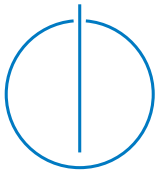
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Effective and Scalable Sentence Representation through Dynamic Grassmannian Ellipsoids

Philipp Charles Dowling



FAKULTÄT FÜR INFORMATIK

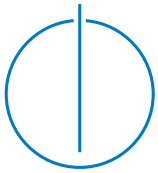
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Effective and Scalable Sentence Representation through Dynamic Grassmannian Ellipsoids

Effektive und skalierbare Darstellung von Sätzen durch Ellipsoide in dynamischen Graßmann-Manningfaltigkeiten

Author:	Philipp Charles Dowling
Supervisor:	Prof. Stephan Günnemann
Advisor:	Dr. Pablo Paredes
Submission Date:	April 16, 2018



I confirm that this Master's thesis is my own work and I have documented all sources and material used.

Munich, April 16, 2018 Philipp Charles Dowling

Acknowledgments

I would like to thank Professor Stephan Günnemann, my supervisor, and Dr. Pablo Paredes of Stanford University, my advisor, for their time, insight, and guidance they have offered me in preparing and developing this thesis.

I would also like to extend special thanks to James Landay for hosting me at the Stanford HCI lab for some of the work that went into this thesis, and to Andrea Kuduk for her help and support in organizing my stay there. I also thank Guillaume Lachaud, Yeshuang Zhu, Greg Valiant, Pascal Odek, Honghao Wei, the IxD research group, and everyone else that offered me feedback, input, or other support during my time at the HCI lab, as well as Prof. Coye Cheshire, Vasilis Oikonomou, and Biye Jiang at UC Berkeley.

I thank my family for all of the love, support, and guidance they've given me over the years; to my dad and to my grandfather Paul also for helping proof-read this thesis. Finally, I thank Annie Felix Groth for all the help, love, and support that she has given me along this entire process, as well as beyond.

I'm grateful to all of you for making this work possible.

Abstract

Vector spaces arising from distributional word vector methods such as `word2vec`, `GloVe`, or `FastText` exhibit high degrees of structural regularities in regards to semantic and syntactic relationships between words. This surprising characteristic promises to be useful in developing novel methods and algorithms; nevertheless, the phenomenon is still underexplored.

Subspace-based approaches constitute one way of investigating and leveraging such structure. Under a recently introduced Grassmannian view of sentence semantics, sentences are represented as low-rank subspaces that are spanned by the word vectors of the contained words [MBV17]. In this work, we adopt and extend this view in multiple ways, and investigate further applications of Grassmannian sentence semantics.

We refine the subspace sentence embedding model introduced by Mu et al. to introduce the Dynamic Grassmannian Ellipsoid (DGE) model, achieving strong performance on textual similarity benchmarks. Through a combination of hyperparameter changes and improvements to the method, our model achieves an average performance improvement of 2.4% over Mu et al.'s model.

We also investigate the approximate nearest subspace search (ANSS) problem on the textual domain, extending a recently introduced technique by Iwamura et al., making a range of improvements and adapting the approach to the domain of different-dimensional subspaces. We also introduce `Schuhlöffel`, a method for constrained optimization of subspace basis matrices on the Stiefel manifold. Our combined alterations yield an ANSS method that achieves 90% top-10 accuracy at a 6-fold performance gain compared to brute force search.

Finally, we introduce Grassmannian Embedded Topics (GET), a method that is closely related to DGE, for estimating interpretable topic distributions of texts. We provide a qualitative empirical exploration of interpretable and interactive text semantics through a set of interaction examples.

We find that Grassmannian techniques can be effectively and scalably applied in the textual domain, and that they are particularly well-suited for the evaluation of semantic similarity of sentences. We also highlight the fact that these techniques bring forth structure in word-vector spaces in easily interpretable ways, making them particularly suitable for human-facing applications. Overall, our research casts Grassmannian methods as a valuable tool in the study of distributional semantics.

Contents

1. Introduction

The volume and availability of unstructured textual data on the web is continuing to grow. This has created significant challenges in enabling comprehensive access to the knowledge contained within such data. At the same time, it has also led to opportunities for specialized and innovative uses of this data, which has, in turn, lead to an increased need for sophisticated means of structuring, searching, and exploring textual datasets.

The motivation for this work is firmly rooted within this trend, in that its aim is to foster better exploration and understanding of text corpora. Employing the view that words are fundamental units of semantics, sentences may be seen as the unit of semantics at which concepts, thought, ideas, and questions are expressed. This view suggests that sentence-level approaches are of particular value in the development of systems that go beyond conventional search (i.e. locating information), and focus on more encompassing use-cases such as summarization, exploration, insight discovery, and other forms of qualitative of structural analysis.

This work serves as an exploration of a particular approach to sentence representation, which follows a two-fold goal: To enable *objectively effective* (and efficient) representation and comparison of sentences, and to enable *subjectively interpretable* downstream applications relating to interactive exploration of text collections.

Word-level semantics have received much attention in recent years, with popular approaches like `word2vec` [Mik+13], `GloVe` [PSM14], and `FastText` [Boj+17] being in ubiquitous use across many application areas. It has been observed that the word embeddings produced by these techniques tend to exhibit certain structural regularities in regards to, for instance, syntactic, grammatical, and semantic analogies [MYZ13], as well as word-sense phenomena like polysemy [Aro]. These results suggest that word vector spaces possess a high degree of internal structure; however, we find that there has been a lack of research into leveraging this structure more fully.

Under the Grassmannian view of word vector spaces, this structure may be incorporated into models and techniques through the use of (low-rank) linear subspaces of the full vector space. Instead of viewing sentences (or documents) as vectors—the same representation shared by words—they are viewed as subspaces defined by the word vectors they contain [MBV17].

We adopt and extend this view in multiple ways. In Part I, we begin by refining the

subspace sentence embedding model introduced by Mu et al. and ultimately introduce the Dynamic Grassmannian Ellipsoid (DGE) model, which achieves strong performance on textual similarity benchmarks. Subsequently, in Part II we investigate the scalability of Grassmannian text representations by considering the approximate nearest subspace search problem, applying and extending a recently introduced technique by Iwamura et al. to the textual domain. Finally, Part III offers a qualitative exploration and discussion of further practical applications of DGE in the textual domain, putting focus on its applicability to interactive search and exploration problems.

We find that Grassmannian techniques can be effectively and scalably applied in the textual domain, and that they are particularly well-suited for the evaluation of semantic similarity of sentences. We also highlight the fact that these techniques bring forth structure in word-vector spaces in easily interpretable ways, making them particularly suitable for human-facing applications.

2. Fundamentals

2.1. Word vectors

The basis of this work is rooted in semantic word vector models, which in recent years have established themselves as a core ingredient in approaches to many problems within NLP and text mining. The fundamental idea behind word vectors (also called word embeddings) is to learn a mapping $\mathrm{v} : [D] \rightarrow \mathbb{R}^d$ that each word in a vocabulary of size D to a real-valued dense vector of dimensionality d, which captures the particular word's semantics.[1] This representation allows for the comparison of words through a variety of vector similarity measures, and also enables the usage of word semantics as rich, comparable features in downstream applications such as clustering or classification.

The key commonality of these approaches is that they view "semantics" as an emergent phenomenon of word usage and co-occurrence, as suggested by the distributional hypothesis of Harris [Har54]. Word embedding methods are commonly "trained" on large, unlabeled corpora of natural language text, and the "learned" vectors emerge from and reflect co-occurrence patterns in the usage of words. The unsupervised nature of this training makes them attractive, as large text corpora are easily available; however, the type of corpus used will strongly influence the applicability of a resulting model to a given domain. For instance, a word vector model trained on news data may not be useful on biomedical texts.

Early approaches to the training of such models included dimensionality reduction by means of factorization of word-cooccurrence matrices [Sch92], while today neural-network based approaches using backpropagation are more common. One of the popular word embedding models today is the `skip-gram` approach in `word2vec` [Mik+13], which combines a neural network's ability to capture non-linearities in data with highly scalable training by avoiding full softmax computations, and can thus be efficiently trained on very large datasets. Mikolov et al. also showed that the resulting vectors, in addition to capturing word similarity, capture linear structures that may correspond to syntactic and semantic relationships between words. These regularities in the space's structure allow for the expression of simple analogies in algebraic form, such as for

[1] For convenience, we will use the notation $\mathrm{v}(\text{"hello"})$ to refer to the vector for the word "hello"—in practice, every word is uniquely mapped to an integer ID.

instance a Country-Capital relationship: $\mathrm{v}(\text{"Germany"}) - \mathrm{v}(\text{"Berlin"}) \approx \mathrm{v}(\text{"France"}) - \mathrm{v}(\text{"Paris"})$ [MYZ13]. Such structural properties of word vector models have received much attention in recent literature [PSM14] [LG14] [Lev+15] [Boj+17] [Aro].

In our work, we focus primarily on two more recent word vector models: `GloVe` and `FastText`. We will give a brief overview of their key features.

2.1.1. GloVe

The `GloVe` model by Pennington et al. [PSM14] represents an iteration of the ideas presented by Mikolov et al. in `word2vec`. They combine strong properties of (global) matrix factorization approaches with (local) context-window modeling methods, aiming to explicitly model word semantic using global corpus statistics while still capturing the subtleties of sub-structure contained in "fine-grained" word usage. It could be argued that `GloVe` is more theoretically guided than the `word2vec` family of models. This approach ultimately outperforms the `skip-gram` model on a variety of tasks, including analogy reasoning and named-entity recognition (NER).

`GloVe` vectors are widely used in practice today, specifically pre-trained models trained on Wikipedia, Common Crawl and Twitter datasets available at `https://nlp.stanford.edu/projects/glove/`. In our work, we utilize the training code available at the same address.

2.1.2. FastText

Bojanowski et al.'s `FastText` model [Boj+17] is based on the `skip-gram` approach, and also uses negative sampling to avoid the computation of the full softmax for efficient training. However, rather than modeling words strictly as discrete tokens, embeddings are learned for words and sub-word features (i.e. the set of character n-grams contained in each word), and a words final representation is the sum of these embeddings. Leveraging these sub-word features not only adds morphological information to word representations, it also allows for "sharing" semantic units across words (e.g. "*fire*hose" and "*fire*truck") and thus also the estimation of word vectors for less common / out-of-vocabulary (OOV) terms ("brush*fire*"). While Bojanowski et al. did not initially evaluate the model against `GloVe`'s performance, more recent work by Mikolov et al. establishes `FastText` (in conjunction with other optimizations) as one of the most highly-performant word vector models on a variety of tasks, including analogy reasoning, rare words and reading comprehension [Mik+17].

In this work, we use the `FastText` implementation available at `https://github.com/facebookresearch/fastText`.

2.2. The Grassmannian

The Grassmannian $\mathrm{Gr}(k, \mathbb{V})$ is the set of all k-dimensional linear subspaces of a d-dimensional space $\mathbb{V}$. Technically, $\mathbb{V}$ could be any vector space; however, in this work we assume $\mathbb{V} = \mathbb{R}^d$, and use the shorthand $\mathrm{Gr}(k, d) = \mathrm{Gr}(k, \mathbb{R}^d)$. Note that the Grassmannian is itself a space (more specifically, it is a Riemannian manifold), thus it is essentially a space of subspaces. As a simple example, consider $\mathrm{Gr}(k, 3)$: at $k = 1$, this represents the space of all lines that pass through the origin. Similarly, for $k = 2$, it is the space of all planes that contain the origin.

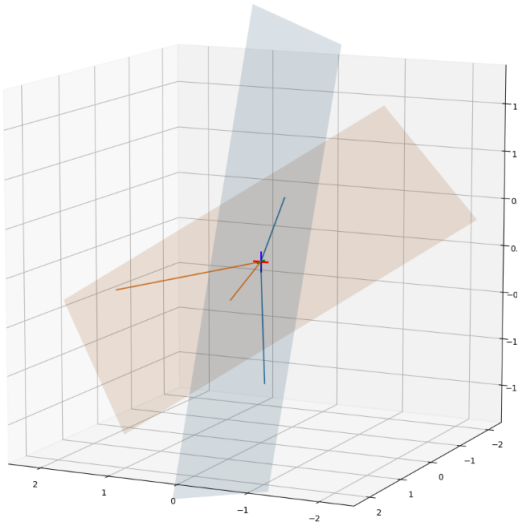


Figure 2.1.: Visualization of random elements of $\mathrm{Gr}(1, 3)$ and $\mathrm{Gr}(2, 3)$, i.e. lines and planes passing through the origin of $\mathbb{R}^3$.

Note that there are potentially an infinite number of representations for each element of $\mathrm{Gr}(k, d)$, since there are e.g. infinitely many sets of vectors that span a given subspace. In this work, we will commonly represent a subspace in terms of an orthogonal basis matrix that spans it. Accordingly, in place of $\mathbb{V}_A \in \mathrm{Gr}(k, d)$ we write $A \in \mathrm{Gr}(k, d)$ (where $A \in \mathbb{R}^{k \times d}$ is a row-wise orthonormal matrix). This representation is not unique, but this does not matter for practical purposes.

As a rather fundamental geometric object, the Grassmannian is used in a number of areas within applied mathematics, science, engineering and computer science. It is used for a variety of different problems in image processing, such as face recognition [Hu14] and background separation in video [HBS12]. It has also been applied to data visualization [Asi85] and finds use in high-energy Physics [Ark+14] and quantum computing [Fuj02].

2.3. Grassmannian Kernels and Similarity

It is possible to define a variety of kernels and distance functions on the Grassmannian. One of the most commonly used is the geodesic distance, which can be thought of as a curve of shortest distance (essentially a "straight-line" distance on a globe). This notion of distance arises naturally from Grassmannians being Riemannian manifolds, as it represents the length of the shortest curve on the manifold connecting two subspaces [Har+14]. It is defined as follows [AMS04]:

$$\begin{aligned}\text{dist}_{\text{Geo}}(A, B) &= \sqrt{\theta_1^2 + ... + \theta_k^2} \\ &= \left(\sum_{i=1}^{k} \theta_i^2\right)^{\frac{1}{2}},\end{aligned}$$

where A and B are some orthonormal matrices spanning $\mathbb{V}_A, \mathbb{V}_B \in \text{Gr}(k, d)$, and θ_i is the i'th largest principal angle between $\mathbb{V}_A$ and $\mathbb{V}_B$ (θ is sorted in decreasing order). The principal angles θ_i between subspaces can be computed using the singular value decomposition (SVD) [GV12]:

$$\begin{aligned}\text{SVD}(A^T B) &= USV^T \\ \cos(\theta_i) &= \text{diag}(S)_i.\end{aligned}$$

The geodesic distance requires computing the principal angles, and may thus be inconvenient or costly to compute. There are a number of other distances and metrics defined on the Grassmannian, some of which can be computed directly from orthonormal bases A and B without the singular value decomposition. One such example, which we will refer to in later sections, is the projection metric [HL08]:

$$\begin{aligned}\text{dist}_{\text{Proj}}(A, B) &= \left(k - \sum_{i=1}^{k} \cos^2(\theta_i)\right)^{\frac{1}{2}} \\ &= \left(k - \left\|AB^T\right\|_F^2\right)^{\frac{1}{2}},\end{aligned}$$

where $\|\cdot\|_F$ is the Frobenius norm. This metric also induces a simple kernel, which represents subspace similarity (as opposed to distance) [HL08]:

$$\text{sim}_{\text{Proj}}(A, B) = \left\|AB^T\right\|_F^2$$

2.3.1. Similarities between subspaces of different dimensionalities

So far, our distance and similarity functions assume subspaces A and B to have the same dimensionality k. Note that in this context, dimensionality refers to the number of "axes" of each subspace, not the dimensionality of the ambient space they are embedded in (d). We can adapt our measures using results by Ye & Lim [YL16], who show that valid comparison measures for subspaces $A \in \mathrm{Gr}(k,d), B \in \mathrm{Gr}(l,d), k \leq l \leq d$ can be derived by utilizing "the Schubert variety of k-planes contained in B" $\Omega_{-}(B)$, that is:

$$\Omega_{-}(B) := \{X \in \mathrm{Gr}(k,d) : X \subseteq B\},$$

which is essentially the "sub-Grassmannian" of B constrained to k dimensions. The actual distance is then computed as the distance between A and the closest element of $\Omega_{-}(B)$:

$$\begin{aligned}\mathrm{dist}_{\mathrm{Geo}}(A,B) &= \min\{\mathrm{dist}_{Geo}(A,X) : X \in \Omega_{-}(B)\} \\ &= \left(\sum_{i=1}^{\min\{k,l\}} \theta_i^2\right)^{\frac{1}{2}}.\end{aligned}$$

This holds generally, also for cases where $l \leq k \leq d$. As it turns out, many distance and similarity measures can be adapted by simply using $\min\{k,l\}$ in place of k. Intuitively, the distance or similarity being measured is that of the lower dimensional subspace, say A, and the closest subspace of equivalent dimension that is contained within B. We can adapt the projection metric as follows [YL16][2]:

$$\begin{aligned}\mathrm{dist}_{\mathrm{Proj}}(A,B) &= \left(\min\{k,l\} - \sum_{i=1}^{\min\{k,l\}} \cos^2\theta_i\right)^{\frac{1}{2}} \\ &= \left(\min\{k,l\} - \|AB^T\|_F^2\right)^{\frac{1}{2}}.\end{aligned}$$

As the projection *kernel* is not an expression over k, it stays unchanged for subspaces of different dimensionalities.

[2] Ye & Lim list this as the "Chordal" metric, and define $\sin\theta_k$ as the "Projection" distance. However, they cite Edelman et al. (1998) [EA98], where an equivalent formulation of our definition of the projection metric is listed as the "Projection F-norm", and Ye & Lim's definition is listed as the "Projection 2-norm". Confusingly, Edelman et al. also list a "Chordal" 2-norm and Frobenius-norm, which are again subtly different. Both papers also express $\mathrm{dist}_{\mathrm{Proj}}$ with a different (but equivalent for $k = l$) formula in terms of the basis matrices, $\frac{1}{\sqrt{2}}\|A^TA - B^TB\|_F$.

Part I.

