, r_1)$ and paragraph $\mathcal{D}$ as matrix $\mathbf{D}$ which is an element of the set $\mathcal{S}(p, r_2)$, where $p, r_1, r_2 > 0$. From a linguistic perspective, these matrices can be considered as a set of latent variables that govern the semantics of a word or a document. Each column is given some arbitrary unit of linguistic information to encode, a latent variable that contributes to the mathematical representation of a word or a document. For example, the columns of a matrix $\mathbf{D}$ that represents the document $\mathcal{D}$ might encode latent variables that contain information about some topic contained in that document. Similarly, the columns of the word

matrix $\mathbf{U}$ might encode information about a specific context in which a polysemous word $\mathcal{U}$ appears. We also keep the number of columns for word matrices less than or equal to the number of columns for sentence/document matrices, i.e., $r_1 \leq r_2$, so that the number of latent variables governing a word should not be more than the ones that govern a sentence or paragraph.

Novel metric. To model the above mentioned linguistic representation mathematically, we define a novel similarity metric for the ambient space in which we train our matrix embeddings. The proposed metric function is a measure of similarity between two sets of latent variables (matrices) - a function analogous to the cosine similarity measure for vectors in the Euclidean space. Given two arbitrary matrices $\mathbf{A} \in \mathcal{S}(p, r_1), \mathbf{B} \in \mathcal{S}(p, r_2)$, we propose the similarity metric $\mathbf{g} : \mathcal{S}(p, r_1) \times \mathcal{S}(p, r_2) \to \mathbb{R}$ as

$$\mathbf{g}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{r_1} \sum_{j=1}^{r_2} a_i^\top b_j}{r_1 r_2}, \tag{5.1}$$

where $\mathbf{A} = [a_1\, a_2\, \cdots\, a_{r_1}]$, $\mathbf{B} = [b_1\, b_2\, \cdots\, b_{r_2}]$, $a_i, b_j \in \mathbb{R}^p\ \forall i \in [1, 2, \cdots r_1]$, $\forall\, j \in [1, 2, \cdots r_2]$.

Motivation for our similarity metric. The metric $\mathbf{g}$ (5.1) calculates the average of all the entries in the matrix $\mathbf{A}^\top\mathbf{B}$. The linguistic intuition behind the choice of this metric is that we want to define a metric that takes the average of dot products between all possible pairs of latent variables (columns) from each matrix. In the case of $r_1 = r_2 = 1$, $\mathbf{g}(\mathbf{A}, \mathbf{B})$ reduces to the cosine similarity metric between unit norm vectors $\mathbf{A}$ and $\mathbf{B}$ which is the metric used in the spherical space model of `JoSE` . However, in the case of higher values of r_1, r_2, For example, let two words $\mathcal{V}_1$ and $\mathcal{V}_2$ be represented by proposed $p \times r_1$ matrix embeddings - $\mathbf{V}_1 = [\mathrm{a}_1, \mathrm{a}_2]$ and $\mathbf{V}_2 = [\mathrm{b}_1, \mathrm{b}_2]$, where $p = 1$ and $r_1 = 2$. The proposed similarity metric $\mathbf{g}(\mathbf{V}_1, \mathbf{V}_2)$ is computed as $(a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2)/4$. Note that this is different from computing the cosine similarity which gives $(a_1b1 + a_2b_2)$. Moreover, the regular cosine similarity between word and paragraph embedding matrices with unequal dimensions ($p \times r_1$ and $p \times r_2$ respectively) is not defined. On the other hand, our proposed similarity metric is still applicable.

Modelling. As our model has the same generative process as `JoSE`, we take the same max-margin loss function and substitute the cosine similarity metric with our new similarity metric $\mathbf{g}$ where the word matrices $\mathbf{U}, \mathbf{V}, \mathbf{N} \in \mathcal{S}(p, r_1)$ and paragraph matrix $\mathbf{D} \in \mathcal{S}(p, r_2)$ with $r_1 \leq r_2$. We get the following loss, i.e.,

$$\begin{aligned} &\mathcal{L}(\mathbf{V}, \mathbf{U}, \mathbf{N}, \mathbf{D}) \\ &\quad = \max\,(0, m - \mathbf{g}(\mathbf{V}, \mathbf{U}) - \mathbf{g}(\mathbf{U}, \mathbf{D}) \\ &\qquad\qquad + \mathbf{g}(\mathbf{V}, \mathbf{N}) + \mathbf{g}(\mathbf{N}, \mathbf{D})). \end{aligned} \tag{5.2}$$

where $m > 0$ is the margin.

Optimization. For the purpose of optimization, matrices of different dimensions are reshaped and embedded into Riemannian spherical manifolds of different dimensions. Overall, they are combined using the Riemannian product manifold structure. Therefore, the optimization of $\mathcal{L}$ (5.2) is done by performing two reshape operations per iteration while training. For example, the unit Frobenius norm matrices of dimension $\mathbb{R}^{p \times r}$ can be reshaped into vectors of dimension $\mathbb{R}^{pr}$ with the unit norm. To calculate the value of our loss function (5.2) at every iteration and the Euclidean gradient (partial derivatives), the vectors in question are reshaped into matrices for calculating the $\mathbf{g}$ values and their

gradients. Subsequently, the matrices are reshaped back into vectors. We then apply the Riemannian gradient descent update rule to update the parameters [52, 2, 69, 22]. Note that our proposed modeling and optimization are different from just training on the spherical manifold with unit vectors and using the cosine similarity metric (which is the case in JoSE).

5.3 Experiments

To highlight the quality of our obtained matrix representations, we run the same set of evaluations as JoSE with a relatively lower number of columns, i.e., $1 \leq r_1 \leq r_2 \leq 6$. We notice that for even higher values, the quality of our embeddings gradually decreases. We also add semantic textual similarity benchmark tests to show that sentences can benefit from this matrix representation model. Unless otherwise stated, our model and JoSE are trained for 35 iterations on the respective corpora; the local context window size is 5; the embedding dimension is kept at 100; the number of negative samples is 2. Other hyperparameters in our model are kept the same as JoSE .

5.3.1 Word Similarity

We run Word similarity evaluation on three benchmark datasets -WordSim353 [25], MEN [17] and Simlex-999 [39]. The training corpus chosen was a part of the 2019 Wikipedia dump. The Spearman's rank correlation scores are reported in Table 5.1 for various values of r_1 and r_2. As it can be seen, the best result is obtained when $r_1 = r_2 = 1$ which suggests that there is no intrinsic benefit in modeling words as matrices. It is largely due to the fact that such standard word evaluation tasks focus on evaluating a "general" sense of meaning between the words. As more columns are added to the words, the senses kind of spread out in the columns making it hard to compute similarity in such a fashion.

Table 5.1: Evaluation results for Word similarity on Wikipedia dataset ($r_1 = 1, r_2 = 1$ is JoSE score). Pearson correlation scores for benchmark datasets - WordSim353, MEN, and Simlex-999 are provided in that respective order. Word embeddings do not benefit from matrix representations as the best score is obtained when $r_1 = 1, r_2 = 1$

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	**0.6921, 0.7109, 0.3218**	0.6885, 0.7007, 0.3131	0.6755, 0.6940, 0.3027	0.6589, 0.6828, 0.3019
r_1=2	–	0.6472, 0.6630, 0.2903	0.6470, 0.6560, 0.2862	0.6341, 0.6508, 0.2801
r_1=3	–	–	0.5841, 0.5987, 0.2559	0.5728, 0.5929, 0.2534
r_1=4	–	–	–	0.4983, 0.5450, 0.2290

5.3.2 Document Clustering

We perform document clustering on the 20 Newsgroup[1] dataset using spectral clustering. Each paragraph in the dataset is separated by a new line and is considered a separate document while training. JoSE uses K-Means and SK-Means as the clustering algorithm that assumes the ambient space to be the Euclidean and the spherical space, respectively. Our non-Euclidean space with its custom metric requires a clustering algorithm that allows the freedom of using custom metric, i.e., the algorithm should be space agnostic. We found spectral clustering to suit those requirements perfectly. The four external measures used for validating the results are kept unchanged from JoSE [8, 51, 71]. These measures are Mutual Information (MI), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Purity. We run the clustering algorithm with our custom similarity metric as written in (5.1) with kernel coefficient, $\gamma = 0.001$. Table 5.2 shows quantitatively how matrix representations benefit document embeddings for clustering tasks. Keeping $r_1 = 1$ fixed, we see a steady increase in performance as r_2 is increased from 1 (the score of our baseline model - JoSE) to 6.

Table 5.2: Evaluation results for spectral clustering of document embeddings on the 20 Newsgroup dataset for kernel coefficient, $\gamma = 0.001$ ($r_1 = 1, r_2 = 1$ is JoSE score). Document embeddings benefit from matrix representations as demonstrated by better scores for higher values of r_2. r_1 is set as 1 for all the different r_2 values

$r_2 \backslash r_1 = 1$	MI	NMI	ARI	Purity
$r_2 = 1$	1.73	0.58	0.45	0.64
$r_2 = 2$	1.75	0.59	0.46	0.63
$r_2 = 3$	1.75	0.59	0.46	0.62
$r_2 = 4$	1.77	0.60	0.46	0.63
$r_2 = 5$	1.84	0.62	0.49	0.65
$r_2 = 6$	**1.85**	**0.62**	**0.49**	**0.67**

5.3.3 Document Classification

Following [52], the document classification evaluations are ran on the following two datasets: the topic classification 20 Newsgroup dataset (which we used for document clustering as well) and a binary sentiment classification dataset consisting of 1 000 positive and 1 000 negative movie reviews[2]. The train/test split is the original split for 20 Newsgroup while for the movie review datasets, the splitting

[1] http://qwone.com/~jason/20Newsgroups/
[2] http://www.cs.cornell.edu/people/pabo/movie-review-data/

Table 5.3: F1-macro, F1-micro for 20 Newsgroup dataset classification using K-NN with K=3 ($r_1 = 1, r_2 = 1$ is `JoSE` score). Increasing the value r_2 benefits documents embeddings in classification tasks.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.74, 0.74	0.77, 0.77	0.78, 0.78	**0.78, 0.78**
r_1=2	–	0.76, 0.76	0.77, 0.77	0.76, 0.77
r_1=3	–	–	0.76, 0.76	0.73, 0.74
r_1=4	–	–	–	0.72, 0.72

Table 5.4: F1-macro, F1-micro for movie review dataset classification using K-NN with K=3 ($r_1 = 1, r_2 = 1$ is the `JoSE` score).

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.74, 0.74	0.75, 0.75	0.76, 0.76	0.75, 0.76
r_1=2	–	0.75, 0.75	0.75, 0.75	0.74, 0.74
r_1=3	–	–	0.74, 0.74	**0.76, 0.76**
r_1=4	–	–	–	0.74, 0.74

is done by randomly selecting 80% of the data as training and 20% as testing. The classification algorithm we use is K-NN with $k = 3$ and a custom distance metric that is suitable for our space. The custom distance metric for two paragraph matrices $\mathbf{U} = [u_1\, u_2\, \cdots\, u_{r_2}]$ and $\mathbf{V} = [v_1\, v_2\, \cdots\, v_{r_2}]$ where $\mathbf{U}, \mathbf{V} \in \mathcal{S}(p, r_2)$ and $u_i, v_i \in \mathbb{R}^p\ \forall i \in [1, 2, \cdots, r_2]$ is defined as

$$\text{dist}^2(\mathbf{U}, \mathbf{V}) = \frac{\sum_{k=1}^{r_2} \sum_{l=1}^{r_2} (u_k - v_l)^\top (u_k - v_l)}{r_2^2}. \tag{5.3}$$

The intuition for the distance metric in (5.3) comes from our interpretation of each individual column as encoding a latent variable governing the semantics of that specific document. A quick look at (5.3) tells us that the distance metric takes the square root of the average of the squared Euclidean distances between all pairs of columns formed from one matrix with another. Tables 5.3 and 5.4 list the Macro-F1 and Micro-F1 scores for 20 Newsgroup dataset and Movie Reviews dataset respectively for increasing values of r_1 and r_2. We again see an increase in scores for higher values of both r_2 and r_1 compared to `JoSE` (r_1=1, r_2=1).

5.3.4 Semantic Textual Similarity Task

Semantic Textual Similarity Benchmark comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval [18] between 2012 and 2017[3]. We perform semantic textual similarity tasks on the sts-benchmark dataset to show that even sentences can benefit from being represented as matrices. The benchmark comprises of 8 628 sentence pairs split into 3 partitions: train, development and test. The results are reported on both the test and dev sets. Each sentence in the dataset is treated as a separate document by our model and we use all the sentences in the train, development and test set to train. The rationale for this is that the model is completely unsupervised, i.e., it takes only the raw text and uses no supervised or annotated information, and thus there is no need to hold out the test data as it is unlabelled. We train for 1 000 iters with window size 15 and negative samples 5 while the rest of the hyperparameters were kept at their default values. To score a sentence pair representation, similarity was computed between them using our custom metric described in 5.1 for our model. We report the dev and test Pearson correlation score for $r_1, r_2 = 1, 2, 3, 4$, $r_1 \leq r_2$. As Table 5.5 reports, higher values of r_2 give better scores compared to our baseline model `JoSE` ($r_1 = r_2 = 1$).

Table 5.5: Pearson Correlation for STS Benchmark on dev and test data ($r_1 = 1, r_2 = 1$ is the `JoSE` score). Even sentences can benefit from our matrix representation as demonstrated by better scores with higher values of r_2.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.51, 0.40	0.51, 0.39	0.52, 0.40	0.53, 0.40
r_1=2	–	0.53, 0.40	0.53, 0.40	0.53, 0.40
r_1=3	–	–	0.53, 0.40	**0.54, 0.40**
r_1=4	–	–	–	0.53, 0.40

5.4 Conclusion

In this chapter, we extend the joint modeling idea used for training text embeddings from vectors with unit norm to matrices with unit Frobenius norm. Each word/sentence/document matrix is made to encode information in a way that each column of the matrix represents some latent topic, context, or discourse. Since the standard vector dot product can no longer be applied, we introduce a novel similarity metric that allows the measurement of similarity between matrices of an arbitrary number of columns. For optimization simplicity, we reshape our matrices to vectors of unit norm that allow using the Riemannian gradient descent optimization algorithm on the spherical manifold. Our theory

[3] http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark

is validated quantitatively by the results which show that our text embeddings outperform or produce similar results when compared with `JoSE` in document classification, clustering, and semantic textual similarity tasks.