Dynamic Grassmannian Ellipsoids

3. Introduction

Word vector models are useful in determining semantic features and similarity on the word-level, so a natural extension of the concept is to try and model sentences (or paragraphs) using embeddings. Rather than evaluating word similarity, such models may be used for the evaluation of sentence-level similarity, or as the basis for document classification tasks such as sentiment analysis. Semantic representations of sentences are useful in a variety of application areas, such as information retrieval [Pal+16], machine translation evaluation [Guz15], conversational agents [Yan+16], and qualitative data analysis [Par+17].

Much of the recent literature in this area has focused on approaches based on deep neural networks, in particular recurrent neural networks (RNNs) [Kir+15] [HCK16] and in some cases convolutional neural networks (CNNs) [Gan+17]. However, there has also been a resurgence of simpler methods that are comparatively "shallow", but based on well-established and refined methods from word vector models. For example, Wieting et al. [Wie+15] show that fine-tuned word vector averaging represents a strong sentence embedding method for paraphrase identification. In a similar vein, Arora et al. [ALM17] show that a combination of weighted word vector averages and the (global) removal of the first principal component of all sentences is a very performant sentence embedding method. Both of these approaches tend to outperform even sophisticated neural network based approaches. This suggests that the use of word vectors and their properties warrant further exploration in the context of sentence embedding and text representation.

We introduce Dynamic Grassmannian Ellipsoids (DGE), a model for sentence embedding that is based on work by Mu et al. [MBV17]. The basic insight behind the model of Mu et al. is that word vectors in a sentence tend to cluster around a low-rank subspace, that is, it is possible to represent a sentence with e.g. >10 words with only around 4 orthonormal basis vectors, with little loss of information. We extend this model in three main ways:

1. We utilize `FastText` and weighted PCA for more robust representation learning.
2. We allow varying subspaces sizes to dynamically capture the energy in each sentence.

3. Sentences are represented as ellipsoids on elements of the Grassmannian to more accurately capture the energy distribution within each sentences.

The DGE model can be understood as a fairly direct representation of the distribution that word vectors follow within sentences after unsupervised training. We evaluate this model on the Semantic Textual Similarity (STS) tasks of the SemEval evaluation series from 2012 to 2016. We find that each of the changes listed above improves overall model performance. Compared to our own baseline implementation of the model Mu et al. describe, these changes yield an average improvement of 14.08% on these tasks, and in comparison to the results reported by Mu et al. themselves, we achieve a 2.44% average improvement. DGE also outperforms the approach introduced by Arora et al. [ALM17], using an underlying vector model that was trained on significantly less data than theirs.

Further contributions presented in this work are as follows:

1. We introduce a generalized version of the subspace cosine similarity measure used by Mu et al. that allows for the comparison of subspaces with varying dimensionalities (and show that this similarity measure is a kernel on $\mathrm{Gr}(k, d)$).

2. We further extend this similarity measure to account for basis vector magnitude, rendering it a measure of comparison for ellipsoids as opposed to just subspaces.

3. We find that `FastText` appears to be more effective than `GloVe` for sentence representation for semantic similarity tasks.

We also provide a Python implementation of DGE, at `https://github.com/phdowling/dynamic-grassmannian-ellipsoids`.

4. Related Work

We will first examine methods for computing sentence embeddings, where we distinguish between *deep* and *shallow* neural network approaches based on word vector models. This is done primarily to give an overview of what approaches to sentence embedding exist, but also in anticipation of evaluating of our own approach. The distinction between deep and shallow may be a bit uncertain overall, but in this context we assume that deep approaches are multi-layered neural networks that model word order such as recurrent or recursive neural networks, while shallow networks employ simpler architectures, usually some combination of additive approaches with a non-linear activation function, and no capturing of word order.

Subsequently, we look at algorithms that apply Grassmannian approaches or other subspace models to word embeddings. This work is relevant mainly in describing and/or providing motivation for the use of subspaces in order to model text, and to give a glimpse into what problems such approaches can be applied to.

4.1. Sentence embedding methods

While we evaluate our own work using the STS tasks from SemEval, which include supervision data, we will mainly focus this overview on unsupervised methods that result in general-purpose sentence embeddings, with the exception of some models that trained for particular specialized tasks.

Early work on sentence "embeddings" can be attributed to the field of information retrieval, where bag-of-words, TF–IDF weighting, and dense embeddings via LSA have long been used to represent documents [MRS08] [DDL90], and can equivalently be applied to sentences. More recent work has focused primarily on connectionist approaches, relying on neural networks that are usually trained via gradient descent optimization. Our distinction of deep vs. shallow approaches essentially differentiates between neural networks with complex architectures, and simple bilinear-style models that may involve different pre- or post-processing methods.

4.1.1. Deep approaches

The recursive autoencoder (RAE) architecture by Socher et al. [Soc+11b] [Soc+11a] was an early attempt at explicitly learning unsupervised sentence embeddings through deep neural networks. In this approach, a recursive tree of autoencoders is constructed over each sentence, where nodes in the tree encode their two children into a fixed-size vector. The leaf nodes are represented by word vectors, and the root node represents the full sentence embedding. The tree is greedily constructed to minimize the reconstruction loss over child nodes. Socher et al. successfully apply this model and related approaches to sentence classification tasks such as sentiment analysis and paraphrase detection. The MV-RNN model by Socher et al. [Soc+12] expands on this approach, representing words, phrases and sentences as $(\text{vector}, \text{matrix})$ 2-tuples, where the matrix is applied to a node's neighbor vectors. Though the representations in MV-RNN are not usually trained in an unsupervised manner, the model nevertheless has some relevance to this work as it also utilizes 2-dimensional matrix representations.

Following the introduction of shallow `word2vec` word embedding model, a number of deep sentence embedding approaches utilizing similar ideas were presented. Building on the sequence-to-sequence architecture of Sutskever et al. [Sut14], Kiros et al. introduced the skip-thoughts model [Kir+15], which trains an LSTM encoder-decoder network to predict the previous and next sentence given a sentence of some coherent corpus. The output vector of the encoder can be used as a general-purpose sentence embedding. This is in some ways similar to how the `skip-gram` architecture in `word2vec` is trained to predict words in a given context. Also based on this approach is work by Gan et al. [Gan+17], who introduce an encoder-decoder architecture based on convolutional neural networks, trained to predict the next sentence in a text. The common theme of these models is to apply the distributional hypothesis [Har54] on the sentence level. Of course, it is not obvious that the hypothesis would hold true on this level as well; the meaning of a sentence may not necessarily be defined by the nature of the sentences that tend to surround it. Nevertheless, research by Poaljnar et al. [PRC15] suggests that surrounding sentences do contribute to overall sentence semantics.

One drawback to these methods is that their training requires coherent, multi-sentence text corpora, which makes them harder to apply in some domains where such text is not easily available (e.g. Twitter data). Sequence-to-sequence sentence embedding approaches that do not rely on predicting surrounding sentences include a weakly supervised encoder-decoder architecture by Palangi et al. [Pal+16], which learns representations by mapping search engine queries to result texts, as well as the SDAE model by Hill et al. [HCK16], an encoder-decoder trained to reconstruct a noisy sentence with random deletions and permutations.

Finally, we draw attention to Lin et al. [Lin+17], who introduce a self-attention model

that generates 2-D embeddings that capture particularly important parts of a sentence. This model is similar to the approach we introduce in this work in that it represents sentences as matrices rather than simply as vectors, and the penalization term employed by the authors shares some similarity to the orthogonality requirement of subspace basis vectors. However, the training of this self-attention model is contingent on downstream tasks such as sentiment classification, and is thus not trained in an unsupervised manner.

4.1.2. Shallow approaches

The Paragraph Vector model by Le & Mikolov [LMC14] (sometimes referred to as `doc2vec`) is one of the first proposed text embedding algorithms based on `word2vec`. In this model, a special token is added for each sentence[1] in a corpus, after which a slightly modified `word2vec` training scheme is applied. The special token representations are trained by using them to predict missing words in the sentence, and after training they may be used as sentence representations. To generate representations for unseen sentences, an inference step via gradient descent is required, which is cited as a drawback by Kiros et al. [Kir+15].

FastSent by Hill et al. [HCK16] is essentially a shallow version of the skip-thoughts model of Kiros et al. Instead of representing sentences as sequences and applying an LSTM-based architecture, both input and output sentences are represented as bag-of-words vectors, and prediction is realized through a simple additive log-linear model.

Wieting et al. [Wie+15] contribute an embedding method that is trained on the Paraphrase Database [GVC13], and also show that sophisticated deep models like skip-thought underperform simple word-vector averaging on semantic similarity tasks such as the SemEval STS tasks and SICK [Mar+14].

Kenter et al. [KBR16] contribute the Siamese CBOW architecture, where sentence embeddings are also simple word vector averages, but word embeddings are directly trained for this purpose, in that the similarity of consecutive sentence pairs is maximized during training as the training criterion (and that of other random sentences minimized).

Arora et al. [ALM17] further investigate averages as an effective embedding mechanism and contribute their own word embedding model. They argue that a weighted average of sentence word vectors (weighted inversely to word frequency) followed by removal of the global first principal component (they call this the "common discourse vector") should serve as a baseline for sentence embedding models.

Sent2vec [PGJ17] extends the `word2vec`'s `CBoW` model to train embeddings for n-gram

[1] Technically one can use chunks of text of any granularity, e.g. sentences, paragraphs, or documents.

features. The sentence embeddings in this model are still simple vector sums; however, the model manages to outperform many other approaches on textual similarity tasks.

Discussion of the subspace model by Mu et al. [MBV16] will follow in Section 5.

4.1.3. Differences in methods

A comprehensive quantitative evaluation and comparison of sentence embedding methods is difficult to achieve, due to various issues such as availability of implementations, varieties in training hyperparameters, corpora and preprocessing steps, as well as the rapid progress being made in the field. Hill et al. [HCK16] provide a systematic comparison of a different unsupervised embedding methods. They find that, simply put, different models perform well at different tasks, but that no overall "best" model can clearly be determined.

Qualitatively, we find that valuable insights can be gained from a variety of different approaches, and these can be combined effectively. Many of the models listed here build on common basic building blocks, and specific techniques used are often compatible. For instance, the frequency weighting scheme and common discourse removal of Arora et al. [ALM17] could be applied to almost any model that involves direct use of pre-trained word embeddings.

A common theme in the research reviewed here is that shallow models tend to perform better on tasks that require evaluating sentence similarity, while deep models are better for supervised classification tasks [Wie+15] [HCK16] [PGJ17]. Our own model is a fully unsupervised shallow sentence embedding method.

4.2. Subspace approaches in word embeddings

Subspace methods have not yet been extensively applied to word vectors; however, interest in the intersection of these topics appears to be rising. Applications so far span a variety of problem settings.

Mahadevan et al. [MC15] use a Grassmannian approach to improve performance on the analogy task introduced by Mikolov et al. [MYZ13]. Their insight was that vectors of related words occupy a low-dimensional subspace, and they use a similarity measure on this subspace to solve the analogy task. Kumar & Araki [KA16] also make use of the notion that "analogies" or "relations" in word vector spaces roughly reside on subspaces, and make this idea explicit by introducing a training regularizer that forces vectors representing a certain relation (e.g. country-capital) to approximately occupy a rank-1 subspace. They also explore the idea of extending such an approach to higher-rank subspaces, but do not implement it.

Subspaces have also been applied to the problem of quantitatively evaluating word vector models. Tsvetkov et al. [Tsv+15] evaluate vector models by aligning word vector subspaces to manually constructed linguistic resources. Yaghoobzadeh & Schütze [YS16] extract subspaces from a full vector model to evaluate the model with respect to certain criteria they identify as important for representations.

Rothe et al. [RES16] and Rothe & Schütze [RS16] apply global subspace decompositions to word embeddings, in order to obtain interpretable subspaces that reflect specific phenomena such as sentiment or other features that are particularly salient for a certain downstream task, while maintaining the overall structure and similarity measures of the full vector space.

Other work in the area includes identifying and removing gender bias from word embedding models [Bol+16], domain adaptation [Ast+15], and visualization of word embedding models [Liu+16].

5. Related Work: Grassmannian Sentence Embeddings

Our work is based on the Grassmannian approach to sentence embedding introduced by Mu et al. [MBV17]. The main insight behind their model is the empirical observation that words in sentences tend to cluster a few semantic "topics", or more specifically that the word vectors of a sentence approximately reside on a low-rank subspace.

Intuitively, this idea can be demonstrated by considering the similarities of groups of words in a sentence. Consider the following example:

```
Thanksgiving has been celebrated nationally on and off since 1789, after
        Congress requested a proclamation by George Washington.
```

We can group the words in this sentence by (subjective) topical coherence, resulting in groups like `("Thanksgiving", "celebrated")`, `("Congress", "nationally", "Washington")`, etc. We can visualize such a grouping by assigning colors to words according to their group:

```
Thanksgiving has been celebrated nationally on and off since 1789, after
        Congress requested a proclamation by George Washington.
```

This insight suggests that sentences could be represented as mixtures of semantic topics (for this sentence, e.g. `"mostly blue, some red, ..."`). Such coherent groupings tend to exist for most natural sentences. In fact, co-occurrence based training methods for word vector models assume that a word's typical context defines its meaning.

5.1. Method

Mu et al.'s method, which we will refer to as Grassmannian Sentence Embedding (GSE), formalizes this intuitive notion using vector space semantics and Grassmannian subspaces. Word vectors such as those resulting from `word2vec` or `GloVe` are well-suited for capturing the type of semantic word similarity we subjectively applied in the example above, and the separate topical directions of a sentence can each be represented by orthogonal vectors in the semantic space. In this approach, words are not discretely

Algorithm 1 GSE algorithm

1: **procedure** ComputeGSE($\mathbf{w}, k, \mathrm{v}(\cdot)$)
2: $M \leftarrow \{\mathrm{v}(\mathbf{w}_i) | w_i \in \mathbf{w}\}$
3: $USV^T \leftarrow \mathtt{SVD}(M)$
4: $A \leftarrow V^T[0:k,:]$
5: **return** A

assigned to clusters, but instead a soft clustering is made, i.e. each word has some activation level for each cluster. This approach is somewhat similar to LSA modeling for documents; however, in LSA a subspace of a sparsely populated high-dimensional space is found *globally*, whereas in this approach a separate subspace of very low rank is found for each sentence independently (and the ambient space is already relatively low-rank).

Formally then, the GSE approach is to stack the word vectors of a sentence into a matrix M, and compute a low-rank subspace that optimally fits this matrix which can be efficiently computed using (truncated) SVD [Shl14]. [1]

Algorithm 1 gives a method that, given a word embedding model $\mathrm{v}(\cdot)$ and a rank k compute a subspace representation $A \in \mathrm{Gr}(k, d)$ of a sentence $\mathbf{w}$. As noted in Section 2.2, such a representation is not unique, since rotations of the basis A still represent the same subspace; however, at this point, this does not matter for practical purposes.

An approximation of a resulting embedding from this process is shown in Figure 5.1. We visualize a 2-dimensional Grassmannian embedding in a 3-dimensional ambient space. The vectors are projected down from 300 dimensional glove vectors and visualized as points in 3-D space, while the plane (or the contained axis vectors) represents the actual embedding. Unlike described in Algorithm 1, for better visualization the data was centered in this instance.

Dimensionality

Mu et al. conduct an empirical investigation to determine a good choice for k: they collect around 50k sample sentences, compute subspace embeddings for different values of k, and consider the fraction of total energy captured by the embedding. In this context, energy essentially refers to amount of variance in the stacked word vector

[1] Note that this is very similar to performing PCA, with the difference that in PCA, the data matrix X is centered via $X \leftarrow X - \overline{X}$. According to personal communication with one of the authors, this step is omitted in their method. Centering the data would move the domain of this technique from the Grassmannian to the affine Grassmannian, and would require a different set of mathematical tools. Following the methodology of Mu et al., we do not apply centering, in order to provide a more direct comparison to their method.

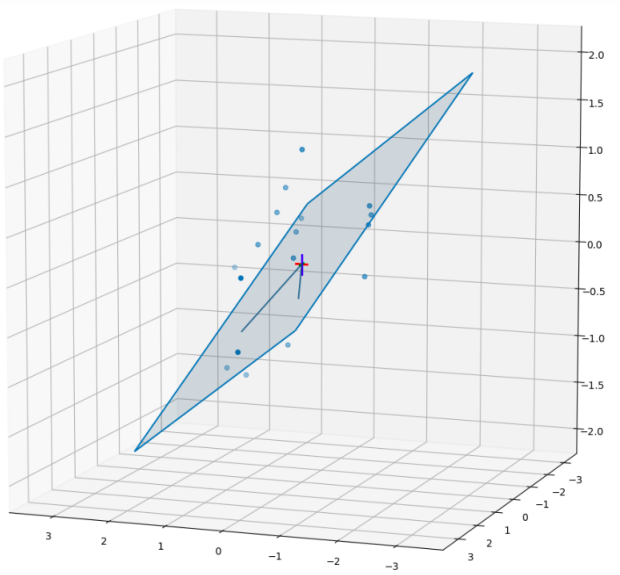


Figure 5.1.: Visualization of a Grassmannian sentence embedding ($d = 3, k = 2$).

matrix. The energy captured by the top k components of the SVD $M = USV^T$ is calculated as $\sigma_k^2 = \sum_{i=1}^{k} \text{diag}(S)_i^2$. The total variance can be computed either as $\|M\|^2$, if a truncated SVD is used, or by simply taking the full sum $\sigma_n^2 = \sum_{i=1}^{n} \text{diag}(S)_i^2$ where n is the sentence length (assuming no words are duplicated). Mu et al. select k such that on average, 80% of the energy in a sentence is retained, i.e. $\frac{\sigma_k^2}{\sigma_n^2} \geq 0.8$. This leads to a selection of $k = 4$.

Cosine similarity on the Grassmannian

Mu et al. expand the vector definition of vector cosine similarity to define sim_{Cos}, a generalization to the subspace domain:

$$\begin{aligned} \text{sim}_{\text{Cos}}(A, B) &= \left(\frac{1}{k} \sum_{i=1}^{k} \cos^2(\theta_i) \right)^{\frac{1}{2}} \\ &= \frac{1}{\sqrt{k}} \left\| AB^T \right\|_F \end{aligned}$$

For $k = 1$, this is equivalent to cosine similarity on vectors. This measure is closely related to the projection metric:

$$\mathrm{sim}_{\mathrm{Cos}}(A,B) = \left(\frac{\mathrm{sim}_{Proj}(A,B)}{k}\right)^{\frac{1}{2}}$$

Values of $\mathrm{sim}_{\mathrm{Cos}}(A,B)$ range from 0 (totally dissimilar) to 1 (identical). As the square root of the projection kernel is also a kernel [HL08], and the only other addition is a constant normalization term, $\mathrm{sim}_{\mathrm{Cos}}(A,B)$ is also a kernel on $\mathrm{Gr}(k,d)$.