This paper is meant to serve as a groundwork for more involved research topics which integrate concepts of differential geometry and NLP. Future directions could include qualitative analysis of the columns of the matrices to see what tangible information they encode which will allow for better modeling. Another direction would be to exploit other matrix structures on the embeddings, e.g., treating each word embedding as a symmetric positive definite matrix and studying whether they can be beneficial.

Chapter 6

Spectahedron Manifold Word Embeddings

6.1 Multi-sense embedding structure

The best form of word representation to tackle the multi-sense problem would be to have a single mode of representation per word that encodes the multiple senses of the said word such that some computationally efficient function can break down the representation to produce a single vector representation for each sense. As we saw, two possible ways to do that would be to encode the words as points in a Grassmannian manifold, which are essentially r-dimensional subspace for some $r \in \mathbf{N}$ (chapter 4) and long matrices where we associate each sense of a word to a matrix column explicitly (chapter 5). In this current chapter, we build upon the Grassmannian manifold idea and add some new constraints that are more suited for the problem we are trying to solve. Points in a Grassmannian manifold are subspaces which can be computationally represented as orthonormal matrices of size $d \times r$ where d is the dimension of the entire vector space and r is the dimension of the subspace. Since the senses of a word are largely unrelated to each other, we conjectured that each non-zero linearly independent eigenvector that makes up the subspace would represent an individual sense of the word. Therefore, a full rank matrix of rank r should encode r different senses of the word. From a practical standpoint, even though there is an improvement in a handful of qualitative tasks in the Grassmannian chapter, we could not find any quantitative evidence to support the strong conjecture we put out. We rectify our conjecture by introducing a sense of "weights" to the senses. A particular sense will be more prominent if it occurs more frequently in the corpus. In fact, there is a high possibility of one sense of a word completely dominating another owing to its higher occurrence in the corpus. So, it becomes necessary to somehow normalize the frequency of occurrence of all the senses associated with the word. Mathematically, these weights are the corresponding eigenvalues of the eigenvectors. The manifold of symmetric positive semi-definite matrix will allow us to encode exactly that. We propose an efficient model to train words in a Spectahedron manifold and provide qualitative results that show the above model provides a better representation of polysemous words than single vector representations. We also provide an Information geometric perspective to our embeddings that arises from the choice of our similarity metric and helps explain our experimental observation.

6.2 Related Works

The issue of polysemy in contextualised embeddings have been addressed before through sense representation models. [61] proposes the multi-sense skip-gram model which builds upon the original skip-gram architecture of `Word2vec` by clustering together the similar contexts in which a focus word appears and having individual vector representation for each context cluster. These individual representations are the sense representations of that word. They also added flexibility by having a parametric and non-parametric version of their model. Similarly, [10] proposed a non-parametric Bayesian version of the Skip-gram model that turns out to be capable of automatically learning the required number of senses for each word in the corpus. [64] takes pre-trained embeddings and "de-conflate" them using an external semantic inventory like `WordNet`. They also show that in downstream tasks like topic categorization, multi-sense representations outperform pre-trained single vector word embeddings. [41] also does something similar by proposing a topic modeling based skip-gram approach for learning multi-prototype word embeddings.

Information geometric representation of words, especially from a Gaussian perspective, has also gained quite a bit of traction ever since [75] proposed `Word2Gauss`. Instead of the traditional position based representation in `Word2vec` and `GLoVe` , `Word2Gauss` is a density-based distributed embeddings which presents a method for learning representations in the space of Gaussian distributions. They also introduce an asymmetric measure of similarity between words using KL-divergence which has been seen to model hypernymy relations between words where the distribution for one word (e.g. 'music') encompass the distributions for sets of related words ('jazz' and 'pop'). [72] proposes training words as points in the hyperbolic manifold(Poincare disk and Poincare ball) and provides a mathematical connection between their representation and `Word2Gauss` that helped explain the hypernymy-hyopnymy relations in Gaussian word embeddings better. The `Word2Gauss` model is extended even further when [7] proposed a Gaussian mixture model to account for the multiple senses of the word. Each sense of a word is given a Gaussian distribution which are then linearly summed to get the Gaussian distribution for the word as a whole. Qualitative results are provided to show that they capture semantic information better than skip-gram and Gaussian embeddings in word similarity and lexical entailment. [65] builds on their idea to provide Gaussian mixture embedding for a word where each component is a Gaussian embedding representation of a sememe(unit of semantic information in a word). The loss function proposed also explicitly takes into account synonymy and hypernymy information and models such semantic relations well enough to produce state-of-the-art results in word similarity and lexical entailment. They also propose a novel "contextualizer" model which takes Gaussian embeddings as input and produces contextualised Gaussian representation for context-sensitive tasks such as named entity recognition and text classification.

To our knowledge, this is the first work where Spectahedron manifold is used to as the latent space for word representation. The work is also quite novel in that it uses a standard matrix function like singular value decomposition on a overall word representation(matrix) to extract the different sense representations of a word.

6.3 Spectahedron Manifold

Let $U \in \mathbb{R}^{d \times r}$ be a full rank matrix. A word w is represented by $d \times d$ symmetric positive semi-definite matrix UU^T whose rank is r. For homogeneity, we want the eigenvalues of matrix UU^T to be non-negative and sum to 1. Given the fact that all the eigenvalues of a positive semi-definite matrix are already non-negative and the trace of a matrix is equal to the sum of its eigenvalues, we can add the constraint $\text{tr}(UU^T) = ||U||_F = 1$ to the optimization space of U where $||.||_F$ is the frobenius norm operator. Any optimization problem on this matrix space of U with the above constraint can be turned into a unconstrained optimization problem in the Spectahedron manifold. [42] discusses Spectahedron manifold in detail. The Spectrahedron manifold, also known as the set of correlation matrices (symmetric positive semi-definite matrices) of rank r with unit trace is mathematically defined as -

$$\mathcal{S}(d,r) = \{Q \in \mathbb{R}^{dxd} | a^\top Q a \geq 0 \quad \forall a \in \mathbb{R}^n \ , \text{tr}(Q) = 1, \\ Q = UU^T \text{ for } U \in \mathbb{R}^{d \times r} \text{ and } \text{rank}(Q) = \text{rank}(U) = r\} \quad (6.1)$$

Please also note that $r \leq d$ for all intents and purposes. We now need matrix representations for the tangent space, closed form expression for the gradient and a retraction operation to perform Riemannian gradient descent on the manifold. The tangent space representation $T_Q\mathcal{S}(d,r)$ for a point $Q \in \mathcal{S}(d,r)$ is given by -

$$T_Q\mathcal{S}(d,r) = \{Z \in \mathbb{R}^{d \times r} : \ \text{tr}(Z) = 0 \ , Z^\top Q = Q^\top Z\} \quad (6.2)$$