5.2. Results

Task	ST	SC	D2V	GloVe		skip-gram	
				Avg	Subspace	Avg	Subspace
STS12	41.05	47.50	31.87	51.63	**53.88**	35.75	48.08
STS13	31.34	46.14	43.08	46.95	**59.73**	39.73	48.60
STS14	44.44	60.41	33.37	51.24	**65.56**	42.13	62.20
STS15	26.75	30.66	25.42	47.16	**69.67**	45.72	63.93
Average	35.90	46.18	33.43	49.24	**62.21**	40.83	55.70

Table 5.1.: Mu et al.'s reported average Pearson's correlation (x100) for the SemEval 2012 - 2015 STS tasks. Best results in bold.

Mu et al. evaluate the GSE model on the 2012-2015 SemEval semantic textual similarity tasks [Agi+12][Agi+13][Agi+14][Agi+15], and compare a range of other models. They test both `skip-gram` and `GloVe` vectors as the basis for their method, and compare to simple vector averaging, as well as three neural network based approaches: `skip-thought` (ST) vectors, Siamese CBOW [KBR16], and doc2vec [LMC14]. Their results are reported in Table 5.1. On average, their approach outperforms simple vector averaging by 13.9% on these tasks, and the neural network based approaches by 20.5%, with `GloVe` vectors clearly outperforming `skip-gram` vectors.

5.3. Downstream applications

Besides strong performance on semantic textual similarity tasks, Mu et al.'s Grassmannian approach to sentence/phrase embedding has also been used for a variety of novel downstream applications such as word-sense induction and disambiguation [MBV16],

preposition sense disambiguation [Gon+17], achieving state-of-the-art results in both domains, as well as testing for compositionality [GBV17]. These approaches make use of the subspaces in different ways, such as considering intersections of different subspaces or the distances/similarity between them. Such work highlights the broader applicability of Grassmannian methods to various problems in distributional semantics and NLP more generally. While in this work we do not perform experiments in these domains, we draw attention to such applications to provide additional motivation for the further investigation and improvement of the GSE model.

6. DGE: Model improvements

We introduce Dynamic Grassmannian Ellipsoid (DGE) embeddings, a subspace based approach that makes a variety of improvements on the GSE method. We incorporate the insight that more information can be captured by subspace embeddings and that they may be dynamically sized and compared. As a result, we do not use subspaces as the underlying mathematical object of our model, but rather ellipsoids that inhabit subspaces.

We outline our introduced changes in three separate steps, to show that each change is independently effective and improves the overall model performance. At each step we evaluate to a previous baseline. We also evaluate against a set of other unsupervised sentence embedding models for an overall comparison in Section 7.

6.1. General improvements

We first consider a pair of simple hyperparameter and learning method changes to improve the baseline performance of the subspace embeddings. Our first simple change is to make use of sub-word features via `FastText` vectors, and the second is to implicitly incorporate word frequency information into the model via weighted PCA.

Subword features via `FastText`

Since word embeddings in `FastText` are computed as sums of sub-word embeddings, it stands to reason that they may exhibit the type of linear sub-structure that is found in e.g. `skip-gram` vectors to a similar or even higher degree. The following is a simplified example to illustrate this: if $\mathrm{v}(\text{"blackboard"})$ is computed as $\mathrm{v}(\text{"black"}) + \mathrm{v}(\text{"board"})$, and $\mathrm{v}(\text{"whiteboard"}) = \mathrm{v}(\text{"white"}) + \mathrm{v}(\text{"board"})$, then $\mathrm{v}(\text{"blackboard"}) - \mathrm{v}(\text{"black"}) + \mathrm{v}(\text{"white"}) = \mathrm{v}(\text{"whiteboard"})$ holds. Of course such relationships do not hold trivially for most analogies in English; however, subword information is nevertheless beneficial in capturing word (and sentence) semantics, and substantially reduces the number of OOV terms encountered at test time.

Frequency weighting via wPCA

The second change we introduce at this stage is the implicit incorporation of word frequency information into the model. It has long been known that frequent words carry less information, and weighting schemes such as TF-IDF are commonly used in NLP and IR approaches.

In the GSE model, basis matrices are computed using PCA, which treats every data word equally. In this context, word frequency information can thus be incorporated by assigning lower weights to more frequent terms, and then computing a subspace to fit the word vectors using weighted PCA (wPCA). Essentially, whereas in standard PCA the k-th component is computed to minimize its squared distance to the data after subtraction of component $(k-1)$, in wPCA the *weighted* squared distance is minimized (where weights are applied sample-wise). The resulting subspace basis will thus be a "better" fit, in that the information contained in more important words is prioritized and more accurately reflected, while unimportant words are disregarded.

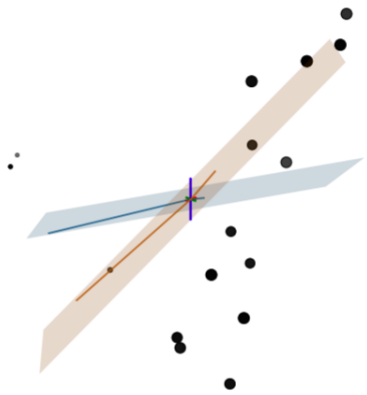


Figure 6.1.: Effect of wPCA on subspace embedding. Blue subspace from PCA, orange from wPCA. Larger point size indicates a higher word weight.

We borrow the weighting scheme used by Arora et al. [ALM17], where each word w is weighted inversely to it's global occurrence probability $p(w)$, and a hyperparameter α is used to control the overall strenght of the weights:

$$\omega(w) = \frac{\alpha}{\alpha + p(w)}$$

Word probabilities can be estimated using a reasonably large corpus, and are computed as each word's frequency divided by the total number of words in the corpus. Larger values of α lead to weights that are close to 1 for all words, while small values lead to

stronger weighting effects. We do not specifically tune the value of α in this work, but instead follow Arora et al.'s example and use $\alpha = 0.001$.

Evaluation

We evaluate the changes in three steps: `FastText` only, wPCA only, and finally `FastText` plus wPCA. For our evaluation we follow Mu et al. and use the SemEval STS tasks, and additionally include the STS2016 dataset [Agi+16]. The baseline model is our own implementation of "vanilla" GSE using `GloVe` vectors. Note that our implementation underperforms Mu et al.'s reported results, but it nevertheless serves as our initial baseline to first illustrate the relative improvements achieved by the changes we introduce. We ultimately compare against the actual results reported in the work of Mu et al. in Section 7.1.

All models are set to the same rank of $k = 4$, with an ambient vector space of dimensionality $d = 300$. Word probabilities are estimated on the Toronto Book Corpus [Zhu+15], and we also use this corpus to train `GloVe` and `FastText` vector models. `GloVe` was trained using a window size of 15, and terms occurring at least 5 times were kept, using the implementation at `https://nlp.stanford.edu/projects/glove/`. FastText was trained using default parameters, using the implementation at `https://github.com/facebookresearch/fastText`. We use the same stopword list in all models.

Task	No change	FastText only	WPCA only	WPCA+fasttext
STS12	0.448	0.501	0.459	**0.533**
STS13	0.435	0.547	0.444	**0.553**
STS14	0.539	0.637	0.561	**0.640**
STS15	0.601	0.707	0.637	**0.727**
STS16	0.591	0.666	0.608	**0.669**
Average	0.523	0.612	0.542	**0.624**

Table 6.1.: Pearson's correlation for `FastText` and wPCA vs. vanilla GSE.

Table 6.1 shows the evaluation results for these changes. Both changes in fact independently improve overall correlation with human similarity judgements, and their combination again leads to an improvement in every case. `FastText` alone has a stronger effect than wPCA, leading to an increase of 8.88% alone (vs. 1.9% for wPCA). Applied together, performance over the baseline increases by 10.16%.

6.2. Dynamic rank capture

In GSE, all representations share the same rank k, which Mu et al. choose such that representation on average capture 80% of the energy in a sentence. One problem with this approach is that it may strongly under- or overfit particular sentences, leading to poor representations. Intuitively, longer sentences may require a higher-dimensional space to model than short sentences. Instead of choosing a global k that fits sentences to 80% energy on average, it's also possible to vary k for each sentence and choose it such that the right rank is chosen depending on the distribution of word vectors within the sentence. This is akin to constructing each space such that it is as "large" as it needs to be, and not smaller or larger.

Formally, under this approach, k is chosen to be as small as possible while still capturing at least a given threshold proportion of overall energy thresh_e:

$$\begin{aligned} k &= \underset{k'}{\text{argmin}} \sum_{i=1}^{k'} \text{diag}(S)_i^2 / \sigma_n^2 \\ &= \underset{k'}{\text{argmin}}\, \sigma_{k'}^2 / \sigma_n^2 \\ &\text{subject to } \sigma_{k'}^2 / \sigma_n^2 \geq \text{thresh}_e \end{aligned}$$

Figure 6.2 shows distributions of subspace ranks that result for a different threshold values, computed on a 100k sentences sample from BookCorpus. We can see that even at $\text{thresh}_e = 0.8$, only less than 60% of the sentences are being fully captured by fixed-size subspaces with $k = 4$. Such a model effectively underfits the remaining sentences. We also visualize $\text{thresh}_e = 0.9$, as this is the threshold we ultimately use in our experiments.

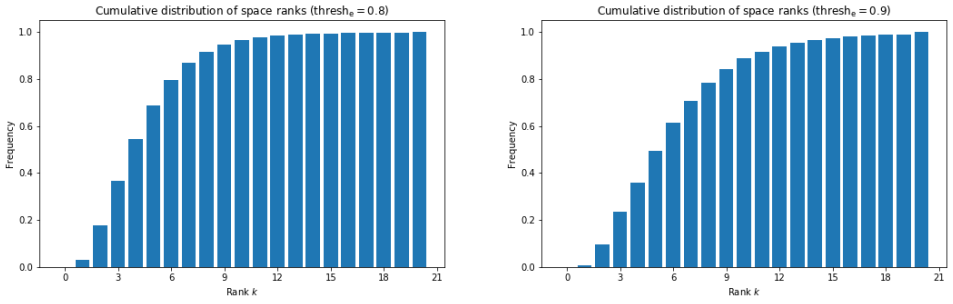


Figure 6.2.: Histograms showing the cumulative distributions of subspace ranks for $\text{thresh}_e = 0.8$. and $\text{thresh}_e = 0.9$.

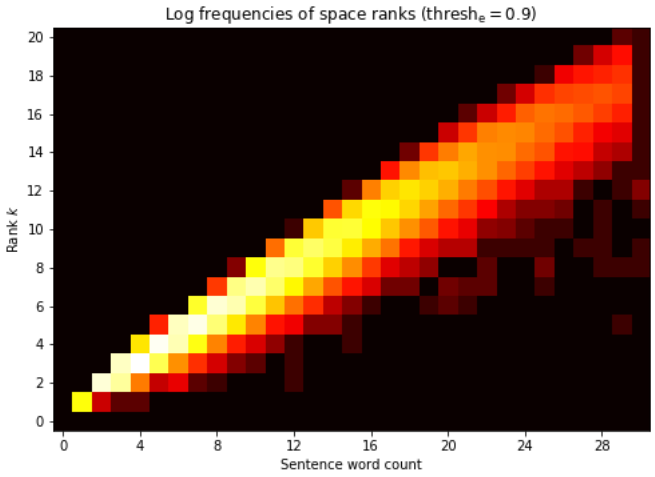


Figure 6.3.: Heatmap of rank frequencies for different sentence lengths (thresh$_e = 0.9$).

As already suggested, the final subspace rank is strongly related to the sentence length. To investigate this effect, we visualize the distribution of $\log(\text{rank frequency} + 1)$ of different ranks k as a function of sentence length (without stopwords or OOV terms) in Figure 6.3. We find that the intuition that sentence length predicts the required space rank holds true, and that the selected k appears to grow sub-linearly as a function of sentence length. This aligns with the interpretation that long sentences do not necessarily introduce more variety of semantic information with each added word, but that they effectively tend to stick to a smaller set of "topics".

Similarity

As each subspace may now have a different dimensionality, the previously given formulation of sim_{Cos} no longer works. We apply the results introduced in Section 2.3.1 to the definition of sim_{Cos} to generalize it for subspaces $A \in \text{Gr}(k,d), B \in \text{Gr}(l,d)$:

$$\begin{aligned}
\text{sim}_{\text{Cos}}(A,B) &:= \left(\frac{1}{\min(k,l)} \sum_{i=1}^{\min(k,l)} \cos^2(\theta_i) \right)^{\frac{1}{2}} \\
&= \frac{1}{\sqrt{\min(k,l)}} \left\| AB^T \right\|_F \\
&= \left(\frac{\text{sim}_{\text{Proj}}(A,B)}{\min(k,l)} \right)^{\frac{1}{2}}.
\end{aligned}$$

Again, this definition is equivalent to vector cosine similarity for $k = 1$, and it is also equivalent to the definition introduced by Mu et al. for $k = l$.

Evaluation

We evaluate the dynamically sized embeddings under the same conditions as outlined in Section 6.1. We vary thresh_e between 0.7 and 0.9 to investigate the effect of capturing subspaces more approximately or more precisely. The baseline we compare to is the previous best-performing model, i.e. rank 4 subspaces using `FastText` features and wPCA. The dynamically sized models use the same features, and also employ wPCA, only the threshold hyperparameter is varied.

Task	$k = 4$ (baseline)	$\text{thresh}_e = 0.7$	$\text{thresh}_e = 0.8$	$\text{thresh}_e = 0.9$
STS12	**0.533**	0.510	0.503	0.518
STS13	0.553	0.556	0.558	**0.586**
STS14	0.640	0.620	0.659	**0.676**
STS15	**0.727**	0.700	0.714	0.720
STS16	0.669	0.672	0.689	**0.695**
Average	0.624	0.612	0.625	**0.639**

Table 6.2.: Pearson's correlation for dynamically sized vs. fixed-size subspace embeddings.

Results are shown in Table 6.2. Note that for two of the 5 tasks, fixed size embeddings outperform the dynamic embeddings; however, dynamic sizing at $\text{thresh}_e = 0.9$ improves overall performance by 1.46% on average. Performance at $\text{thresh}_e = 0.8$ is roughly equivalent to fixed embeddings at $k = 4$, while capturing less energy leads to a decrease in performance.

Note that the average subspace dimensionality at $\text{thresh}_e = 0.8$ is $\bar{k} = 4.8$, and at $\text{thresh}_e = 0.9$ it is $\bar{k} = 6.3$. This suggests that simply making subspaces larger in general adds some benefit. We also separately evaluated fixed size subspaces at $k = 5$ and $k = 6$ dimensions, and found that performance increases slightly, to 0.629 and 0.625 respectively. This is a better result than dynamic sizing at the $\text{thresh}_e = 0.8$ setting; however, the performance reached at $\text{thresh}_e = 0.9$ is still significantly higher, suggesting that dynamic sizing adds a clear benefit.

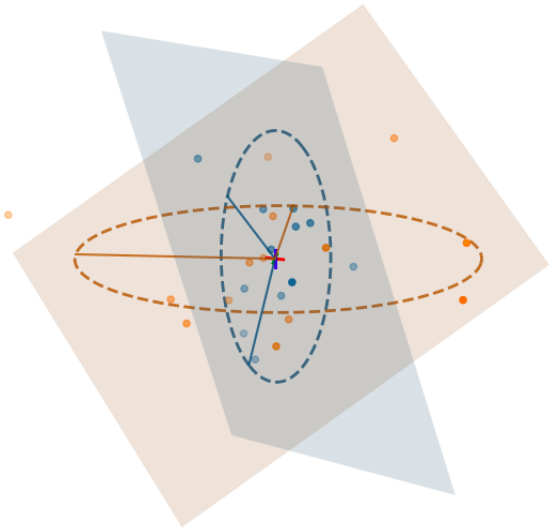


Figure 6.4.: Two DGE embeddings ($k = 2$, $d = 3$).

6.3. Ellipsoid representations

Dynamically sized subspaces allow us to more accurately capture the total energy of a sentence; however, our approach so far does not represent information about the actual distribution of this energy. Two sentence representations may be very alike if they concern the same set of topics, even if one sentence is primarily focused on a topic that has low significance in the other. We incorporate this additional information by complementing the subspace representation A with an additional importance vector s_A of length k. The vector value $s_{Ai} = \text{diag}(S)_i^2 / \sigma_n^2$ represents the relative importance of the ith basis vector as a fraction, so for instance a value of $s_A 1 = 0.7$ means 70% of the energy in the sentence is explained by the first component.

For convenience, we explicitly represent a sentence as a pair of one basis matrix and one importance vector (rather than scaling the basis vectors and using only one matrix). This representation $\{A, s_A\}$ can be understood as an origin-centered ellipsoid embedded in a k-dimensional subspace, as the importance values can be seen as radii across the different orthogonal dimensions, thus we refer to this model as Dynamic Grassmannian Ellipsoid (DGE) embeddings. The vector s can be easily calculated from the diagonal singular value matrix S resulting from the SVD that is calculated as part of the embedding process. Figure 6.4 visualizes low-dimensional DGE embeddings on two sample sentences.

Ellipsoid similarity

The ellipsoid representation also warrants a new method of evaluating sentence similarity, since sim_{Cos} does not account for vector length. We want a single similarity measure that captures both the degree of alignment of basis vectors, and the degree to which their relative importances of those vectors are matching (which are expressed as vector magnitudes, i.e. scalar values). sim_{Cos} exactly captures the former only.

Looking at vectors first, we can define a simple measure that expresses the agreement of their magnitudes as simply the ratio between the smaller and the larger value:

$$\text{sim}_{\text{mag}}(\vec{x}, \vec{y}) = \frac{\min(\|\vec{x}\|, \|\vec{y}\|)}{\max(\|\vec{x}\|, \|\vec{y}\|)}.$$

Alternatively, we can define the measure directly on scalars: $\text{sim}_{\text{mag}}(s_1, s_2) = \min(s_1, s_2) / \max(s_1, s_2)$. This measure will vary between 0 and 1, where 1 means perfect agreement in magnitude (importance).

sim_{Cos} for basis matrices A, B is essentially a normalized Frobenius norm of the inner product matrix of A and B. These inner products are themselves just the cosine similarities of each pair of basis vectors $(\vec{a}, \vec{b})$, since the basis vectors are unit length. A simple way to combine sim_{Cos} and sim_{mag} on the matrix level is thus to multiplicatively penalize the individual inner products in $(AB^T)_{ij}$ using the corresponding importance agreements $\text{sim}_{\text{mag}}(s_{Ai}, s_{Bj})$. We additionally introduce another hyperparameter β for controlling the degree of this penalization, via $1 - \beta(1 - \text{sim})$. At $\beta = 1$ this leads to full multiplicative penalization and at $\beta = 0$ there is none. We define the magnitude agreement over importance vectors s_A and s_B in outer-product fashion:

$$\text{sim}_{\text{mag},\,\beta}(s_A, s_B) = \mathbf{1}_{k,l} - \beta \left(\mathbf{1}_{k,l} - \frac{\left\langle s_A^T, s_B \right\rangle_{\min}}{\left\langle s_A^T, s_B \right\rangle_{\max}} \right)$$

where $\left(\left\langle \vec{x}^T, \vec{y} \right\rangle_{\text{f}}\right)_{ij} = f(\vec{x}_i, \vec{y}_j), i \in [1, k], j \in [1, l]$ and $\mathbf{1}_{k,l}$ is the $k \times l$ matrix of ones.

Finally, we define the overall ellipsoid similarity:

$$\text{sim}_{\text{Elps}}(\{A, s_A\}, \{B, s_B\}) = \left(\frac{\left\| AB^T \odot \text{sim}_{\text{mag},\,\beta}(s_A, s_B) \right\|_F}{\min(k, l)} \right)^{\frac{1}{2}}.$$

This measure is again equal to 1 for identical ellipsoids, but now a deviation in either the subspace spanned or the importance vectors will each affect the total similarity. When the spanned subspaces are orthogonal, this measure is equal to 0.[1] For $\beta = 0$, it reduces to sim_{Cos} since the penalization matrix will simply be $\mathbf{1}_{k,l}$ (and is applied via element-wise multiplication).

[1] Technically, it could also equal zero when all-zero importance values occur in either ellipsoid (and the other basis vectors are all orthogonal), but this does not occur in practice.

Task	$\beta = 0$ (baseline)	$\beta = 0.5$	$\beta = 1$	$\beta = 0.5$ (no WPCA)
STS12	0.518	0.564	0.563	**0.571**
STS13	0.586	**0.613**	0.571	0.581
STS14	0.676	**0.668**	0.615	**0.668**
STS15	0.720	0.741	0.716	**0.746**
STS16	0.695	**0.710**	0.682	0.694
Average	0.639	**0.659**	0.629	0.652

Table 6.3.: Pearson's correlation for ellipsoid features under different values of β.

Evaluation

We again follow the same experimental setup as introduced in Section 6.1, this time investigating the effect of incorporating ellipsoid features by varying the β parameter. $\beta = 0$ corresponds to be baseline, i.e. the strongest performing model from Section 6.2. All other model hyperparameters are the same as for the baseline, with the exception of one experiment where we disable WPCA for comparison, as in practice regular PCA / SVD has considerably faster implementations.

Table 6.3 shows the experimental results. We can see that moderate weighting with $\beta = 0.5$ increases performance by 2%, but that full importance penalization with $\beta = 1$ leads to a slight decrease of 1%. We can also see that once all other improvements are introduced, the effect of WPCA alone becomes relatively small at only 0.7% improvement.

Algorithm

Algorithm 2 DGE algorithm

procedure ComputeDGE($\mathbf{w}, \mathrm{v}(\cdot), \alpha, \mathrm{thresh}_e$)
  $M \leftarrow \{\mathrm{v}(w_i) | w_i \in \mathbf{w}\}$
  $\omega \leftarrow \{\frac{\alpha}{\alpha + p(w_i)} | w_i \in \mathbf{w}\}$
  $S, V^T \leftarrow \mathrm{WPCA}(M, \omega)$
  $\sigma_n^2 = \sum_i \mathrm{diag}(S)_i^2$
  $k \leftarrow \min\{k' | \sum_i^{k'} S_i^2 / \sigma_n^2 \geq \mathrm{thresh}_e\}$
  $A \leftarrow V^T[0:k, :]$
  $s_A \leftarrow \mathrm{diag}(S)[0:k] / \sigma_n^2$
  return A, s_A

For reference, Algorithm 2 outlines the full procedure of computing DGE embeddings.

7. Discussion

7.1. Comparison to other approaches

STS Task	OURS		Mu et al.	Arora et al.	Others		
	DGE	GSE	GSE	(unsup.)	ST	SC	D2V
2012	**56.4**	44.8	53.9	56.2	41.1	47.5	31.9
2013	**61.3**	43.5	59.7	56.6	31.3	46.1	43.1
2014	66.8	53.9	65.6	**68.5**	44.4	60.4	33.4
2015	**74.1**	60.1	69.7	71.7	26.8	30.7	25.4
Average	**64.7**	50.6	62.2	63.3	35.9	46.2	33.4
Difference	-	-14.1	-2.4	-1.4	-28.8	-18.5	-31.2

Table 7.1.: Comparison of DGE to GSE and other semantic similarity models on SemEval STS tasks. Results are Pearson's correlation x100.

We compare our DGE model to a variety of other unsupervised sentence embedding methods on SemEval STS tasks in order to put its overall performance into context. We include our baseline implementation of GSE, the results reported by Mu et al. for the same model, as well at the `skip-thought`, Siamese CBOW and `doc2vec` models that Mu et al. also compare to. We also include the model introduced by Arora et al. as it represents a strong baseline on sentence similarity tasks. The DGE model was configured as follows: `FastText` features with WPCA subspace learning, $\alpha = 0.001$, $\beta = 0.5$ and $\text{thresh}_e = 0.9$.

We report our results in Table 7.1. DGE outperforms the other models on all tasks except on STS 2014, where it is slightly outperformed by Arora et al.'s model. Overall, however, DGE still outperforms Arora et al.'s model by 1.4%. DGE improves on our own GSE baseline by around 14.1%, and 2.4% when compared to the original GSE baseline. We suspect that DGE performance may be optimized even further, as Mu et al.'s superior results to our baseline implementation of their model suggest that our underlying vector model, pre-processing, or other hyperparameters, are not sufficiently tuned. Nevertheless, even given this lower starting point DGE represents a significant

improvement over other approaches.

Our choice of comparison models is meant to highlight primarily the relative improvement of DGE over the GSE baseline, as well as contextualize the model more generally. We omit many related models in this comparison. One example are the paraphrastic embeddings of Wieting et al. [Wie+15]. This model received much attention, and also outperforms our model on some tasks, but it relies on what is essentially supervised training for semantic relatedness of sentences, as Wieting et al. make use of a paraphrase database [GVC13]. FastSent by Hill et al. [HCK16] is also an interesting model to compare to; however, only results for the STS 2014 task are available. Our model does not represent the state of the art on the STS tasks in general; however, it achieves very strong performance given that it is a relatively simple unsupervised approach based directly on pre-trained word embeddings.

7.2. Insights

Each of the changes we introduce to the GSE model to arrive at DGE was separately shown to be an effective improvement. Some of these techniques may be applicable to other approaches as well, and some are inspired by other approaches themselves. We find that `FastText` vectors bring a substantial improvement to our model, which suggests that restricting semantic models the word-level discards important semantic information. This insight could be applied in many other models for various tasks. We also introduce a similarity measure that allows for incorporating vector magnitude information, something that is commonly discarded in approaches based on the vector space model. We suspect that such approaches might also be applicable in purely vector based models, though we do not investigate this further in this work.

In practical applications of DGE, it may be helpful to consider the changes we introduce separately and weigh their individual benefits and tradeoffs. For instance, the use of WPCA adds significant computational cost to the model, but the overall improvement it contributes is comparatively small, once other changes are also introduced. Other properties may be effective for performance gains on specific tasks like sentence similarity but hard to incorporate into downstream approaches. For example, we investigate approximate nearest subspace search in Part II, where we disregard the ellipsoid features of DGE, as they are not easy to combine with the method employed there.

7.3. Future work

There are a variety of aspects to the DGE model that could be investigated and tuned further. In terms of overall performance tuning, we do little tuning of hyperparameters. Evaluating the model under more fine-grained combinations of values for α, β and thresh_e as well as further tuning of factors like the `FastText` training (hyperparameters and training data) and preprocessing steps is likely to lead to increased performance.

In fact, the training of the word embeddings themselves could be augmented to directly optimize for learning subspace based embeddings, for example by introducing a regularizer during training that encourages word vectors in a sentence or context to be close to some shared subspace.

Investigating the ellipsoid model more closely may also lead to alternative formulations of similarity measures on these objects. The sim_{Elps} measure we introduced was constructed on an intuitive understanding of (concentric) ellipsoid similarity specifically in the context of sentence embeddings. In considering commonly used ellipsoid similarity measures we would generally face the problem that these measures deal with "full-rank" ellipsoids that have some level of overlap with a meaningful volume, while the ellipsoids in the DGE model reside on different subspaces and generally have no "overlap" in the conventional sense. For instance, we considered using the Bhattacharyya distance[1], viewing the ellipsoids as multivariate Gaussians with zero mean and covariate matrix $\Sigma_A = A_s^T A_s$ (where A_s is A with rows scaled according to importances s_A); however, since the determinant of a rank-deficient matrix is 0, this distance is undefined for such distributions.

Another aspect that could be considered in such research would be the relaxation of the Grassmannian constraint in DGE, and instead defining embeddings on the affine Grassmannian, i.e. applying PCA or WPCA *with* mean subtraction, and subsequently comparing ellipsoids with different center points. The center coordinates of the ellipsoid would then become another component of the overall sentence representation. Distances between subspaces of different dimensionalities on the affine Grassmannian have also been studied [LWY16] and could be applied to this approach.

Just as many other shallow approaches including word averaging, word order is not accounted for in DGE. While the model nevertheless outperforms commonly used models that do account for word order, it is still the case that word order and sentence syntactical features do contribute to overall sentence semantics. Investigating ways of successfully incorporating such information is another potential path towards further improvement of sentence embedding approaches like DGE.

Finally, it would be interesting to apply DGE on downstream tasks such as those

[1] For Gaussians with mean 0 covariate matrices Σ_A, Σ_B, it is $\frac{1}{2}\ln((\det\Sigma_A + \det\Sigma_B)/2\sqrt{\det\Sigma_A \det\Sigma_B})$.

referenced in Section 5.3. This would provide insight into the extent to which DGE is a useful general-purpose model, as opposed to solely effective for semantic similarity tasks.

8. Conclusion

We introduce Dynamic Grassmannian Ellipsoids, a shallow unsupervised sentence embedding method based on word vectors and subspaces. We introduce a number of changes and additions to the subspace embedding model (GSE) introduced by Mu et al [MBV16], and provide both intuitive motivation and quantitative evidence that these changes are indeed effective. Overall, DGE is highly effective on sentence similarity evaluations, improving on the initial GSE approach by 2.4% on STS 2012-2016, as well as outperforming the recently introduced strong baseline model by Arora et al.[ALM17] by 1.4%. No supervised training is required for the model, and all of its algorithmic components are effectively post-processing operations based on an underlying pre-trained word vector model.

Besides strong quantitative results, we note that, unlike many vector based models, GSE and DGE have some qualitatively interpretable qualities. In both models, basis vectors can be related back to "topical" words that support them in a sentence, which provides insight into the make-up of the embedding and may be useful in decomposing the embeddings themselves in downstream applications. In DGE, the ellipsoids features additionally express the relative importance of each "topic" within the text.

We also contribute a number of auxiliary findings and techniques that may be useful in other work, such as the apparent superiority of `FastText` embeddings over `GloVe` for shallow sentence similarity models, the incorporation of vector magnitude in a similarity measure, a generalization of Mu et al.'s sim_{Cos} measure for subspaces of arbitrary dimensionality, as well as some theoretical background on this measure.

Part II.

Scalable Dynamic Nearest Subspace Search

9. Introduction

Subspace representations are known to be effective for applications in a variety of fields like computer vision, bioinformatics, and communication. More recently, they have also been applied to natural language data, where they have proven useful in addressing problems like semantic similarity, sense induction and disambiguation, and identification of compositionality. However, subspace representations have a high number of parameters, and the evaluation of similarity on the Grassmannian manifold tends to be significantly more expensive than that of vectors in the same ambient space. This renders these techniques difficult or impractical to apply in certain large-scale settings. In vector based models, approximate search methods such as locality sensitive hashing (LSH) [Dat+04] are commonly used to enable the efficient retrieval of data points' nearest neighbors. Related techniques have also been proposed for subspace data; however, the application domains of such methods tend to be limited to computer vision problems, where the structure of the underlying data may have substantially different properties compared to the domain of textual data.

We follow a recently introduced technique for approximate nearest neighbor search on the Grassmannian manifold. The approach proposed by Iwamura et al. [IKK16] decomposes the similarity calculation on the Grassmannian manifold into multiple similarity calculations in Euclidean space. We make a set of alterations to their method, including replacing their projection kernel sim_{Proj} with the sim_{Cos} measure, adapting the method to handle subspaces of varying dimensionalities, and adding a "full" distance calculation step after approximate neighbor retrieval. Our experiments indicate that the method is applicable to the textual domain; however, the overall efficiency and speed of the method will likely not allow for the same level of scalability as that of vector based methods. Nevertheless, compared to brute-force search, we are able to achieve a significant speed-up (~3x-6x) while maintaining high top-10 accuracy (~85%-94%).

We also propose `Schuhlöffel`, an approach where subspace basis sparsity is optimized on the Stiefel manifold prior to indexing the basis vectors, to encourage sparsity in the inner products between bases. We find the method slightly improves the performance of our approximate nearest neighbor search; however, it does so somewhat inconsistently. Nevertheless, in certain settings, the method improves performance significantly, and we believe it may be more effective in other application domains that are more pertinent to those settings.

10. Related Work

The problem of approximate nearest neighbor discovery in vector spaces under similarity measures such as cosine similarity, hamming distance, or euclidean distance has received much attention, and many efficient and accurate solutions exist. Among the most common approaches to this problem is locality sensitive hashing (LSH) [Dat+04], in which vectors are hashed such that objects with similar features receive the same hash values. Many efficient implementations exist of vector based approximate search methods, in this work we rely on the NMSlib library[1] [BN13], specifically we make use of their Hierarchical Navigable Small World (HNSW) [MY16] graphs implementation. This approach has shown to be very efficient and accurate on high-dimensional vector data.

In the subspace domain, approximate nearest-neighbor search (also called approximate nearest subspace search, or ANSS) is still a lesser-developed topic. Basri et al. [BHZ07][BHZ09][BHZ11] present a method (commonly referred to as BHZ) that is based on mapping subspaces to points and subsequently approaching the ANSS problem in the vector domain. While the method is an effective approach to ANSS, the resulting vectors have dimensionality $d(d+1)/2$ (where d is the ambient space dimensionality), which is prohibitively large for high dimensional domains (such as the GSE subspaces based on word embeddings we examine here). Subsequent work by various researchers has given attention to subspace-to-point nearest neighbor retrieval [Liu+12] [VJG14], as well as point-to-subspace retrieval[SZW14], in various domains such as computer vision and active learning.

To address shortcomings in previous work relating to high-dimensional performance as well as sub-linearity guarantees, Wang et al. introduce GLH [Wan+13], an ANSS method that adapts the LSH approach to the subspace domain. Their method hashes subspaces according to their distances to randomly chosen randomly chosen lines, and addresses general cases such as point-to-subspace and subspace-to-subspace search. Their method is particularly effective for higher-dimensional ambient spaces when compared to BHZ.

Similar to the BHZ method, Iwamura et al. [IKK16] approach the ANSS problem from a vector perspective; however, their core insight is that similarity calculations between subspaces can be decomposed into a number of d-dimensional vector similarity

[1] `https://github.com/searchivarius/nmslib`

calculations that can subsequently be aggregated to form the full basis similarity. They report equivalent accuracy at significantly lower cost compared to BHZ and GLH. As we implement and extend this method ourselves, we will give a more detailed overview of it in Chapter 11.

11. Method

The approach introduced by Iwamura et al. serves as the basis for our method. We extend this approach, adapting it to the domain of dynamically sized subspaces, and in addition we contribute a variety of performance improvements to the approach. We finally apply the resulting method to the textual domain for quantitative evaluation.

11.1. Basic approach

Iwamura et al. propose an ANSS method that consigns the actual nearest neighbor search to the vector domain. Their core insight is that the computation of similarity under certain kernels in the Grassmannian domain can be represented as an aggregation of individual vector similarities. For example, consider the projection kernel:

$$\begin{aligned} \mathrm{sim}_{\mathrm{Proj}}(A, B) &= \left\| AB^T \right\|_F^2 \\ &= \sum_{i=1}^{k} \sum_{j=1}^{l} \left((A_i)(B_j)^T \right)^2 \end{aligned}$$

This kernel can be seen as a sum over individual vector inner products. Further, since the goal of ANSS is to find pairs of subspaces with high similarity, we are particularly interested in cases where this sum contains high-valued inner products. This problem, efficient retrieval of high-valued inner product pairs, is exactly the case addressed by ANN methods in the vector domain.

In addition, Iwamura et al. observe that inner product distributions over a collection of subspace data follows a roughly Gaussian distribution with a "fat tail" on the top end of the spectrum (see Figure 11.1 for our recreation of this analysis), implying that the values of subspace similarities of close pairs may be dominated by a small number of very high vector inner products.