A natural choice of riemannian metric for the above tangent space is $\text{tr}(A^T B)$ where $A, B \in T_Q\mathcal{S}(d,r)$. The gradient is discussed in subsection 6.3.3. The Riemannian gradient at Q for a function F, $\text{grad}_X F$ is calculated as follows:

$$\text{grad}_X F = \nabla_Q F - \text{tr}([\nabla_Q F]^\top X)X. \quad (6.3)$$

where $\nabla_Q F$ is the euclidean gradient of F at point Q. Finally, a natural retraction of choice which is also computationally feasible is -

$$Ret_Q(Z) = \frac{Q + Z}{||Q + Z||_F} \quad (6.4)$$

where Z is the search direction on the tangent space $T_Q\mathcal{S}(d,r)$,

6.3.1 The Optimization problem from Spectahedron to Spherical

We have a positive training pair (u, v) where word v appears in the local context window of word u and negative training tuple (u, n) where word n is negatively sampled and not present in the context window of u. Let word u, w and n be represented by the positive semi-definite matrices $\bar{U} = UU^T$, $\bar{V} = VV^T$, $\bar{N} = NN^T$ where U, W, V are all $d \times r$ size matrices. This means we would have to optimize with computations on square matrices $\bar{U}, \bar{V}, \bar{N}$. This becomes an extremely expensive operation when the dimension size, d is higher. Therefore, we optimize instead over the unit norm matrices U, V, N and with some careful observation, we find that the unit norm matrices and the gradient formula calculation

allows the spherical geometry optimization ideas to be used. Similar to our optimization procedure in the previous model, the matrices of size $d \times r$ are reshaped and stored as vectors of dimensions dr. At each step, the vectors in question are reshaped back into matrices during the calculation of the loss function and Euclidean gradient. Following that, they are reshaped into the vectors again and the Riemannian gradient expression of the spherical geometry is used because the gradient expression would have been the same but in the matrix form if it were kept as matrices. Finally, the retraction operation for the sphere is used for the update as they enforce the unit norm again. We express the loss function in terms of the matrices and derive the expressions for the Euclidean gradients of the loss with respect to these matrices.

6.3.2 Metric and Loss Function

We have the choice of two different metrics to train our Spectahedron manifold. The first one is a similarity metric derived from the Bures-Wasserstein distance while the second metric is the Frobenius-norm between the two word matrices.

$$S_{BW}(U, V) = \text{tr}(\text{sqrtm}((U^\top V V^\top U))) \tag{6.5}$$

$$S_{FR}(U, V) = \sqrt{\text{tr}(U^\top V V^\top U))} \tag{6.6}$$

where sqrtm(.) is the matrix square root of a positive semi-definite matrix. We use the Bures-Wasserstein metric as it has interesting geometric implications we discuss later. Note that the Frobenius-norm metric has faster computation of gradients compared to BW metric due to the latter using matrix square root operation in its gradient calculation. Note that both S_{BW} and S_{FR} admit values in the range of $[0, 1]$. Since the metric outputs are taken on a very limited range of values, a ranking-based loss is more suitable for our model. A max-margin ranking objective, similar to that used in Rank-SVM [76] or Wsabie [76] or in Word2Gauss and JoSE, which pushes scores of positive pairs above negatives by a margin. So, the loss function for our Skip-gram negative sampling model is

$$L_m(U, V, N) = \max(0, m - S_{BW}(U, V) + S_{BW}(U, N)) \tag{6.7}$$

where $m > 0$ is the margin, U is the focus word matrix, V is the context word matrix and N is a negatively sampled word matrix.

6.3.3 Gradients

Closed form euclidean gradient expression for U,V and N are given by the following:

$$\frac{\partial L}{\partial U} = -2V(V^\top U)\text{dsqrtm}(V^\top U U^\top V, I) + 2N(N^\top U)\text{dsqrtm}(N^\top U U^\top N, I) \tag{6.8}$$

$$\frac{\partial L}{\partial V} = -2U(U^\top V)\text{dsqrtm}(V^\top U U^\top V, I) \tag{6.9}$$

$$\frac{\partial L}{\partial N} = 2U(U^\top N)\text{dsqrtm}(N^\top UU^\top N, I) \tag{6.10}$$

where $\text{dsqrtm}(X, Z)$ is the derivative of matrix square root of X along the matrix Z and I is an Identity matrix of order r.

6.4 A Gaussian Embedding Perspective

The Bures-Wasserstein similarity metric allows each word embedding to be interpreted as a zero-centred multivariate Gaussian distribution [50]. For $r < d$, the symmetric positive semi-definite matrix of a word embedding is equivalent to the positive semi-definite covariance matrix i.e. $\mathcal{N}(0, U)$ for a word matrix U. In the case of $r = d$, the covariance matrix is positive definite(full rank). We know that the covariance matrix defines both the spread (variance), and the orientation (covariance) of any data. The eigenvector with the largest eigenvalue is the direction along which the data set has the maximum variance and for successive eigenvalues the data set variance goes down. As we will see in the next section, the eigenvectors are associated with the senses of the word. This suggests that the most dominant sense of the word has the maximum variance and the least dominant sense has the least variance. If each occurrence of the word in the corpus is one "data sample", we are essentially modelling the total occurrences of that word in that corpus as the total data space of that word where each senses moves towards its respective cluster. Such a behaviour is actively forced as a part of modelling in [61] but due to the geometry of Spectahedron it occurs more naturally for us, albeit a little weakly.

6.5 Qualitative analysis of Polysemous Words

We perform eigen decomposition of the Spectahedron matrix of a polysemous word (e.g. apple). Each of the resulting eigenvectors represents each individual sense of the word "apple" with the corresponding eigenvalues representing the "weight" of that sense in the corpus. For practical purposes, we extract only the top r eigenvectors since the rest $d - r$ eigenvectors of $d \times d$ matrices have eigenvalues close to 0. So, each word has now r vectors to represent its r senses. We run a nearest neighbour word search on these $r \times V$ words (V = total number of words in the vocabulary) with cosine similarity as the metric of similarity between vectors. The tables below show how each eigenvector of a word captures one specific sense of it. We observe pretty interesting trends. For example, in the case of "apple" at r=5, words in the context of Newton discovering gravity seem to show up as the fifth eigenvector. The linguistic implication of such results is huge as one could use this technique to see how word contexts have changed over centuries by training various historical texts in different periods of time.

Table 6.1: Nearest Neighbor words of the eigenvectors of a few spectahedron words trained on the BW metric with $d = 200, r = 3$.

Word	First Eigen Vector	Second Eigen Vector	Third Eigen vector
apple	macintosh, amiga, mac, os, pc, ibm, microcomputer, compatibles, hardware, gui, commodore, clones, software, iic, workstations	pomegranate, palm, teas, berries, flowers, peripherals, tart, pears, grapevine, herbs, roasted, olive, barley, mango	dark, oil, cup, cultivar, much, widely, consists, watershed, mediterranean, stages, solution, lemon, yellow, leaves, fruits
DC	wildstorm, watchmen, superheroes, supervillain, miniseries, fantagraphics, superpowered, supergirl, nightwing, smallville	amplifier, signals, size, bandwith, signal, frequency, transistor, connectors, impedance, transistors, analog, setup, device	film, makeup, external, returns, ninja, billion, million, sandman, volume, director, marvel, detective, clerks, villains
soul	immortality, eternal, god, mind, divine, atman, spirit, afterlife, consciousness, love, spiritual, heaven, souls, grace, everlasting, contemplation, goodness, essence, intellect	soul, judas, synthpop, muggs, sampling, hit, rappers, righteous, adult, cure, folk, djs, acts, pianists, arrangements, electric, sweat, alternative, radiohead	unconscious, obelix, despised, window, gets, proofs, triune, gnomes, runs, devil, table, balcony, tubulin, ousia, adventures
matrix	matrices, linear, dimensional, vector, eigenvalues, function, invertible, multiplication, scalars, rref, reloaded, finite, polynomial, vectors, algebra, equation, euclidean, exponential, integers	divides, temperature, composite, vesticules, polynomial, protects, synpases, binds, factors, insulating, gradient, activation, rings, fibrils, solutions pde, equivalently	sizes, hid, heisenberg, end, basic, accelerator, hamiltonian, usenet, tab, observables, limit, closet, user
bill	clinton, johnson, frist, carter, jim, tony, bob, mike, joe, president, act, gerry, larry, steve, dave	british, reality, uk, candian, house, legislation, played, owner, entitled, andy, prime, members, enacted, king, last, japanese	bit, war, album, feet, science, vote, until, broadcast, greatest, announced, won, meters, saw, completed, age

Table 6.2: Nearest Neighbor words of the eigenvectors of a few spectahedron words trained on the BW metric with $d = 300, r = 5$.