Thus, efficient near subspace retrieval can be realized by indexing all database subspace basis vectors in a simple vector ANN index (keeping track of what vectors belong to what subspace), and at query time querying this index for every query basis vector Q_i as well as the complements $-Q_i$, collecting results (i.e. individual $(Q_i)(B_j)^T$ and $(-Q_i)(B_j)^T$ terms) and summing their squares, leading to an approximation of the

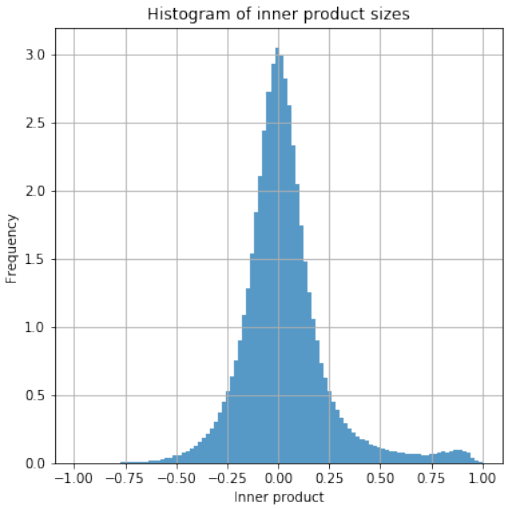


Figure 11.1.: Distribution of inner products for a sample of sentence subspaces generated using GSE.

overall value of $\mathrm{sim}_{\mathrm{Proj}}(Q, B)$). The results are finally returned sorted according to this approximation. Iwamura et al. show that this approach is both effective and efficient on various image datasets, leading to a speed-up of up to 7.3x with no loss of accuracy.

11.2. Alterations

The core of our implementation follows this idea closely, with a few alterations. First, we replace the $\mathrm{sim}_{\mathrm{Proj}}$ similarity measure with $\mathrm{sim}_{\mathrm{Cos}}$. Since

$$\mathrm{sim}_{\mathrm{Cos}}(A, B) = \left(\frac{\mathrm{sim}_{\mathrm{Proj}}(A, B)}{\min\{k, l\}} \right)^{\frac{1}{2}},$$

this change is relatively trivial. Note that this also naturally extends the original approach to subspaces of different dimensionalities.[1]

We also observed that the "fat tail" of inner products in our analysis is less pronounced than that reported by Iwamura et al., thus we expected our data to be less suited for the direct approximation of similarity through partial summing of inner product squares. We thus implemented another change: under the `exact` configuration, the method uses Iwamura et al.'s approximate method to generate a set of candidates, and subsequently re-orders the candidate set according to the *full* subspace similarity. This

[1] Iwamura et al. also propose applying Ye & Lim's results [YL16] to achieve this.

change was also motivated by the empirical observation that the purely approximative approach underperformed in our experiments.

We introduce this change in two variants: in the first, the full set of candidates is re-ranked according to the full similarity, and in the second, a threshold prefilter (pf) is supplied and only the top pf $\times$ 100% of candidates (according to approximate similarity) are re-ranked.

11.2.1. `Schuhlöffel` - Optimizing subspace bases on the Stiefel manifold

As the accuracy of the similarity approximation under this method depends on the underlying ANN method finding close neighbors to basis vectors, the similarity of basis matrices is no longer important from a Grassmannian perspective alone. Specifically, since a rotation of a basis matrix within its spanned subspace does not affect the subspace it defines in terms of Grassmannian similarity (the subspace defined by the basis matrix remains the same), such a rotation may change individual angles between pairs of basis vectors of the original and the rotated subspace. Thus, a key shortcoming is that even a subspace with an *identical* span to that of the query may not always be retrieved under this approach.

We propose a countermeasure to this problem by considering the basis matrices not just as elements of the Grassmannian manifold $\mathrm{Gr}(k, d)$, but rather as elements of the Stiefel manifold $V_k(\mathrm{R}^d)$. The Stiefel manifold is the set of orthonormal k-frames in R^d, and, like the Grassmannian, it exhibits Riemannian structure. Unlike the Grassmannian, elements in the Stiefel manifold identical if and only if the basis matrices are actually the same, not if they just span the same subspace.

Given this view, the idea of `Schuhlöffel` is, prior to indexing, to alter each subspace basis matrix such that they are closer to some common "orientation" in the ambient space I_d. More specifically, we aim reduce the basis matrices distances to the axes of the ambient space This effectively brings each basis matrix close to a common orientation, and encourages sparsity. In turn, this also encourages sparsity in the inner products of close pairs of matrices, as basis matrices more "aligned". The intuitive motivation of `Schuhlöffel` for our ANSS method is that this preprocessing operation should result in a) a higher likelihood that the underlying ANN method will retrieve relevant near vectors and b) improved similarity approximation, as the overall similarity becomes more dominated by individual high inner products.

Objectives for rotation / reflection

In general, we will alter each basis A by rotating (and/or reflecting) it, within its span, towards some objective that is static in the ambient space. However, it is not obvious

what rotation constitutes a good alignment. We propose and evaluate a number of such objectives: `trace`, `nearest`, and `max`.

Objective 1: `trace`

In `trace`, we (arbitrarily) choose to rotate or reflect each basis so that it is maximally close (as measured on the Stiefel manifold) to the "first" k axes of the ambient space, i.e. the matrix $I_{k,d} = [I_k 0_{k,(d-k)}]$. The squared euclidean distance on the Stiefel manifold is defined as $\text{dist}^2_{\text{Euc}}(A, B) = 2\text{tr}\left(I_k - 0.5(AB^T + BA^T)\right)$. For $B = I_{k,d}$, we have

$$\begin{aligned}\text{dist}^2_{\text{Euc}}(A, I_{k,d}) &= 2\text{tr}\left(I_k - 0.5\left(AI_{k,d}^T + I_{k,d}A^T\right)\right) \\ &= 2k - \text{tr}(A).\end{aligned}$$

Thus, minimizing this objective is equivalent to maximizing the trace of A under the constraint that $\text{span}(A)$ does not change.

Objective 2: `nearest`

Instead of arbitrarily choosing $I_{k,d}$ as a target orientation, we can also make a more motivated selection. Let $\pi(k, d)$ represent the set of d-dimensional permutation matrices that are truncated to k rows, and $\pi^{\pm}(k, d)$ be the same but allowing elements to have negative signs. Under the `nearest` objective, we first find closest (via Grassmannian similarity) basic k-hyperplane (spanned by some $B_0 \in \pi(k, d)$), and subsequently select as the rotation target the basis $B_1 \in \pi^{\pm}(k, d)$ for this hyperplane that is already closest to A on the Stiefel manifold. This effectively means we find an ordered set of k axes that are already close to A, and then optimize to move A closer to them. We find B_0 as $\text{argmin}_B \text{sim}_{\text{Proj}}(A, B)$, with $B \in \pi(k, d)$. Let $\text{argsort}_{i,m} f(i)$ denote the ordered (decreasing order) set of inputs i' that yield the highest m values of $f(i')$. The optimal solution for B_0 is then given by

$$(B_0)_{ij} = \begin{cases} 1, & \text{if } i \in \text{argsort}_{i',k} \sum_{j'}^{k} \left(A_{i'j}\right)^2 \\ 0, & \text{otherwise.} \end{cases}$$

B_0 is a $k \times d$ dimensional matrix with k values equal to 1, each on a different row and column. More precisely, the top k columns maximizing the total sum of squares are selected, and we denote their indexes as j_{best}.

Similarly, the overall optimization target is defined as $B_1 = \text{argmin}_B \text{dist}^2_{\text{Euc}}(A, B)$ with $B \in \pi^{\pm}(k, d)$. The one-hot/one-cold ($+1/-1$) rows of B simply select certain entries in A (and may invert them), and the span of B_0 and B_1 must be the same. Minimizing the distance can therefore be re-cast as the following two-part problem:

Name	$\ell(A)$
`trace`	$-\mathrm{tr}(A)$
`nearest`	$-\sum_{ij} \left(B_1 \odot A\right)_{ij}$
`max`	$-\sum_i \left(\max_j \left\|A_{ij}\right\|\right)$

Table 11.1.: Loss functions for the different objectives in `Schuhlöffel`.

1. Select k indexes (i, j) of different rows and columns of A such that $\sum_{(i,j)} |A|_{ij}$ is maximal and $j \in j_{\text{best}}$,[2]
2. Determine the correct sign associated with each index such that the closest axis (positive or negative direction) to A is selected.

The first step is equivalent to the assignment problem and can be solved using the Hungarian Algorithm on the $k \times k$ matrix given by selecting the columns of A indexed by j_{best}. The second step is also simple, as the sign is the same as the sign of the corresponding value in A. Finally, note that B_1 is a permutation of the rows of B_0, possibly with some flipped signs.

Objective 3: `max`

The final objective, `max`, is similarly motivated to `nearest`, but much simpler. Instead of actually finding the closest basic k-hyperplane via optimizing an assignment problem, we make the simplifying (sometimes incorrect) assumption that the correct selection of k axes is simply given by selecting the index of maximum absolute value along each row of A. The objective is then to maximize the sum of these maximum absolute values.

An overview of the loss functions $\ell(A)$ defined by these three objectives can be found in Table 11.1. These will serve as the the actual loss functions in our optimization.

[2]This follows from the definition of the Euclidean distance on the Stiefel manifold. Similar to how we maximize the trace in the `trace` objective, we now maximize a different particular selection of elements of A, but we use the absolute value because a highly negative element still signifies that the corresponding axis is close (but oriented in the opposite direction).

Optimization

The overall optimization procedure for finding suitable rotations can be cast as a gradient descent problem with certain constraints. In general, our optimization problem is:

$$\begin{aligned} \underset{A'}{\text{minimize}} \quad & \ell(A') \\ \text{subject to} \quad & A'^T A' = I_k,\ \text{span}(A') = \text{span}(A). \end{aligned}$$

We first observe that for any pair of square orthonormal matrices $R_1, R_2 \in V_k(\mathrm{R}^k)$, the product $R_1 R_2$ is also an element of $V_k(\mathrm{R}^k)$, since these matrices simply rotate and/or reflect points on multiplication, preserving distances. Similarly, for $A \in V_k(\mathrm{R}^d), R \in V_k(\mathrm{R}^k)$, the product RA remains in $V_k(\mathrm{R}^d)$; in fact it remains in the span of A. Optimization of rotation (or reflection) constrained to the span of A can therefore be cast as an optimization over R of the product RA.

Further, the set of orthonormal $k \times k$ matrices can be parameterized using the Cayley transform, which constructs an orthonormal matrix for every skew-symmetric $k \times k$ matrix C. We thus write R as

$$R = \text{Cayley}(C) = (I_k + C)^{-1}(I_k - C).$$

Combining these insights, we can restate our optimization problem:

$$\begin{aligned} \underset{C_\theta}{\text{minimize}} \quad & \ell((I_k + C_\theta)^{-1}(I_k - C_\theta)A). \\ \text{subject to} \quad & C_\theta^T = -C_\theta. \end{aligned}$$

The constraint of this problem (skew-symmetry of C_θ) is fulfilled by choosing $\binom{k}{2}$ free parameters θ_i and defining C_θ such that $\text{triu}(C_\theta)_i = \theta_i$ and $C_{ij} = -C_{ji}$ (where $\text{triu}(\cdot)$ selects elements $(i, j), i > j$ from a matrix and flattens the result). The optimization yields assignments for the parameters θ, which are finally used to construct the desired rotated basis $A' = (I_k + C_\theta)^{-1}(I_k - C_\theta)A$.

Under this formulation, we finally use gradient descent to perform the optimization. We use the `Adagrad` algorithm [DHS11] for training, and our overall implementation relies on PyTorch [Pas+17].

11.3. Implementation

We implement the described approach in Python, relying on the `nmslib` [BN13] package for the underlying vector nearest neighbor search. Specifically, we construct vector

indexes using the library's `HNSW` [MY16] method, using the following hyperparameter settings: efConstruction : 2000, efSearch : 50, maxM : 64, maxM0 : 64. For reference, a pseudocode description of the algorithm is provided in Algorithm 3.

Algorithm 3 ANSS query algorithm. DB is a pre-constructed vector ANN search index, c refers to the number of neighbors this index should return.

procedure SUBSPACEQUERY($\mathbf{A}, c, \text{pf} = 0.2, \text{exact} = \mathit{True}, \text{DB}$)
  $R_{all} \leftarrow \{\text{ANNQuery}_{\text{DB}}(\vec{a}, c) | \vec{a} \in [A, -A]\}$
  $R_i \leftarrow$ Vector similarities R_{all} grouped by DB subspace ID
  $\text{Sim} \leftarrow \left(\sum R_i^2 / \min\{k, l\}\right)^{\frac{1}{2}}$
  Sort Sim in descending order
  IF pf THEN:
   keep only top pf $\times$ len(Sim) candidates
  IF exact THEN:
   $\text{Sim} \leftarrow \{\text{sim}_{\text{Cos}}(A, B_i) | B \in \text{DB} \wedge B \in \text{Sim}\}$
   Sort Sim in descending order
  return Sim[: 10]

11.3.1. Experimental settings

We test our approach on a sample of sentences from the LiveJournal corpus introduced by Paredes et al. [Par+17]. We generate subspace embeddings for 1,000 query sentences and 10,000 database sentences using the *DGE* method introduced in Section 6; however, we use no word-frequency re-weighting in estimating the subspace embeddings, use the subspace-only setting and compare subspaces using sim_{Cos} with $\alpha = 0.$. We determine subspace ranks dynamically using $\text{thresh}_e = 0.9$, i.e. our experiment test performance in a setting where subspace dimensions vary.

Prior to evaluation we a) compute `Schuhlöffel` optimization (where necessary) and b) construct indexes using `nmslib`. The processing time of this step is not taken into account in our reported query times (we assume this time is amortized over sufficiently many queries). We evaluate top-10 accuracy, i.e. the true-positive rate in the closest 10 spaces returned averaged over all queries, against the query "speed", i.e. the number of queries processed per second. The query speed is affected by the number of neighbors c we require the underlying vector ANN index to return. For larger neighborhoods, query time increases, in the edge case of a full database search the algorithm is approximately the same as brute-force neighbor search.

Ground truth results are generated using simple brute-force comparison of a query subspace with every subspace in the database. To compare our method to brute force

search taking also into account relative time difference over lower accuracies, we also report "partial" brute-force results. These are simply the theoretical time taken and top-10 accuracy reached if only scanning a smaller percentage of the database were scanned.

Our implementation and experiment scripts are available at `https://github.com/phdowling/dynamic-grassmannian-ellipsoids`.

11.4. Evaluation

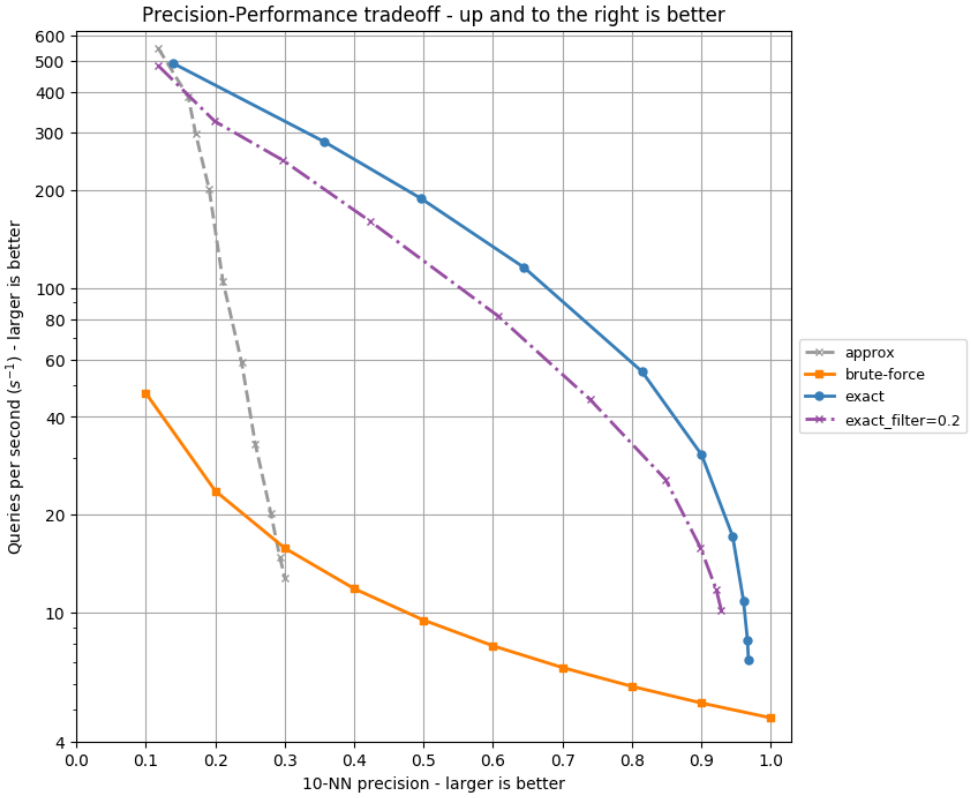


Figure 11.2.: Top-10 accuracy vs. queries per second for `approx`, pf = 0.2, and `exact` configurations.

We test our implementation as described in Section 11.3.1. We first highlight the effect of the different precision settings: `approx`, `exact`, and `approx` with pf = 0.2. The results of this evaluation are shown in Figure 11.2.

The `exact` setting, i.e. adding precise re-evaluation of subspace similarities on top of the approximate computation, drastically improves the methods performance in our experiments. The improvement remains strong even when only the top 20% of candidates are kept for precise evaluation, suggesting that good initial nearest neighbor candidates are generated by the original method, but similarity approximations are imprecise.