Word	First Eigen-Vector	Second Eigen-Vector	Third Eigen-vector	Fourth Eigen-Vector	Fifth Eigen-Vector
apple	macintosh, mac, os, pc, ipod, ibm, desktop, iigs, windows, imac, gui, iic, pcs, intel, powerpc, hardware, commdore	honey, lotus, leaves, fruit, sweet, juice, machine, drink, fermented, berries, beans, grape, apples, leaf, water, bread	performed, popularity, writing, production, several, roman, closed, teaching, latin, moving. program, hands, moving, discussion, terms	king, mystery, chronicles, news, billboard, caustic, singles, revenge, curtain, river, seer, trucker, homer, lasted, genre, remixes, knuckles, album, sleeve, beastie	events, plate, gravity, market, inertia, exchange, grounds, falls, bodies, chronicle, local, rotational, causality, acceleration, newton, meaning, gravitation, providence, inertial, rotation, refraction, water
DC	marvel, superman, washington, batman, kirby, supergirl, lois, luthor, smallville, superboy, clark, waid, superpowered, romita, miniseries, schengen, lex	locations, washington, villains, superhero, broadway, prince, strip, space, science, series, conventions, bird, theater, has	voltage, amplifier, circuits, wired, resistor, known, single, impedance, voltages, circuit, transistors, resistors, device	persons, land, increase, sq, months, about, statute, income, amounted, degrees, than, km, species, hours, hundred, kiss, passengers, election, typically, weight	group, website, db, engine, strategy, replace, acronym, years, controversy, islands, future, europe, mitsubishi, official, aims, footnotes, bbci
sin	sins, god, eternal atonement, venial, damnation, repentance, satan, heaven, everlasting, sinner, redemption, righteousness, depravity, mortal, apostasy	cos, case, cdots, frac, cdot, greek, qquad, y, let, pi, quad, ldots, exp, dt, theta, begin, exp. sqrt, mathbf	dimension, matrix, dog, space, roles, little, emotions, special, stone, heaven, god, child, good, features, show	official, duplicated, value, freely, product, heritage, dominated, devil, decline, stress, possibilites	writing, surpassing, metropolitan, mathematician, wolf, ring, needless, detroiters, burrito, circling, mayor, education, skyline, vestibule

Chapter 7

Word Analogy - a Lie group perspective

7.1 Lie Groups - a primer

A Lie group is a smooth manifold $\mathfrak{G}$ together with a group structure. $\mathfrak{G}$ as a group has the associated group operation $\odot : \mathfrak{G} \rightarrow \mathfrak{G}\times\mathfrak{G}$ and the identity element $E \in \mathfrak{G}$ such that inverse map $(.)^{-1} : \mathfrak{G} \rightarrow \mathfrak{G}\times\mathfrak{G}$ can be defined. Both the group operator and the inverse operator are smooth(infinitely differentiable) with respect to the topology of the manifold structure. The simplest example of a Lie group is the Euclidean vector space, $\mathbb{R}^n$, with the group operation $\odot$ being our regular vector addition operation $(+)$ and the inverse of a vector $v \in \mathbb{R}^n$ is $(-v)$ with the 0 vector being the identity element. In the next section, we will introduce a manifold which turns into a lie group when endowed with a specific Riemannian metric. Other examples include the orthogonal group, the special orthogonal group, and the general linear group. Readers are advised to check out [36] - an introductory book about the theory of Lie Groups.

7.1.1 Lie Algebra

A lie algebra is a vector space $\mathfrak{g}$ over a field F with an operation called a Lie bracket, $[\cdot,\cdot] : \mathfrak{g}\times\mathfrak{g} \rightarrow \mathfrak{g}$. The Lie bracket satisfies the following axioms:

- It is bilinear i.e $[ax + by, z] = a[x, y] + b[y, z]$ and $[z, ax + by] = a[z, x] + b[z, y]$
- It is skew-symmetric i.e $[x, x] = 0$ which implies $[x, y] = -[y, x] \quad \forall x, y \in \mathfrak{g}$
- It satisfies the Jacobi identity $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0 \quad \forall x, y, z \in \mathfrak{g}$

7.1.2 Lie Group-Lie Algebra correspondence

The Lie Group - Lie Algebra Correspondence allows the association of a Lie Group with a Lie algebra. Essentially, the Lie algebra is the tangent space at the identity element E of the Lie Group and

the lie bracket is defined for $x, y \in T_E\mathfrak{G}$. A powerful consequence of that is that every element of the lie group $G \in \mathfrak{G}$ can be uniquely mapped to a point $g \in T_E\mathfrak{G}$ with the help of these two special mappings -

- **Exponential Map** - The exponential bijection mapping maps an element in $T_E\mathfrak{G}$ to an element in $\mathfrak{G}$ i.e exp : $T_E\mathfrak{G} \rightarrow \mathfrak{G}$
- **Logarithm Map** - The logarithm bijection mapping maps an element in $\mathfrak{G}$ to an element in $T_E\mathfrak{G}$ i.e log : $\mathfrak{G} \rightarrow T_E\mathfrak{G}$

The exponential and logarithm are inverse mappings of each other. So, for $G \in \mathfrak{G}$, $\exp(\log(G)) = G$ holds. for our proposed theory, we restrict ourselves to Lie groups where Lie algebras can be represented using a Matrix (Matrix lie groups) for clarity of representation.