Conversely, while pre-filtering has a positive effect on the methods accuracy, this method is barely faster than exact similarity computation but offers less overall im-

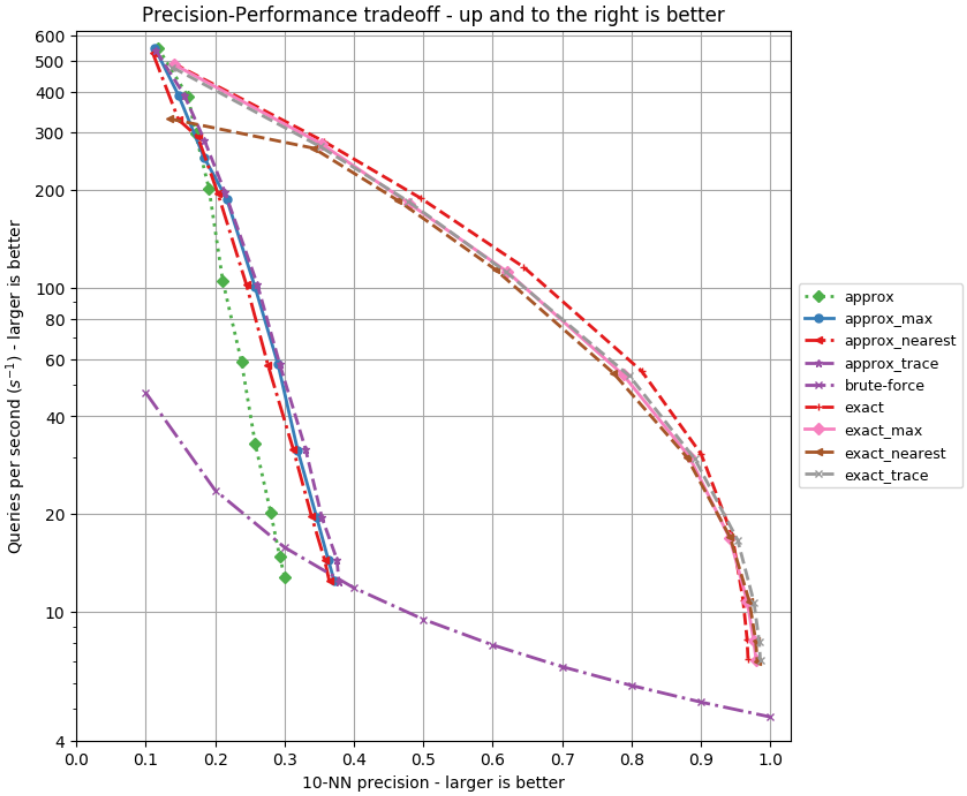


Figure 11.3.: Top-10 accuracy vs. queries per second for different `Schuhlöffel` optimization objectives under `approx` and `exact` configurations.

provement. To reach a top-10 accuracy of 90% under the pf = 0.2 setting, the method is ultimately slightly slower than simply using the `exact` configuration. More generally, the precision-performance tradeoff curve for this setting never crosses that of the `exact` setting; the speed-up it offers is too small to justify the accompanying loss in performance, and we therefore find this addition is not useful.

The `approx` method exhibits very low performance on this dataset, reaching a maximal top-10 accuracy of 0.3 at 12.7 queries per second (slower than an equivalent brute force search). This result is counter to the results reported by Iwamura et al., who use only the `approx` configuration, albeit on very different data.

Under the `exact` setting, our method offers a ~6x speed-up at a top-10 accuracy of 0.9 compared to brute force search; however, for the maximum accuracy of 0.97, the speed-up is less than 2x.

Next, we evaluate the effect of different `Schuhlöffel` optimization objectives under

Algo	Build time (s)	AUC	Best top-10 acc.	Best acc. q/s
approx	170.34	35.89	0.30	12.73
approx_trace	173.23	48.23	0.38	12.39
approx_nearest	175.79	42.84	0.36	12.42
approx_max	175.30	45.80	0.37	12.46
exact_pf=0.2	170.34	122.79	0.93	10.18
exact_pf=0.2_trace	173.23	117.63	0.93	10.03
exact_pf=0.2_nearest	175.79	105.84	0.93	10.04
exact_pf=0.2_max	175.30	114.40	0.93	10.04
exact	170.34	159.24	0.97	7.16
exact_trace	173.23	150.19	0.99	7.05
exact_nearest	175.79	131.56	0.98	7.03
exact_max	175.30	150.65	0.98	7.07
brute-force	0.0	11.26	1.0	4.73

Table 11.2.: Evaluation results for different ANSS configurations. AUC refers to the area under the accuracy vs. queries per second curve (as graphed in Figures 11.2 and 11.3).

the `approx` and `exact` settings. These results are shown in Figure 11.3. We also report area under the curve (AUC), and additional experiment results in Table 11.2.

All three optimization objectives (`trace`, `nearest` and `max`) are effective improvements to the `approx` setting, lifting maximal accuracy by around 7% on average at little loss in speed. For the `exact` setting, however, it appears that all methods only add slight accuracy improvements at high values of c, while slightly decreasing both accuracy and speed for most settings.

The `trace` objective performs best under the `approx` setting (AUC$=$ 48.23), while for `exact`, the `trace` and `max` objectives exhibit similar performances (AUCs 150.19 and 150.65 respectively). Still, both ultimately lose out to "vanilla" subspaces (AUC $=$ 159.24). There is a slight increase in maximal top-10 accuracy of 1-2% for all optimized methods; however, query speeds are also slightly lower. For pf $= 0.2$, the optimization does not appear to add any performance in terms of AUC, top-10 accuracy, or query time.

While `Schuhlöffel` optimizations are somewhat expensive to compute, they do not appear to have a significant impact on index build times themselves (their pre-processing times are not included in Table 11.2, but effectively increase overall build times by a significant amount).

12. Discussion

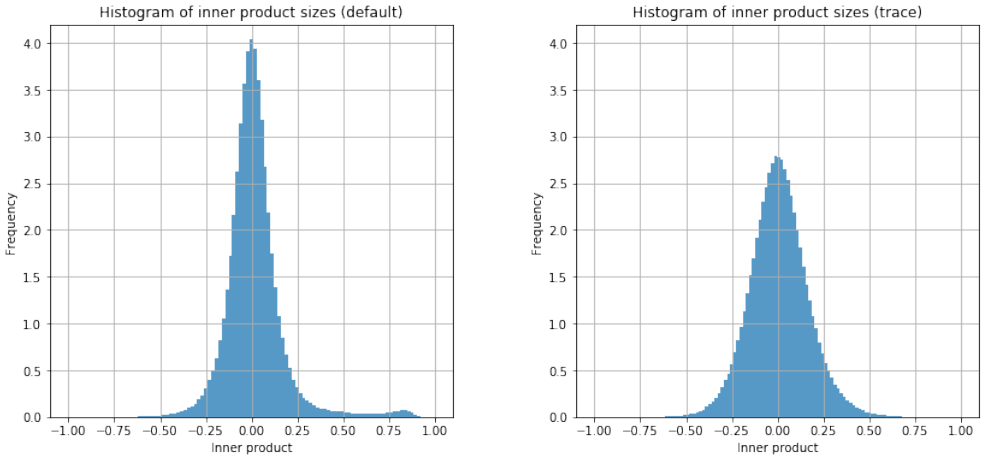


Figure 12.1.: Distribution of inner products using default subspace bases and `Schuhlöffel`-optimized subspaces under the `trace` objective.

One of the major discrepancies in our result when compared to the work of Iwamura et al. is the difference in performance of the `approx` configuration. Iwamura et al. make no mention of adding a full similarity computation step, therefore we believe the discrepancy may be due to the structure of data being modeled. As highlighted in Section 11.1, the fat tail in the inner product distribution of our data is less pronounced. This insight, combined with the poor performance of `approx` querying, leads us to postulate that sentence subspaces may exhibit a higher variety in structure and orientations than image subspaces.

The aforementioned difference of inner product distributions was one of the motivations for the `Schuhlöffel` optimizations we introduce. We are therefore surprised to find these optimizations do not have a positive overall effect on performance in the `exact` setting, and also do not elevate the `approx` setting to satisfactorily high accuracy.

On closer inspection, we find that `Schuhlöffel` does not actually achieve the desired effect of adding more "pronouncement" to the fat tail in inner product distribution,

in fact, as shown in Figure 12.1,[1] they appear to totally remove this fat tail, while widening the bell curve overall. This helps explain our empirical results: the shift means that squares of similarities are higher on average, but less extreme values are encountered. Thus, there are more sufficiently high inner products, evenly distributed, to help improve the `approx` setting (which essentially does not incorporate lower inner products at all). Conversely, in the `exact` setting, the additional *slightly* higher inner products that are not already incorporated through the full similarity are only "found" by the ANN method at very high values of c, where they lead to a more complete set of candidate subspaces to draw from (which are then ranked exactly).

We also find that a) there is not a large difference in the choice of optimization objectives for `Schuhlöffel`, and b) the `trace` objective slightly outperforms the other objectives. We postulate that this may be due to the fact that `trace` provides a single global orientation target, albeit a somewhat arbitrarily chosen one. In contrast, the other objectives individually determine an orientation target for each sample.

Overall, our experiments show that the approach by Iwamura et al. is viable for textual data and can lead to improved query performance when compared to brute-force search, but it does not yet enable the level of scalability that is feasible in purely vector-based methods. As far as we are aware, this work also represents the first evaluation of an approximate nearest subspace search method on textual data, and there remains much work to be done to fully characterize this domain.

12.1. Future Work

We see a few avenues of research that could improve the approach outlined here. First, we were surprised to find that our optimization targets for `Schuhlöffel` had a different effect on the inner product distribution than we anticipated. It may therefore be interesting to examine more optimization targets (or entirely different approaches on the Stiefel manifold) to find whether there are ways of achieving the desired effect, namely to increase sparsity in the distribution of inner products between `GSE` subspaces. Such methods could potentially also improve the Iwamura et al. approach in other domains.

We also postulate that both the fundamental method by Iwamura et al., as well as the `Schuhlöffel` optimizations, may prove more effective on Grassmannian data that also "naturally" exhibits more structural freedom on the Stiefel manifold. While subspace basis rotations result in equivalent representations from a strictly Grassmannian viewpoint, they do not make intuitive sense from a word-vector perspective. This is because the basis vectors are locally determined by the distribution of specific groups

[1] We only show the `trace` objective here, but we observe similar patterns for `nearest` and `max`.

of words, and rotating the basis vectors represents an arbitrary deviation from this distribution. For other types of data such as images, such subspace rotations may be more intuitively meaningful, due to global structure with some degrees of invariance to changes in rotation such as image content, changes in lighting conditions, etc. This idea is strictly theoretical at this stage, and more empirical investigations would be needed to confirm it.

Further, we currently perform the optimization on the Stiefel manifold using a relatively naïve gradient descent method, which takes considerable time to converge. This could possibly be sped up using e.g. curvilinear search [WY13] or other methods. Finally, we recognize a need for further investigation of existing ANSS techniques, as well as the introduction of entirely new approaches. Approximate search in the subspace domain remains a difficult problem which, if solved or significantly improved, could have a large impact in a variety of application domains.

13. Conclusion

We implement, extend, and evaluate a recently proposed method for approximate nearest subspace search on Grassmannian subspace sentence embeddings. Our alterations include evaluating subspace similarity across samples of different dimensionality, as well as introducing a re-ranking step that leads to significantly increased search accuracy.

With `Schuhlöffel`, we also propose and implement a set of optimization objectives to alter subspace representations on the Stiefel manifold. Although the empirical effects and results achieved with this method are not in line with our initial expectations, the approach adds performance in specific settings, and the method overall could serve as a basis for further research and improvements on ANSS (and potentially other problems). Recognizing that a Stiefel manifold-level view of subspace representations can inform the design of Grassmannian-level methods and algorithms is a vital step in this direction.

Our overall findings indicate that the performance of the approach proposed by Iwamura et al. appears to be subject to the overall structure of the subspace representations under consideration. Their method, even under a fully-approximative setting, yields competitive results on various image datasets; however, on textual data it is almost useless out-of-the-box, and requires refinements to be made useful. With these refinements, the method yields respectable performance, yielding up to $6x$ faster search at 90% top-10 accuracy compared to brute force search. While this is a significant speed-up, in practice it may prove insufficient for many large-scale applications.

Part III.

Grassmannian Embedded Topics

14. Introduction

We provide a qualitative exploration of different aspects and properties inherent to the DGE model, both on the sentence-level and document-level. We demonstrate that subspace axes and their corresponding importance weights can be interpreted as topic distributions embedded dynamically into an ambient semantic word-vector space. Conceptually, this view of DGE, which we refer to as Grassmannian Embedded Topics (GET), highlights similarities between DGE and Topic Models such as Latent Dirichlet Allocation (LDA). However, GET differs from LDA in a few key aspects, primarily the fact that in GET, "topics" are embedded in a continuous space dynamically and are allocated per-document, rather than globally. Nevertheless, under GET, topics in different texts can still be easily semantically compared because they inhabit the same ambient semantic vector space.

Under GET, each text is mapped to a dynamic set of topic vectors, which are not globally shared. This allows for more precise representation of different topical aspects that are present in each text, as well as their relative importance. Previously introduced topic models tend to share a set of global topics, and each text is represented according to the extent to which certain topics match the content of that text, allowing for less nuance in individual representations.

This insight can be made useful in various ways within human-facing downstream applications. We show how topics in GET can be interpreted for human consumption, and we introduce simple and intuitive query re-weighting methods for both subspace and vector based information retrieval. We demonstrate both of these methods qualitatively, and apply the vector based query re-weighting scheme to Inquire, a tool for insight discovery in qualitative research.

The examples we present here are meant to provide a broader view of how subspace methods can be applied in text semantics, as well as serve as inspiration for the design of interpretable methods that may exhibit properties that are useful in interactive settings. We hope to highlight the fact that the Grassmannian approach not only allows for effective sentence similarity computations, but that the intuition behind the method that yields these similarities can be understood, and that this understanding can be utilized in a broader context.

15. Related Work

As a general line of research, topic models are most closely related to the interpretation of DGE we present here. Latent semantic analysis (LSA) [DDL90] represents one of the earliest models in this domain, employing SVD to find a dense low-rank approximation of a sparse term-document matrix. The term topic model was coined later, with the introduction of the Latent Dirichlet Allocation (LDA) model by Blei et al. [BNJ03], later followed by a range of related models such as the Hierarchical Dirichlet Process (HDP) [Teh+06] and Dynamic Topic Models [BL06]. The underlying idea of topic models is to represent documents as interpretable and comparable vectors of topics, where a topic is in turn some distribution over words. Topic vectors tend to be sparse, capturing the intuition that a document is usually well-represented by only a small number of topics.

Work has also been done on combining the techniques and concepts of topic models like LDA with word embedding methods like `word2vec`. Das et al. [DZD15] use pre-trained `word2vec` embeddings within an LDA-style model, such that words with low euclidean distances are likely to be grouped into the same topic. Batmanghelich et al. [Bat+16] similarly extend HDP using a distribution on the unit sphere that uses cosine similarity, yielding a model that both incorporates semantic regularities and automatically learning the number of topics. Moody [Moo16] further introduces the `lda2vec` model, which follows a similar idea to `doc2vec`, i.e. extending `skip-gram` by using a document vector (here, in conjunction with the current context word) to predict missing context words. However, in `lda2vec`, document vectors are inferred to be sparse vectors over a set of topics, and further, the word embeddings are jointly trained with the topics.

Common to all of these approaches is that they learn a set of global topics, and documents are modeled as distributions over some of these topics, in most cases represented by sparse vectors. In contrast, DGE foregoes learning any *global* topics, instead learning a small set of *local* topic vectors only.

16. Method

The core method of computing text representations under GET is the same as that of DGE, as outlined in Chapter 6. As such, GET is not so much a new model as it is a different view on DGE. We make small adjustments to interpret DGE more in the style of conventional topic models, and we also demonstrate how GET can be used to perform "directional" search in semantic vector spaces.

The idea behind GET is to remove the constraint that topics are global distributions, but rather that topics can be local, in the sense that each text can be about a variety of topics,which are defined as directions in word vector space, and allocated to best fit the current document. The underlying word embeddings are the only "global" aspect to the model, all topics and text representations are computed on the fly. Similar to LDA, each text is modelled as a set of scored topics, and the number of topics within a text varies dynamically according to text content. Topics vectors result from DGE as the subspace axes A, an the topic scores are simply the ellipsoid features s_A. The overall similarity measure of text representations is also unchanged, given as simply $\text{sim}_{\text{Elps}}(\{A, s_A\}, \{B, s_B\})$. We also make an addition to DGE, in that we also consider affine Grassmannian subspaces as representation objects.

16.1. Interpreting topics

The underlying subspace axes in DGE are computed to align well with the words in a given text, and represent our topic vectors. While they are embedded in the same space as word vectors, they can't simply directly be seen as words, instead their directions define topics, made up of the words with low angular distances (i.e. high cosine similarity). An important detail is the fact that words that characterize a specific topic can either have a very high or very low cosine similarity to the corresponding axis, since the axes of a subspace extend infinitely in both the positive and negative directions. Thus, the square or absolute value of cosine similarity is a good choice for a scoring function for words.

An intuitive interpretation of a topic vector can thus be computed by considering the words contained in the text, computing their topic score (absolute cosine similarity to the topic vector), and ranking the list of words by topic score in descending order. We refer to this scoring approach as the "local" characterization of a topic. The same

technique can also be applied for the full vocabulary, yielding a "global" interpretation of the topic.

As an alternative, we can also compute DGE in the *affine* Grassmannian, meaning we subtract the mean of the word vectors in the text prior to finding the subspace. While the representations resulting from this method are no longer compatible with our previously given similarity function, the resulting subspace / ellipsoid should more precisely capture the distribution of the data. To score words under this setting, we also subtract the data mean from each word vector prior to computing the absolute cosine similarity (as we are interested in the angle in terms of the affine subspace origin, not the ambient space). The score for word i and topic j is thus

$$\text{score}_A(w_i, j) = \left|\text{sim}_{\text{Cos}}(w_i - \mu_A, A_j)\right|,$$

where $\mu_A = \sum_i^m w_i/m$ is the data mean (i.e. the origin of the affine subspace).

16.2. Directed similarity

In addition to the overall text similarity $\text{sim}_{\text{Elps}}(\{A, s_A\}, \{B, s_B\})$, for GET we define a *directed* or partial similarity of documents, which takes the same form but re-weights or omits one or more topics in A (and/or in B). Mathematically, this is simple to achieve, simply by manually adjusting the corresponding importance scores in s_A. High scores will increase a topic's importance, and a score of 0 leads to complete dismissal of a topic (note that there should be no negative values in s_A). Topic importances are subsequently normalized, such that they again sum to thresh_e, in order for importance vectors to remain comparable.

This alteration may be useful in practical settings, such as in information retrieval systems. For instance, a user may find a sentence or document of interest and want to search for similar documents, but she is specifically interested in one or more topical aspects of the selected text. In that situation, she could either boost (by manually increasing topic importance scores) or discount (or entirely discard) certain topics before similarity search is performed. Search results are then encouraged to more precisely match the specific topics requirements that the user specified.

Note that this method is only compatible with non-affine representations under the formulation we present here.

16.3. Re-weighting for vector similarity

While the directed similarity measure defined above is useful for fine-grained topical control of semantic similarity evaluation of texts, it is still fundamentally based on a

subspace similarity measure. Thus, it may be too costly to compute in many settings of practical information retrieval systems (as we outline in Part II of this work), whereas vector based methods allow for very high speed and high scalability. We thus propose a semi-automated query refinement technique for vector based semantic search based on DGE/GET representations.

The technique relies on weighted averages of word vectors for sentence/text representations. It yields a set of k re-weighting configurations for a text with a $k-$dimensional DGE embedding. For topic i, the weights for each word are given by the topic score as defined in Section 16.1, in either the non-affine or the affine setting.[1] Selecting any given topic thus yields a different text vector that weights characterizing words for that topic more highly. These different representations could either be separately stored and indexed for search and retrieval, or a single overall average (e.g. unweighted, or frequency-weighted in the style of Arora et al. [ALM17]) is stored and the k weighting schemes are for queries only.

[1]The topic vector itself could also be used as a representation directly; however, we have found that this yields poor performance compared to weighted word vector averaging, when querying against a real sentence vector database.

17. Qualitative evaluation

17.1. Topic interpretation

We compute both non-affine and affine topics for an example document. The example text we consider here is about freedom of the press, and mentions a variety of events in different countries, as well as different general concepts. It is thus suitable for qualitatively exploring how well GET is able to capture a variety of topics in text. We report the four topics with the highest explained variance, once characterized by words contained in the text, and once for the global vocabulary.

Local characterization

We list the top 10 words for the top 4 topics found in the sample text in Tables 17.1 and 17.2. We are able to see clear distinctions between the contents of most topics, although there is some overlap between Topics 1 and 2 in the affine setting (both are related to killings and media / journalists).

Topic 0 (22.3%)		Topic 1 (5.1%)		Topic 2 (4.3%)		Topic 3 (3.4%)	
government	0.76	kosovo	0.52	europe	0.70	journalists	0.52
political	0.75	caruana	0.52	european	0.64	murder	0.46
whether	0.70	syrian	0.47	american	0.59	editorial	0.43
citizens	0.69	bashir	0.46	malta	0.59	journalism	0.43
people	0.69	kosovar	0.44	china	0.58	media	0.42
officials	0.69	assad	0.43	january	0.54	killing	0.42
politicians	0.69	myanmar	0.43	philippines	0.53	death	0.42
leaders	0.68	lambast	0.43	chinese	0.53	killed	0.42
likely	0.67	dictators	0.42	june	0.52	kill	0.41
others	0.67	impunity	0.42	slovakia	0.52	murders	0.39

Table 17.1.: Top 4 **non-affine** topics characterized by **local** vocabulary (words scored by absolute cosine similarity to topic vector).

Topic 0 (8.1%)		Topic 1 (6.1%)		Topic 2 (4.5%)		Topic 3 (4.0%)	
caruana	0.73	journalists	0.61	murder	0.54	news	0.58
slovakia	0.68	accused	0.55	media	0.52	rational	0.47
malta	0.67	jailed	0.53	journalists	0.52	authoritarian	0.46
kosovo	0.60	impunity	0.51	killed	0.51	week	0.46
daphne	0.60	killings	0.51	journalism	0.49	morning	0.44
martina	0.60	assassination	0.50	killing	0.48	undermined	0.43
slovak	0.59	intimidation	0.50	arrested	0.46	democracy	0.43
slovakian	0.59	well	0.50	death	0.46	humanity	0.42
pristina	0.59	condemning	0.48	international	0.46	arrested	0.42
tibor	0.58	condemned	0.48	murders	0.45	tweeted	0.42

Table 17.2.: Top 4 **affine** topics characterized by **local** vocabulary.

It is interesting to observe that the main difference between the non-affine and affine settings appears to be that topics appear to be shifted "to the left" in the affine setting, meaning that non-affine Topic 1 roughly corresponds to affine Topic 0, and non-affine Topic 3 is very similar to affine Topics 1 and 2. In addition, the non-affine Topics 0 and 2 are no longer present. The disappearance of Topic 0 may have an intuitive explanation: the first principal component of data that is not mean-centered tends to correspond approximately to the mean, since there tends to be high variance in that direction. Topic 0 in non-affine GET is therefore in some sense special, it will tend to capture an average of the vectors in the text, which in a sense gives an overview of the broad text topic. When affine subspaces are used, the mean of the data 0, so that this special Topic 0 effectively disappears and is simply replaced by the direction with the next-highest variance. This difference in the significance of Topic 0 is also highlighted by the fact that Topic 0 has a much higher explained variance in the non-affine setting (22.3%) than in the affine setting (8.1%).

Global characterization

Tables 17.3 and 17.4 show the same topics as before; however, here we use a global vocabulary for characterizations. A few interesting details become apparent when doing this.

First, we note that some topics that are strongly characterized by rare words exhibit many noisy characterizing words, for example Topic 1 in the non-affine case and Topic 0 in the affine case. This makes understanding the meaning of such topics more difficult under the global vocabulary. However, it also highlights the fact that GET is able to

Topic 0 (22.3%)		Topic 1 (5.1%)		Topic 2 (4.3%)		Topic 3 (3.4%)	
government	0.76	zlendi	0.57	europe	0.70	journalists	0.52
concerned	0.76	seitp202	0.56	usa	0.65	columnists	0.48
political	0.75	dv42	0.55	asia	0.65	correspondents	0.46
fact	0.72	islamists	0.55	european	0.64	murder	0.46
even	0.72	04.04.27	0.55	germany	0.63	editors	0.46
concern	0.71	chechnya	0.55	america	0.62	robbery	0.44
authorities	0.71	04.04.25	0.54	france	0.62	murderer	0.44
whether	0.70	polticians	0.54	italy	0.62	editorial	0.43
despite	0.70	jounalists	0.54	singapore	0.60	broadcasters	0.43
apparently	0.69	04.04.24	0.54	canada	0.60	newsrooms	0.43

Table 17.3.: Top 4 **non-affine** topics characterized by **global** vocabulary

Topic 0 (8.1%)		Topic 1 (6.1%)		Topic 2 (4.5%)		Topic 3 (4.0%)	
caruana	0.73	journalists	0.61	murder	0.54	news	0.58
zipslocal.com	0.71	dissidents	0.59	slaying	0.53	inherent	0.57
04.04.27	0.70	denounced	0.59	media	0.52	ideology	0.55
50.97.226.122	0.70	accused	0.55	stabbed	0.52	fundamentally	0.55
04.04.22	0.70	accuse	0.54	journalists	0.52	inherently	0.53
03.09.25	0.70	accusing	0.54	killed	0.51	societal	0.52
snapshot_pm	0.69	jailing	0.53	robbery	0.50	ideals	0.52
http://krwg.org	0.69	jailed	0.53	murderer	0.50	fundamental	0.52
765.361.6100	0.69	denouncing	0.53	arraigned	0.49	moral	0.51
04.04.25	0.69	complicity	0.53	journalism	0.49	insofar	0.51

Table 17.4.: Top 4 **affine** topics characterized by **global** vocabulary.

adequately capture topics that may be globally underrepresented, since both of the examples listed here are coherent under the local vocabulary (their possible topic labels under the local characterization could be "dictatorship" and "names/(slavic) regions, respectively").

Apart from this, the added vocabulary bears a more thorough characterization of what a topic is actually about. For instance, under the local vocabulary, Topic 1 in the affine setting appears to highlight words that are related to journalists, imprisonment and killings, as well as condemning. The same Topic under the global vocabulary becomes more clear, in that "journalists" is still the top-ranked word, but "dissident". "accuse" and "jailing" (and closely related terms) are now the dominating top words, suggesting that while "killing" or "assassination" are also related to this topic, this topic is more directly about dissident journalists being jailed (which qualitatively aligns well with the original text).

17.2. Directed subspace similarity

To illustrate the utility offered by directed subspace similarity as defined in Section 16.2, we give the following example. Consider the three short texts:

- T_A "The concert was great and I loved the band, but the light effects could have been better."
- T_B "This music venue offers great lights and stage effects."
- T_C "The music was incredible, the band created a great vibe. I loved it!"

DGE, using `GloVe` [PSM14] trained on Common Crawl data, embeds these sentences into ellipsoids A, B and C, with 5, 6, and 5 components respectively (under $\text{thresh}_e = 0.9$). Text T_A shares a similar degree of topical connections to both texts T_B and T_C, but neither sentence exactly matches the content of T_A. In fact, when compared directly using sim_{Elps} at $\beta = 1.0$, the pairs' similarities are quite close, with $\text{sim}_{\text{Elps}}(A, B) = 0.526$ and $\text{sim}_{\text{Elps}}(A, C) = 0.527$.

The goal of this exercise is to design changes in topic importance values in A for directed similarity that will increase A's similarity to either B or C. It's easy to see that A and B are similar in the sense that both mention music-related terms and talk about light effects, while A and C also share music as a theme, while specifically talking about a band and enjoyment. Our directed similarity query weighting will be designed using these qualitative assessments.

Using local topic interpretation under the GET view, we can break down the components of text A to see what main components were identified, and to subsequently design a "directed query" according to them. Table 17.5 shows the resulting topics.

Top. 0 (42.8%)		Top. 1 (22.2%)		Top. 2 (12.5%)		Top. 3 (8.8%)		Top. 4 (7.6%)	
band	0.77	effects	0.78	loved	0.64	band	0.45	light	0.721
concert	0.76	concert	0.43	great	0.52	concert	0.42	loved	0.299
great	0.67	band	0.39	better	0.40	light	0.27	concert	0.149
loved	0.61	light	0.33	effects	0.33	loved	0.12	band	0.140
better	0.61	better	0.27	light	0.23	great	0.07	effects	0.114
effects	0.52	great	0.18	concert	0.21	effects	0.05	great	0.035
light	0.47	loved	0.05	band	0.18	better	0.01	better	0.013

Table 17.5.: Topics in sample text T_A.

We can see that Topic 4 weights the word "light" most highly, with some margin before the next work ("loved"), thus this topic would be an interesting candidate for boosting the similarity of texts A and B. Topic 1 similarly has a high importance for the word "effects"; however, words like "concert" and "band" are also weighted relatively highly, making it less discriminative, as these words may also increase similarity of sentences A and C.

Similarly, to specifically boost the similarity of sentences A and C, Topic 0 may be a good choice, because it assigns the lowest weights to "effects" and "light", leading to a relative increase in importance of words that distinguish the similarity of A and C. Since Topic 0 also already has the highest overall importance (42.8% of explained variance), it can be re-weighted less strongly than Topic 4 would need to be in order to achieve the desired effect. Alternatively, lowering the importance of Topic 4 may lead to a relative increase in similarity of A and C over A and B, as it would diminish the influence of the word "light" in A.

Setting	$\text{sim}_{\text{Elps}}(A, B)$	$\text{sim}_{\text{Elps}}(A, C)$
default	0.526	0.527
Topic 4 +100%	**0.579**	0.509
Topic 1 +100%	<u>0.504</u>	0.490
Topic 0 +50%	0.510	**0.562**
Topic 4 -100%	0.464	<u>0.522</u>

Table 17.6.: Sentence similarities under different directed search settings.

The similarity scores under each of these settings are listed in Table 17.6. We can see the different alterations have the desired effects in increasing relative similarity of either sentence pair, with an increase of 100% (doubling the importance score) in Topic 4 being

most effective for the pair (A, B) and a 50% increase of Topic 0 effectively boosting the similarity of the pair (A, C). The alternative approaches of respectively boosting Topic 1 and deleting Topic 4 also lead to a *relative increase* in similarity of each pair over the other, but lead to an *overall decrease* in similarity of A to either sentence, which may lead to worse results if this were a search query rather than a direct comparison.

17.3. Re-weighting vectors for search

We illustrate our vector re-weighting scheme against Inquire [Par+17], a live system of 10 million indexed sentence vectors, computed using simple averages of Common Crawl `GloVe` vectors. In this setting, only the query vector is altered, the database vectors remain static as simple un-weighted averages. Possible query weights are determined through word-topic scores in affine GET, as outlined in Section 16.3. Queries may be run either as a simple un-weighted vector average, i.e. the default Inquire setting, or a weighted average according to the selected topic.

To design a simple and intuitive query refinement interaction, we opt to visualize the GET topics by displaying the query to the user, and re-sizing words according to their importance score in the currently selected topic. We also visualize the topic importance values as a simple unlabelled histogram. There are two possible interaction methods for a user of this method: she can either select a topic directly by selecting the corresponding bar in the topic importance histogram, or she can directly choose a word from the sentence, and the topic that most highly scores this word is automatically selected for querying.

We first apply the mechanism on the same sample sentence for which we demonstrated directed subspace similarity. The top-10 results for the un-weighted vector average and a re-weighting that selects a topic highly prioritizing "light" are shown in Tables 17.7 and 17.8. Note that this method uses affine subspaces, thus the topics for the text are not identical to those reported in the previous section. The topic was selected by simply clicking the word "light" in the query text, and the corresponding topic weights were applied automatically.

We can see that the re-weighting strongly affects the search results towards sentences that mention light and related terms, yet it does not totally disregard the music-related terms of the query. While in the un-weighted query, none of the top 10 result sentences mention lights, in the weighted query, results 2 and 8 contain particularly strong mentions of both topics. At the same time, it appears that the re-weighting may be a bit stronger than necessary in this example, as many search results in the weighted setting seem to talk *only* about the light topic. This shortcoming could be addressed in different ways, such as through an additional dampening factor that allows for a better

	the concert was great and i loved the band but the light effects could have been better
0.89	but anyways the concert was really amazing. great songs.
0.88	But amazing bands with their amazing music, and really nice people made it an excellent night.
0.88	I like concert band more.
0.88	This band plays his music along with alot of other really good music.
0.88	That sure was an awesome concert, good music, good friends.
0.88	im so stoked for that band, like amazing lyrics and music and great people.
0.87	Also, the fact that there was so much music - vocal music, and live music for the dancing afterwards - really set the mood.
0.87	I love the big band sound. :)
0.87	I love the big band sound.
0.87	Their harmonies on the group songs were lovely, and I can't imagine ever going to a better vocal concert. ...

Table 17.7.: Sample averaged query search results. Grey words are OOV / stopwords.

	the concert was great and i loved the band but the light effects could have been better
0.90	good light music.
0.86	The concert was really good except the light was in my eyes and the light was shining off Craig's guitar onto the music.
0.86	The light loved her.
0.86	I miss the light.
0.86	. I like the light.
0.86	The light was great though.
0.85	well, not light.
0.85	but this time with a blacklight, flourescent decorations, a strobe light, and loud dance music.
0.85	I saw a bright light.
0.85	The awesome light.

Table 17.8.: Inquire search results for weighted example query. Query word size corresponds to word score in the weighted average embedding.

Q1	so lets just say im feeling a bit overwhelmed right now with school work
0.98	so lets just say i'm feeling a little overwhelmed right now with school work.
0.96	i guess i'm just feeling overwhelmed right now. so much so that i do nothing
0.96	i guess im just stressed out, and school is becoming so ridiculous that i [...]
0.96	[...] I'm tired of thinking about work so I'm gonna go chill....I feel stressed[...]
0.95	[...] feel "blah". and i'll admit that i'm nervous about school tomorrow
Q2	so lets just say im feeling a bit overwhelmed right now with school work
0.96	i'm feeling so overwhelmed.
0.95	I'm just feeling so overwhelmed and dazed.
0.93	I just feel so overwhelmed.
0.93	I feel so overwhelmed.
0.93	I feel so overwhelmed.
Q3	so lets just say im feeling a bit overwhelmed right now with school work
0.94	im kinda anxious to start school – i just wish school was here.
0.94	so, im going to school.
0.94	im going to school.
0.93	im actually going to school!!
0.93	well im doing stupid school and today im going to a really bad school play...

Table 17.9.: Top-5 query results for different selected topic re-weightings in Inquire.

trade-off between un-weighted and weighted averaging. Nevertheless, the method finds a useful re-weighting configuration out-of-the-box, with no need for the user to manually set weights for each input word. This simple query refinement allows for fast iteration and exploration of a large corpus with little additional effort.

We finally showcase another sample query, which we refine in two different way to highlight the differences in search results yielded by different parametrizations. These results can be seen in Table 17.9.

Topics were selecting by prioritizing the words "overwhelmed" and "school" respectively. We can see in each case that the re-weighting is subjectively effective and leads to usefully refined search results. Again, we observe that it may be beneficial to reduce the overall degree of re-weighting in practical settings. Here, we opt to present the results directly, in order to give a better idea of the raw effect of GET based re-weighting.

18. Discussion

We outline GET, an interpretable view of DGE embeddings that considers ellipsoid axes as topics, and importance values as topic scores. We show how to interpret these topics through local and global vocabulary characterization, and based on this we introduce and demonstrate two query refinement methods.

The methods introduced here are relatively simple to apply in practical settings. Although we only provide a brief qualitative exploration here, we find that these methods can be usefully deployed in practical applications. GET yields topics that provide qualitatively good semantic characterizations of text content, and could be used in many situations where topic models are currently being utilized. While we cannot argue for better quantitative performance of DGE/GET compared to other topic models (other than the results on sentence semantics presented in Part I of this work), we note that these methods have practical features that make them versatile approaches.

Topics in GET are locally computed and aim to ideally represent individual documents, rather than a global corpus. Not only does this make it possible to reinforce or neglect specific aspects of a document with precision when performing searches, but it also allows for the representation of topics that may be very specific and therefore not well-represented over the corpus as a whole. In addition, our method requires no corpus-specific training. It requires only a trained word-vector model, which are abundantly available in high quality across different domains today.

The method also applies well to both sentences and documents, and allows direct similarity comparison of both, and similarity to individual words is also well-defined. This is technically possible in other topic models; however, since common topic models (e.g. LDA, HDP) tend to use sparse vectors to represent text, sentences and words may not be equally well-represented by them in many cases.

Future work in this area may focus on more objective evaluation of the concepts introduced here, for example investigating the impact of directed search on a system's perceived and/or measured usefulness. This could be investigated in a variety of ways, such as a study measuring the the time users take to solve certain information retrieval tasks, investigations into the perceived usefulness of a system as reported by users, or a comparison of the subjective quality of results obtained by users, as measured by some expert judges. Other avenues of research could include an evaluation of GET as a topic model, although it is not immediately obvious how the model should be directly

compared to methods such as LDA or HDP.

In effect, this view of DGE marries desirable aspects of word-vector modelling, sentence embedding and topic models, while also being an interpretable model that can be usefully applied in interactive human-facing settings.

Part IV.

Conclusion

19. Conclusion

We explore the application of Grassmannian methods to the domain of text representation, giving special attention to sentence embedding and semantic textual similarity.

We first introduce DGE, a new sentence embedding model that builds on the subspace sentence embedding model (GSE) introduced by Mu et al. [MBV16]. DGE improves on GSE in a variety of ways, ranging from simple improvements (subword features and frequency-weighting) to more substantial changes (dynamic space sizing and ellipsoid features). Our model outperforms both GSE and many other sentence embedding models across many of the SemEval STS tasks, lending further weight to the insight that shallow models can achieve strong performance on semantic similarity problems [Wie+15][ALM17].

We also contribute an investigation into, and extension of, the approximate nearest subspace search technique proposed by Iwamura et al. [IKK16]. We find that the approach is to some extent applicable to the text domain; however, the performance achieved by the "vanilla" implementation of the authors' method is insufficient for practical application. We implement the authors' approach and extend it using our own additions, in order to improve the method's performance; however, the method ultimately appears to remain too slow for practical large-scale applications. In the scope of this research, we also propose `Schuhlöffel`, a method for tuning subspace basis matrices on the Stiefel manifold, which yields mixed results in this setting, but may serve as a valuable basis for further work.

Finally, we explore practical and qualitative aspects of the Grassmannian approach to text representation. We introduce GET, a topic-model view of DGE that allows for a variety of human-facing downstream applications. The qualitative exploration we provide shows that DGE representations can be intuitively interpreted and used to conduct directional semantic searches, enabling interactive methods for corpus exploration and information retrieval.

Through these contributions, we find that Grassmannian methods are both highly performant and practically useful in the domain of textual data. Methods such as those introduced here open the door for new means of leveraging structure in word-vector spaces. They remove the limitations of using a purely vector-based view by employing more sophisticated representations and mathematical techniques, yet the representations and algorithms can still be visually and intuitively understood, and

are easy to interpret. We hope that this work can inform future research into new applications of Grassmannian methods in natural language processing, information retrieval, and related fields.

List of Figures

List of Tables

Bibliography

[Agi+12] E. Agirre, M. Diab, D. Cer, and A. Gonzalez-Agirre. "Semeval-2012 task 6: A pilot on semantic textual similarity." In: *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics. 2012, pp. 385–393.

[Agi+13] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo. "* SEM 2013 shared task: Semantic textual similarity." In: *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Vol. 1. 2013, pp. 32–43.

[Agi+14] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe. "Semeval-2014 task 10: Multilingual semantic textual similarity." In: *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*. 2014, pp. 81–91.

[Agi+15] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, et al. "Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability." In: *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 2015, pp. 252–263.

[Agi+16] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe. "Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation." In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. 2016, pp. 497–511.

[ALM17] S. Arora, Y. Liang, and T. Ma. "A Simple But Tough-To-Beat Baseline for sentence embeddings." In: *Proceedings of ICLR 2017* (2017).

[AMS04] P.-A. Absil, R. Mahony, and R. Sepulchre. "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation." In: *Acta Applicandae Mathematica* 80.2 (2004), pp. 199–220.

[Ark+14] N. Arkani-hamed, J. Bourjaily, F. Cachazo, A. Goncharov, and A. Postnikov. "Scattering Amplitudes and the Positive Grassmannian." In: *arXiv preprint arXiv:1212.5605* (2014). arXiv: arXiv:1212.5605v2.

[Aro] S. Arora. *Linear algebraic structure of word meanings*.

[Asi85] D. Asimov. "The grand tour: a tool for viewing multidimensional data." In: *SIAM journal on scientific and statistical computing* 6.1 (1985), pp. 128–143.

[Ast+15] R. F. Astudillo, S. Amir, W. Lin, M. Silva, and I. Trancoso. "Learning Word Representations from Scarce and Noisy Data with Embedding Sub-spaces." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* 1 (2015), pp. 1074–1084.

[Bat+16] K. Batmanghelich, A. Saeedi, K. Narasimhan, and S. Gershman. "Nonparametric Spherical Topic Modeling with Word Embeddings." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* 2 (2016), pp. 537–542. DOI: 10.18653/v1/P16-2087. arXiv: 1604.00126.

[BHZ07] R. Basri, T. Hassner, and L. Zelnik-Manor. "Approximate Nearest Subspace Search with Applications to Pattern Recognition." In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. June 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383201.

[BHZ09] R. Basri, T. Hassner, and L. Zelnik-Manor. "A general framework for Approximate Nearest Subspace search." In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. Sept. 2009, pp. 109–116. DOI: 10.1109/ICCVW.2009.5457710.

[BHZ11] R. Basri, T. Hassner, and L. Zelnik-Manor. "Approximate nearest subspace search." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.2 (2011), pp. 266–278. ISSN: 01628828. DOI: 10.1109/TPAMI.2010.110.

[BL06] D. M. Blei and J. D. Lafferty. "Dynamic topic models." In: *Proceedings of the 23rd international conference on Machine learning - ICML '06* (2006), pp. 113–120. DOI: 10.1145/1143844.1143859.

[BN13] L. Boytsov and B. Naidan. "Engineering Efficient and Effective Non-metric Space Library." In: *Similarity Search and Applications - 6th International Conference, {SISAP} 2013, {A} Coruña, Spain, October 2-4, 2013, Proceedings*. 2013, pp. 280–293. DOI: 10.1007/978-3-642-41062-8_28.

[BNJ03] D. Blei, A. Ng, and M. I. Jordan. "Latent dirichlet allocation." In: *The Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[Boj+17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. "Enriching Word Vectors with Subword Information." In: *arXiv preprint arXiv:1607.04606* (2017). ISSN: 10450823. DOI: 1511.09249v1. arXiv: 1607.04606.

[Bol+16] T. Bolukbasi, K.-w. Chang, J. Zou, V. Saligrama, and A. Kalai. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings." In: *Advances in Neural Information Processing Systems* (2016), pp. 4349–4357.

[Dat+04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. "Locality-sensitive hashing scheme based on p-stable distributions." In: *Proceedings of the twentieth annual symposium on Computational geometry* (2004), pp. 253–262. ISSN: 0899-7667. DOI: 10.1145/997817.997857. arXiv: arXiv:1011.1669v3.

[DDL90] S. Deerwester, S. Dumais, and T. Landauer. "Indexing by latent semantic analysis." In: *Journal of the American society for information science* 41.6 (1990).

[DHS11] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159. ISSN: 15324435. DOI: 10.1109/CDC.2012.6426698. arXiv: arXiv:1103.4296v1.

[DZD15] R. Das, M. Zaheer, and C. Dyer. "Gaussian LDA for Topic Models with Word Embeddings." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* 1 (2015), pp. 795–804.

[EA98] A. Edelman and T. As. "The geometry of algorithms with orthogonality constraints." In: *SIAM journal on Matrix Analysis and Applications* 20.2 (1998), pp. 303–353.

[Fuj02] K. Fujii. "Introduction to Grassmann Manifolds and Quantum Computation." In: *Journal of Applied Mathematics* (2002), pp. 371–405. arXiv: 0103011v5 [arXiv:quant-ph].

[Gan+17] Z. Gan, Y. Pu, R. Henao, C. Li, X. He, and L. Carin. "Learning Generic Sentence Representations Using Convolutional Neural Networks." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), pp. 2390–2400. arXiv: arXiv:1611.07897v2.

[GBV17] H. Gong, S. Bhat, and P. Viswanath. "Geometry of Compositionality." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)* (2017), pp. 3202–3208. arXiv: 1611.09799.

[Gon+17] H. Gong, J. Mu, S. Bhat, and P. Viswanath. "Prepositions in Context." In: *arXiv preprint arXiv:1702.01466* (2017). arXiv: 1702.01466.

[Guz15] F. Guzm. "Pairwise Neural Machine Translation Evaluation." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* 1 (2015), pp. 805–814.

[GV12] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2012. ISBN: 9781421407944.

[GVC13] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. "PPDB: The paraphrase database." In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013, pp. 758–764.

[Har+14] M. T. Harandi, M. Salzmann, S. Jayasumana, R. Hartley, and H. Li. "Expanding the family of Grassmannian kernels: An embedding perspective." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8695 LNCS. PART 7. 2014, pp. 408–423. ISBN: 9783319105833. DOI: 10.1007/978-3-319-10584-0_27. arXiv: arXiv:1407.1123v1.

[Har54] Z. S. Harris. "Distributional Structure." In: *WORD* 10.2-3 (1954), pp. 146–162. ISSN: 0043-7956. DOI: 10.1080/00437956.1954.11659520.

[HBS12] J. He, L. Balzano, and A. Szlam. "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video." In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 1568–1575. DOI: 10.1109/CVPR.2012.6247848.

[HCK16] F. Hill, K. Cho, and A. Korhonen. "Learning Distributed Representations of Sentences from Unlabelled Data." In: *arXiv preprint arXiv:1602.03483* (2016). arXiv: arXiv:1602.03483v1.

[HL08] J. Hamm and D. D. Lee. "Grassmann Discriminant Analysis : a Unifying View on Subspace-Based Learning." In: *Proceedings of the 25th international conference on Machine learning - ICML '08* 454.July (2008), pp. 376–383. ISSN: 01678655. DOI: 10.1145/1390156.1390204. arXiv: arXiv:1202.3819v2.

[Hu14] H. Hu. "Face Recognition With Image Sets Using Locally Grassmannian Discriminant Analysis." In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.9 (Sept. 2014), pp. 1461–1474. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2014.2309834.

[IKK16] M. Iwamura, M. Konishi, and K. Kise. "Scalable Solution for Approximate Nearest Subspace Search." In: *arXiv preprint arXiv:1603.08810* (2016). arXiv: arXiv:1603.08810v1.

[KA16] A. Kumar and J. Araki. "Incorporating Relational Knowledge into Word Representations using Subspace Regularization." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* 2 (2016), pp. 506–511.

[KBR16] T. Kenter, A. Borisov, and M. de Rijke. "Siamese CBOW: Optimizing Word Embeddings for Sentence Representations." In: *arXiv preprint arXiv:1606.04640* (2016). arXiv: 1606.04640.

[Kir+15] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. "Skip-Thought Vectors." In: *Advances in neural information processing systems* (2015), pp. 3294–3302. ISSN: 1098-6596. arXiv: 1506.06726.

[Lev+15] O. Levy, S. Remus, C. Biemann, and I. Dagan. "Do Supervised Distributional Methods Really Learn Lexical Inference Relations?" In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2015), pp. 970–976.

[LG14] O. Levy and Y. Goldberg. "Linguistic Regularities in Sparse and Explicit Word Representations." In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning* (2014), pp. 171–180. DOI: 10.3115/v1/W14-1618.

[Lin+17] Z. Lin, M. Feng, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. "A Structured Self-attentive Sentence Embedding." In: *arXiv preprint arXiv:1703.03130* (2017). arXiv: arXiv:1703.03130v1.

[Liu+12] W. Liu, J. Wang, Y. Mu, S. Kumar, and S.-F. Chang. "Compact Hyperplane Hashing with Bilinear Functions." In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)* (2012), pp. 17–24.

[Liu+16] S. Liu, P. T. Bremer, J. J. Jayaraman, B. Wang, B. Summa, and V. Pascucci. "The Grassmannian Atlas: A General Framework for Exploring Linear Projections of High-Dimensional Data." In: *Computer Graphics Forum* 35.3 (2016), pp. 1–10. ISSN: 14678659. DOI: 10.1111/cgf.12876.

[LMC14] Q. Le, T. Mikolov, and T. G. Com. "Distributed Representations of Sentences and Documents." In: *International Conference on Machine Learning* (2014), pp. 1188–1196.

[LWY16] L.-h. Lim, K. S.-w. Wong, and K. Ye. "Numerical Algorithms on the Affine Grassmannian." In: *arXiv preprint arXiv:1607.01833* (2016), pp. 1–27. arXiv: 1607.01833.

[Mar+14] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, R. Zamparelli, et al. "A SICK cure for the evaluation of compositional distributional semantic models." In: *Proceedings of LREC 2014*. 2014, pp. 216–223.

[MBV16] J. Mu, S. Bhat, and P. Viswanath. "Geometry of Polysemy." In: *arXiv preprint arXiv:1610.07569* (2016). arXiv: 1610.07569.

[MBV17] J. Mu, S. Bhat, and P. Viswanath. "Representing Sentences as Low-Rank Subspaces." In: *arXiv preprint arXiv:1704.05358* (2017). DOI: 10.18653/v1/P17-2099. arXiv: 1704.05358.

[MC15] S. Mahadevan and S. Chandar. "Reasoning about Linguistic Regularities in Word Embeddings using Matrix Manifolds." In: *arXiv preprint arXiv:1507.07636* (2015). arXiv: 1507.07636.

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space." In: *arXiv preprint arXiv:1301.3781* (Jan. 2013). arXiv: 1301.3781.

[Mik+17] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. "Advances in Pre-Training Distributed Word Representations." In: *arXiv preprint arXiv:1712.09405* (2017). arXiv: 1712.09405.

[Moo16] C. E. Moody. "Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec." In: *arXiv preprint arXiv:1605.02019* (2016). arXiv: 1605.02019.

[MRS08] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. 2008.

[MY16] Y. A. Malkov and D. A. Yashunin. "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs." In: *arXiv preprint arXiv:1603.09320* (2016). arXiv: 1603.09320.

[MYZ13] T. Mikolov, W.-t. Yih, and G. Zweig. "Linguistic Regularities in Continuous Space Word Representations." In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013), pp. 746–751.

[Pal+16] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, and X. Song. "Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval." In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* (2016), pp. 694–707. arXiv: arXiv:1502.06922v3.

[Par+17] P. Paredes, A. R. Ferreira, C. Schillaci, G. Yoo, P. Karashchuk, D. Xing, C. Cheshire, and J. Canny. "Inquire: Large-scale Early Insight Discovery for Qualitative Research." In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (2017), pp. 1562–1575.

[Pas+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch." In: *31st Conference on Neural Information Processing Systems (NIPS 2017)* (2017).

[PGJ17] M. Pagliardini, P. Gupta, and M. Jaggi. "Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features." In: *arXiv preprint arXiv:1703.02507* (2017). arXiv: arXiv:1703.02507v2.

[PRC15] T. Polajnar, L. Rimell, and S. Clark. "An Exploration of Discourse-Based Sentence Spaces for Compositional Distributional Semantics." In: *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics* (2015), pp. 1–11.

[PSM14] J. Pennington, R. Socher, and C. Manning. "Glove: Global vectors for word representation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[RES16] S. Rothe, S. Ebert, and H. Schütze. "Ultradense Word Embeddings by Orthogonal Transformation." In: *arXiv preprint arXiv:1602.07572* (2016).

[RS16] S. Rothe and H. Sc. "Word Embedding Calculus in Meaningful Ultradense Subspaces." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* 2 (2016), pp. 512–517.

[Sch92] H. Schütze. "Dimensions of meaning." In: *Proceedings Supercomputing '92* (1992). DOI: 10.1109/SUPERC.1992.236684.

[Shl14] J. Shlens. "A Tutorial on Principal Component Analysis." In: *arXiv preprint arXiv:1404.1100* (2014).

[Soc+11a] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection." In: *Advances in neural information processing systems* (2011), pp. 801–809.

[Soc+11b] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions." In: *Proceedings of the conference on empirical methods in natural language processing* (2011), pp. 151–161.

[Soc+12] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. "Semantic Compositionality through Recursive Matrix-Vector Spaces." In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning* (2012), pp. 1201–1211.

[Sut14] I. Sutskever. "Sequence to Sequence Learning with Neural Networks." In: *Advances in neural information processing systems* (2014), pp. 3104–3112. arXiv: arXiv:1409.3215v3.

[SZW14] J. Sun, Y. Zhang, and J. Wright. "Efficient Point-to-Subspace Query in l1 with Application to Robust Object Instance Recognition." In: *SIAM Journal on Imaging Sciences* 7.4 (2014), pp. 2105–2138. ISSN: 1936-4954. DOI: 10.1137/130936166. arXiv: arXiv:1208.0432v2.

[Teh+06] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. "Hierarchical Dirichlet processes." In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1566–1581. ISSN: 01621459. DOI: 10.1198/016214506000000302. arXiv: arXiv:1210.6738v2.

[Tsv+15] Y. Tsvetkov, M. Faruqui, W. Ling, G. Lample, and C. Dyer. "Evaluation of Word Vector Representations by Subspace Alignment." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* September (2015), pp. 2049–2054.

[VJG14] S. Vijayanarasimhan, P. Jain, and K. Grauman. "Hashing hyperplane queries to near points with applications to large-scale active learning." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.2 (2014), pp. 276–288. ISSN: 01628828. DOI: 10.1109/TPAMI.2013.121.

[Wan+13] X. Wang, S. Atev, J. Wright, and G. Lerman. "Fast subspace search via grassmannian based hashing." In: *Computer Vision (ICCV), 2013 IEEE International Conference on* (2013), pp. 2776–2783. ISSN: 1550-5499. DOI: 10.1109/ICCV.2013.345.

[Wie+15] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. "Towards universal paraphrastic sentence embeddings." In: *arXiv preprint arXiv:1511.08198* (2015). arXiv: arXiv:1511.08198v3.

[WY13] Z. Wen and W. Yin. "A feasible method for optimization with orthogonality constraints." In: *Mathematical Programming* 142.1-2 (2013), pp. 397–434.

[Yan+16] Z. Yan, N. Duan, J. Bao, P. Chen, and M. Zhou. "DocChat : An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2016), pp. 516–525.

[YL16] K. Ye and L.-h. Lim. "Schubert varieties and distances between subspaces of different dimensions." In: *arXiv:1407.0900v3 [math.NA]* (2016), pp. 1–20. ISSN: 10957162. DOI: 10.1137/15M1054201. arXiv: 1407.0900.

[YS16] Y. Yaghoobzadeh and H. Schütze. "Intrinsic Subspace Evaluation of Word Embedding Representations." In: *arXiv preprint arXiv:1606.07902* (2016). arXiv: arXiv:1606.07902v1.

[Zhu+15] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.