7.2 Word analogy for Positive Definite Manifold with the Log-Euclidean Metric

A fairly recent discovery in differential geometry optimization is that the manifold of symmetric positive definite matrices(SPD), when endowed with the Log-Euclidean (LE) metric, is a lie group [6]. The manifold of symmetric positive definite matrices, $Sym^+_*(n)$ is the set

$$\{S \in \mathbb{R}^{n\times n} | S^\top = S, v^\top S v > 0 \quad \forall v \in \mathbb{R}^n\}$$

where n is a positive integer. $Sym^+_*(n)$ is a Lie group under the group operation $\odot$ such that

$$S_1 \odot S_2 = \exp(\log(S_1) + \log(S_2))$$

where $S_1, S_2 \in Sym^+_*(n)$, $\log(S)$ and $\exp(S)$ are the matrix exponential and matrix logarithm of a symmetric positive definite matrix S respectively. The inverse operation for the group is the matrix inverse function and thus $S_1 \odot S_1^{-1} = I$ where I is the identity matrix, which is also the Identity element of the group. We call this group $(Sym^+_*(n), \odot)$. The Lie algebra for this group is the space of all symmetric $n \times n$ matrices, $Sym(n)$ where the exponential map and logarithm map are the ones in the matrix sense. We also get the Log-Euclidean distance metric for $S_1, S_2 \in Sym^+_*(n)$

$$d(S_1, S_2) = ||\log(S_1) - \log(S_2)||_F \tag{7.1}$$

Keeping with the above ideas, the loss function for the `Word2vec` - SGNS algorithm is modified for SPD manifold with LE metric. Instead of the usual dot product metric being used inside the sigmoid function, the metric of similarity used is $\log(W_i)^\top \log(\bar{W}_j)$ where W_i is the i-th focus word SPD matrix and $\bar{W}_j$ is the j-th context word SPD matrix. The optimization is done over the matrix of SPD matrices with the loss function $f(W), W \in Sym^+_*(n)$. The gradient for each step is the Euclidean gradient since the tangent space is a vector space. The retraction operation is given by $W_i = \exp(\log(W_i) + \eta * \text{grad}_{W_i} f)$

where η is the learning rate and $\text{grad}_{W_i} f$ is the euclidean gradient of the loss function. Finally, the analogy calculation is done using $W_d = \exp(\log(W_b) - \log(W_a) + \log(W_c))$.

However, in practice, calculating the logarithm differential of a matrix is challenging and our loss function faces the same issue while calculating the gradient. Following the footsteps of [35], [73] and [54], we consider the parameterization of SPD matrices by the symmetric matrices through the matrix exponential. So, for any word matrix W_i, which is a point in the Lie group, it can be re-parameterized as $W_x = \exp(S_x), S_x \in Sym(n)$ where n is the hyperparameter, dimension size. So, the SGNS loss function will be reformulated as $g(W) = L(\exp(S)), S \in Sym(n)$ which is an optimization over symmetric matrices - an Euclidean space. Thus, computation becomes very easy and the `Word2vec` c code can be used directly with some modifications. Note that this simplification is entirely coincidental for this group. Optimizing over lie groups like special orthogonal groups in practice is non-trivial.

7.3 Skip-gram Negative sampling Algorithm generalized for Lie groups

Generalising the methods used in the previous section, we introduce the `Word2vec` SGNS algorithm for an arbitrary Lie group. The only real change happens in the loss function where the inner product between the focus and context word embedding is replaced by the dot product between the tangent space points mapping of the two lie groups word embeddings using logarithm mapping. The Riemannian gradient is exactly the same as the Euclidean gradient as the tangent space of any lie group is just another vector space. For the gradient descent update, the retraction operation depends on the space. So for an arbitrary Lie group $\mathfrak{G}$ with the group operation $\odot$, we have the following changes -

- The similarity for two words embedded in the Lie group i.e. $W_a, W_b \in \mathfrak{G}$, is calculated by

$$\text{sim}(W_a, W_b) = \log(W_a)^\top \log(W_b) \tag{7.2}$$

  Both operation $\log(W_a)$ and $\log(W_b)$ takes the embeddings into $T_{W_I}\mathfrak{G}$ which is the tangent space of the Identity element of the group, W_I. Since the tangent space is a vector space and the mapping is one-to-one, we can perform the cosine similarity operation between two vectors in a vector space leading to (7.2).

- The Skip-Gram Negative sampling loss function remains the same as it does in the traditional `Word2vec` algorithm except we have $\sigma(\text{sim}(W_a, W_b))$ instead of $W_a^\top W_b$

- The gradient for each step is also the Euclidean gradient since the tangent space is nothing but the Euclidean space. A keen observer will note that the "projection" operation is just the identity operation when comparing with the gradient calculation performed in the previous chapter.

- The retraction operation for a word embedding, W_a is $\exp(\log(W_a) + \eta \text{grad}_{W_a} f)$ for a function f defined on the Lie group

After training, word analogy between three words W_a, W_b, W_c is evaluated using $W_d = W_b \odot W_a^{-1} W_c$.

Chapter 8

Conclusions, Limitations and Future Work

In this manuscript, we proposed unsupervised text embedding models that train using Riemannian optimization tools. We show that using these models, words, sentences, and documents can be represented as matrices which can have different interpretations based on the manifold space. This allows us to exploit the geometry of the space to achieve better results in tasks like document clustering, document classification, sentiment analysis, semantic textual similarity, etc. We also show that one such matrix interpretation allows us to retrieve multiple senses of the word using a relatively simple matrix function like SVD or eigen decomposition, without needing to use an external sense inventory. Lie group-Lie algebra theory allowed us to extend the `Word2vec` SGNS algorithm to Lie groups and provide a new expression for word analogy evaluation suggesting that such a linear substructure is not limited to only Euclidean space.

8.1 Limitations

The biggest limitation of most of these matrix based models is efficiency. Unsupervised global word embeddings models should be easily scalable for big corpus and train rapidly enough in relatively low dimension so that they can be put as input to larger language models. In practice, it is very hard to have implementation tools that train as efficiently on large set of matrices as it would do for large set of vectors. Moreover, training on manifolds such as symmetric positive definite matrices manifold requires using matrix logarithm, matrix exponential of square matrices which are very inefficient when the size of the matrix is large. The absence of proper closed form solution for matrix differentials of logarithm and exponential mapping makes it hard to train on lie-groups as well. Most of these ML models also are not suitable for taking inputs which are encoded as points in an arbitrary manifold. Contextualised embedding models like BERT are built only for vector embeddings and thus severely limit the practical usage of these matrix manifold embeddings.

8.2 Future Work

A really interesting idea would be to provide Information geometric explanations of global embeddings. Information geometry is a rapidly growing field that uses statistical manifolds as the ambient space. The simplest example of that are the Gaussian embeddings. However, the `Word2Gauss` makes a lot of assumptions and puts a lot of constraints on the covariance matrices and consequently limiting our understanding of such models. Recently, the theory of optimal transport has been gaining traction in the field of ML. Optimal transport takes ideas from Information Geometry and provides pretty neat solutions for converting one probability distribution to another. It would be very interesting to see how Optimal transport theory can be used to explain the theory behind SGNS-algorithm or provide a better understanding of the word analogy evaluation issue. It would also be very practical to build contextualised embedding models that can take as inputs arbitrary manifold embeddings. Finally, complex embeddings are also very sparsely studied. Word semantic information could be modeled better in a space with a complex structure. So, it might also be worth studying models that train in complex Riemannian manifolds.

Research Papers Based on the Thesis Work

Conference Papers

1. **Souvik Banerjee**, Bamdev Mishra, Pratik Jawanpuria and Manish Shrivastava. Generalised Spherical Text Embedding. *Proceedings of the 19th International Conference on Natural Language Processing, pages 80 - 85* https://aclanthology.org/2022.icon-main.11/

Bibliography

[1] 2018. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *J. Artif. Int. Res.*, **63**(1), 743–788.

[2] Absil, P.-A., Mahony, R., & Sepulchre, R. 2008. *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press.

[3] Allen, Carl, & Hospedales, Timothy. 2019. Analogies Explained: Towards Understanding Word Embeddings. *Pages 223–231 of:* Chaudhuri, Kamalika, & Salakhutdinov, Ruslan (eds), *Proceedings of the 36th International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 97. PMLR.

[4] Arora, Sanjeev, Li, Yuanzhi, Liang, Yingyu, Ma, Tengyu, & Risteski, Andrej. 2016. A Latent Variable Model Approach to PMI-based Word Embeddings. *Transactions of the Association for Computational Linguistics*, **4**, 385–399.

[5] Arora, Sanjeev, Liang, Yingyu, & Ma, Tengyu. 2019 (Jan.). A simple but tough-to-beat baseline for sentence embeddings. 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.

[6] Arsigny, Vincent, Fillard, Pierre, Pennec, Xavier, & Ayache, Nicholas. 2007. Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices. *SIAM J. Matrix Anal. Appl.*, **29**, 328–347.

[7] Athiwaratkun, Ben, & Wilson, Andrew. 2017. Multimodal Word Distributions. *Pages 1645–1656 of: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics.

[8] Banerjee, Arindam, Dhillon, Inderjit S., Ghosh, Joydeep, & Sra, Suvrit. 2005. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions. *J. Mach. Learn. Res.*, **6**, 1345–1382.

[9] Baroni, Marco, & Lenci, Alessandro. 2010. Distributional Memory: A General Framework for Corpus-Based Semantics. *Computational Linguistics*, **36**(4), 673–721.

[10] Bartunov, Sergey, Kondrashkin, Dmitry, Osokin, Anton, & Vetrov, Dmitry. 2016 (May). Breaking Sticks and Ambiguities with Adaptive Skip-gram. *Pages 130–138 of: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[11] Batmanghelich, Kayhan, Saeedi, Ardavan, Narasimhan, Karthik, & Gershman, Sam. 2016. Nonparametric Spherical Topic Modeling with Word Embeddings. *Pages 537–542 of: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics.

[12] Batzies, E., Hüper, K., Machado, L., Leite, F. Silva, Batzies, E., Huper, K., Machado, L., & Leite, F.S. 2015. Geometric mean and geodesic regression on Grassmannians. *Linear Algebra and Its Applications*, **466**(Complete), 83–101.

[13] Bhat, Siddharth, Debnath, Alok, Banerjee, Souvik, & Shrivastava, Manish. 2020. Word Embeddings as Tuples of Feature Probabilities. *Pages 24–33 of: Proceedings of the 5th Workshop on Representation Learning for NLP*. Online: Association for Computational Linguistics.

[14] Bojanowski, Piotr, Grave, Edouard, Joulin, Armand, & Mikolov, Tomas. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, **5**, 135–146.

[15] Brown, Tom, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared D, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, Agarwal, Sandhini, Herbert-Voss, Ariel, Krueger, Gretchen, Henighan, Tom, Child, Rewon, Ramesh, Aditya, Ziegler, Daniel, Wu, Jeffrey, Winter, Clemens, Hesse, Chris, Chen, Mark, Sigler, Eric, Litwin, Mateusz, Gray, Scott, Chess, Benjamin, Clark, Jack, Berner, Christopher, McCandlish, Sam, Radford, Alec, Sutskever, Ilya, & Amodei, Dario. 2020. Language Models are Few-Shot Learners. *Pages 1877–1901 of:* Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., & Lin, H. (eds), *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc.

[16] Bruni, Elia, Boleda, Gemma, Baroni, Marco, & Tran, Nam-Khanh. 2012. Distributional Semantics in Technicolor. *Pages 136–145 of: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: Association for Computational Linguistics.

[17] Bruni, Elia, Tran, N., & Baroni, Marco. 2014. Multimodal Distributional Semantics. *J. Artif. Intell. Res.*, **49**, 1–47.

[18] Cer, Daniel, Diab, Mona, Agirre, Eneko, Lopez-Gazpio, Inigo, & Specia, Lucia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. *In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics.

[19] Conneau, Alexis, Kiela, Douwe, Schwenk, Holger, Barrault, Loïc, & Bordes, Antoine. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *Pages 670–680 of: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics.

[20] Deerwester, Scott C., Dumais, Susan T., Landauer, Thomas K., Furnas, George W., & Harshman, Richard A. 1990. Indexing by Latent Semantic Analysis. *J. Am. Soc. Inf. Sci.*, **41**, 391–407.

[21] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Pages 4171–4186 of: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics.

[22] Edelman, A., Arias, T., & Smith, S. 1998. The Geometry of Algorithms with Orthogonality Constraints. *SIAM J. Matrix Anal. Appl.*, **20**, 303–353.

[23] Ethayarajh, Kawin, Duvenaud, David, & Hirst, Graeme. 2019. Towards Understanding Linear Word Analogies. *Pages 3253–3262 of: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics.

[24] Fefferman, Charles, Mitter, Sanjoy, & Narayanan, Hariharan. 2016. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, **29**(4), 983–1049.

[25] Finkelstein, Lev, Gabrilovich, Evgeniy, Matias, Yossi, Rivlin, Ehud, Solan, Zach, Wolfman, Gadi, & Ruppin, Eytan. 2001. Placing Search in Context: The Concept Revisited. *Page 406–414 of: Proceedings of the 10th International Conference on World Wide Web*. WWW '01. New York, NY, USA: Association for Computing Machinery.

[26] Firth, J. R. 1957. A synopsis of linguistic theory 1930-55. **1952-59**, 1–32.

[27] Gallivan, K.A., Srivastava, A., Liu, Xiuwen, & Van Dooren, P. 2003. Efficient algorithms for inferences on Grassmann manifolds. *Pages 315–318 of: IEEE Workshop on Statistical Signal Processing, 2003*.

[28] Gardenfors, Peter. 2004. Conceptual Spaces as a Framework for Knowledge Representation. *Mind and Matter*, **2**(2), 9–27.

[29] Gittens, Alex, Achlioptas, Dimitris, & Mahoney, Michael W. 2017. Skip-Gram - Zipf + Uniform = Vector Additivity. *In: Annual Meeting of the Association for Computational Linguistics*.

[30] Goh, Alvina, & Vidal, Rene. 2008. Clustering and dimensionality reduction on Riemannian manifolds. *Pages 1–7 of: 2008 IEEE Conference on Computer Vision and Pattern Recognition*.

[31] Goldberg, Yoav, & Levy, Omer. 2014. *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*.

[32] Gong, Boqing, Shi, Yuan, Sha, Fei, & Grauman, Kristen. 2012. Geodesic flow kernel for unsupervised domain adaptation. *Pages 2066–2073 of: 2012 IEEE Conference on Computer Vision and Pattern Recognition*.

[33] Gopal, Siddharth, & Yang, Yiming. 2014. Von Mises-Fisher Clustering Models. *Pages 154–162 of:* Xing, Eric P., & Jebara, Tony (eds), *Proceedings of the 31st International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 32, no. 1. Bejing, China: PMLR.

[34] Gopalan, Raghuraman, Li, Ruonan, & Chellappa, Rama. 2011. Domain adaptation for object recognition: An unsupervised approach. *Pages 999–1006 of: 2011 International Conference on Computer Vision*.

[35] Ha Quang, Minh, San Biagio, Marco, & Murino, Vittorio. 2014. Log-Hilbert-Schmidt metric between positive definite operators on Hilbert spaces. *In:* Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., & Weinberger, K.Q. (eds), *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc.

[36] Hall, Brian C. 2004. Lie Groups, Lie Algebras, and Representations: An Elementary Introduction.

[37] Helmke, Uwe, & Moore, John B. 2012. *Optimization and dynamical systems*. Springer Science & Business Media.

[38] Helmke, Uwe, Hüper, Knut, & Trumpf, Jochen. 2007. *Newton's method on Grassmann manifolds*.

[39] Hill, Felix, Reichart, Roi, & Korhonen, Anna. 2015. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics*, **41**(4), 665–695.

[40] Huang, Zhiwu, Wu, Jiqing, & Van Gool, Luc. 2018. Building Deep Networks on Grassmann Manifolds. *In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. AAAI Press.

[41] Jain, Shobhit, Bodapati, Sravan Babu, Nallapati, Ramesh, & Anandkumar, Anima. 2019. Multi Sense Embeddings from Topic Models. *Pages 34–41 of: Proceedings of the 3rd International Conference on Natural Language and Speech Processing*. Trento, Italy: Association for Computational Linguistics.

[42] Journée, M., Bach, F., Absil, P.-A., & Sepulchre, R. 2010. Low-Rank Optimization on the Cone of Positive Semidefinite Matrices. *SIAM Journal on Optimization*, **20**(5), 2327–2351.

[43] Kiros, Ryan, Zhu, Yukun, Salakhutdinov, Ruslan, Zemel, Richard S., Torralba, Antonio, Urtasun, Raquel, & Fidler, Sanja. 2015. Skip-Thought Vectors. *Page 3294–3302 of: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Cambridge, MA, USA: MIT Press.

[44] Landauer, Thomas K., & Dumais, Susan T. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, **104**, 211–240.

[45] Le, Quoc, & Mikolov, Tomas. 2014. Distributed Representations of Sentences and Documents. *Page II–1188–II–1196 of: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. JMLR.org.

[46] Levy, Omer, & Goldberg, Yoav. 2014. Neural Word Embedding as Implicit Matrix Factorization. *In:* Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., & Weinberger, K.Q. (eds), *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc.

[47] Levy, Omer, Goldberg, Yoav, & Dagan, Ido. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, **3**, 211–225.

[48] Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, & Stoyanov, Veselin. 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*.

[49] Mahadevan, Sridhar, & Chandar, Sarath. 2015. *Reasoning about Linguistic Regularities in Word Embeddings using Matrix Manifolds*.

[50] Malagò, Luigi, Montrucchio, Luigi, & Pistone, Giovanni. 2018. *Wasserstein Riemannian Geometry of Positive Definite Matrices*.

[51] Manning, Christopher D., Raghavan, Prabhakar, & Schütze, Hinrich. 2008. *Introduction to Information Retrieval*. USA: Cambridge University Press.

[52] Meng, Yu, Huang, Jiaxin, Wang, Guangyuan, Zhang, Chao, Zhuang, Honglei, Kaplan, Lance, & Han, Jiawei. 2019. Spherical Text Embedding. *In: Advances in neural information processing systems*.

[53] Meng, Yu, Zhang, Yunyi, Huang, Jiaxin, Zhang, Yu, Zhang, Chao, & Han, Jiawei. 2020. Hierarchical Topic Mining via Joint Spherical Tree and Text Embedding. KDD '20. New York, NY, USA: Association for Computing Machinery.

[54] Meyer, Gilles, Bonnabel, Silvere, & Sepulchre, Rodolphe. 2011. *Regression on fixed-rank positive semidefinite matrices: a Riemannian approach*.

[55] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, & Dean, Jeff. 2013a. Distributed representations of words and phrases and their compositionality. *Pages 3111–3119 of: Advances in neural information processing systems*.

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

[57] Mimno, David, & Thompson, Laure. 2017. The strange geometry of skip-gram with negative sampling. *Pages 2873–2878 of: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics.

[58] Moghadasi, Mahdi Naser, & Zhuang, Yu. 2020. Sent2Vec: A New Sentence Embedding Representation With Sentimental Semantic. *Pages 4672–4680 of: 2020 IEEE International Conference on Big Data (Big Data)*.

[59] Mu, Jiaqi, Bhat, Suma, & Viswanath, Pramod. 2017. Representing Sentences as Low-Rank Subspaces. *Pages 629–634 of: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics.

[60] Muzellec, Boris, & Cuturi, Marco. 2018. Generalizing Point Embeddings Using the Wasserstein Space of Elliptical Distributions. *Page 10258–10269 of: Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Red Hook, NY, USA: Curran Associates Inc.

[61] Neelakantan, Arvind, Shankar, Jeevan, Passos, Alexandre, & McCallum, Andrew. 2014. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. *Pages 1059–1069 of: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics.

[62] Nickel, Maximillian, & Kiela, Douwe. 2017. Poincaré Embeddings for Learning Hierarchical Representations. *In:* Guyon, I., Luxburg, U. Von, Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. (eds), *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc.

[63] Pennington, Jeffrey, Socher, Richard, & Manning, Christopher. 2014. Glove: Global vectors for word representation. *Pages 1532–1543 of: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

[64] Pilehvar, Mohammad Taher, & Collier, Nigel. 2016. De-Conflated Semantic Representations. *Pages 1680–1690 of: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics.

[65] Qian, Chen, Feng, Fuli, Wen, Lijie, & Chua, Tat-Seng. 2021. Conceptualized and Contextualized Gaussian Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**(15), 13683–13691.

[66] Reisinger, Joseph, Waters, Austin, Silverthorn, Bryan, & Mooney, Raymond J. 2010. Spherical Topic Models. ICML'10. Madison, WI, USA: Omnipress.

[67] Salton, G., Wong, A., & Yang, C. S. 1975. A Vector Space Model for Automatic Indexing. *Commun. ACM*, **18**(11), 613–620.

[68] Schütze, Hinrich. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, **24**(1), 97–123.

[69] Smith, Steven Thomas. 2014. Optimization Techniques on Riemannian Manifolds.

[70] Srivastava, Anuj, & Liu, Xiuwen. 2005. Tools for application-driven linear dimension reduction. *Neurocomputing*, **67**, 136–160. Geometrical Methods in Neural Networks and Learning.

[71] Steinley, Douglas L. 2004. Properties of the Hubert-Arabie adjusted Rand index. *Psychological methods*, **9 3**, 386–96.

[72] Tifrea*, A., Becigneul*, G., & Ganea*, O.-E. 2019 (May). Poincaré GloVe: Hyperbolic Word Embeddings. *In: 7th International Conference on Learning Representations (ICLR)*. *equal contribution.

[73] Tsuda, Koji, Rätsch, Gunnar, & Warmuth, Manfred K. 2005. Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection. *Journal of Machine Learning Research*, **6**(34), 995–1018.

[74] Tversky, Amos. 1977. Features of Similarity. *Psychological Review*, **84**, 327–352.

[75] Vilnis, Luke, & McCallum, Andrew. 2015. Word Representations via Gaussian Embedding. *In:* Bengio, Yoshua, & LeCun, Yann (eds), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[76] Weston, Jason, Bengio, Samy, & Usunier, Nicolas. 2011. Wsabie: Scaling Up To Large Vocabulary Image Annotation. *In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*.

[77] Yaghoobzadeh, Yadollah, & Schütze, Hinrich. 2016. Intrinsic Subspace Evaluation of Word Embedding Representations. *Pages 236–246 of: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics.