DENSITY REPRESENTATIONS FOR WORDS AND HIERARCHICAL DATA

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Praphruetpong (Ben) Athiwaratkun

May 2019

DENSITY REPRESENTATIONS FOR WORDS AND HIERARCHICAL DATA

Praphruetpong (Ben) Athiwaratkun, Ph.D.

Cornell University 2019

We demonstrate the benefits of probabilistic representations due to their expressiveness which allows for flexible representations, their ability of capture uncertainty, and their interpretable geometric structures that are suitable for modeling hierarchical data. We show that multimodal densities can be effectively used to represent words in natural text, capturing possibly multiple meanings and their nuances. Probability densities also have natural geometric structures which can be used to represent hierarchies among entities through the concept of encapsulation; that is, dispersed distributions are generic entities that encompass more specific ones. We show an effective approach to train such density embeddings by penalizing order violations which are defined through on asymmetric divergences of probability densities.

ACKNOWLEDGEMENTS

I would like to thank my advisor Andrew Gordon Wilson for all his support. I feel very fortunate to be advised by Andrew, who has given me a lot of freedom to explore many different research directions. He has inspired me to strive for creativity, shown me the value of clear exposition, taught me invaluable research perspectives.

I thank everyone I have collaborated with – I feel lucky to have the opportunities to interact with such intelligent and driven individuals over the course of my PhD. I thank Austin Seppala for giving me the strength and motivation to complete this PhD journey and I thank my parents and siblings for their continuous support in any way possible since my childhood.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Learning to represent data in a low-level manifold has long been the goal of machine learning. In the past decade, deep learning has allowed us to discover such meaningful representations and has fueled the success of many fields such as computer vision, natural language processing, to name a few. However, most representations lack the uncertainty information that can be crucial for modeling realistic data. For instance, an occluded image of a handwritten digit can turn out to be many possible numbers. Failing to capture the uncertainty discards necessary information and leads to inaccurate representation. Recently, there have been increasing interests in incorporating uncertainty. For instance, Oh et al. [2019] proposes using a probabilistic representation to encode uncertainty due to occlusion or blurriness of images and found that modeling uncertainty is key to downstream tasks. Wan et al. [2018] proposes using Gaussian-mixture based loss function for image classification, which demonstrates high classification performance and exhibits more accurate feature distribution. In this thesis, we present the results of our investigation on modeling words in natural languages with probability densities. In particular, the density representations allow us to capture nuances and possible multiplicities of meanings.

Probabilistic representations also allow us to learn geometric structures that reflect relationships among data. Structured representations help support relational reasoning among entities and improve generalization through inductive biases. Knowledge on entities' relationships also help address *explainability* as they illuminate pieces of the "black boxes" that are most of modern deep learning models. Explicit modeling of relationships among variables was previously studied in many

sub-fields of machine learning such as graphical models [Koller and Friedman, 2009], causal reasoning, etc. For instance, in graphical models, different structures of conditional independence among random variables specify entities' relationships and give rise to various complex joint distributions. Many deep learning approaches, however, assume minimal generative and relational assumptions; for instance, discriminative models do not assume the underlying data generation process or model-free reinforcement learning methods do not attempt to learn environment's dynamics. These approaches are powerful but require massive amount of data and iterations to train in order to learn well, due to the lack of *relational inductive biases* [Battaglia et al., 2018]. Recently, there is a resurgence of the "model-based" approach to help address high sample complexity by encode relationships and environment dynamics [Chua et al., 2018]. In fact, a recent model by Zambaldi et al. [2019] introduces relational reasoning over structured representations to achieve state-of-the-art performance of challenging tasks such as StarCraft II mini-games where the learning efficiency far exceeds methods that do not incorporate relational knowledge. This approach is akin to the process human learns: we perform reasoning based on relations and understanding of underlying generative processes and rely less on the bulk of data needed to generalize to unseen concepts. We believe that structured representations that reflect relations are necessary ingredients to human-level intelligence. Therefore, we believe it is important to learn building blocks such as entity relations that can be composed and used to reason about unseen scenarios.

We argue that probabilistic representations are efficient at learning relationships. In fact, we show that relationships among entities represented by distributions can naturally be reflected through encapsulation. Formally, we say that an entity a entails b, denoted by $a \preceq b$, if any a belongs to b. A generic entity such as

'animal' is a broad distribution compared to more specific entities that belongs to the category 'animal' such as 'dog' and 'cat'. In this case, 'dog' and 'cat' entail 'animal' and are peaked distributions that are encapsulated by 'animal' distribution. These relationships can emerge through unsupervised training on text corpus where we provide qualitative and quantitative evidence in Chapter 3. We also learn these structures explicitly through graph data, which is the subject of Chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

Probabilistic framework has demonstrated its effectiveness in many applications including in computer vision [Oh et al., 2019, Wan et al., 2018], natural language processing [Fortunato et al., 2017a, Gan et al., 2017] and reinforcement learning [Azizzadenesheli et al., 2018]. In this thesis, we show two main approaches that use the probabilistic perspective to learn unsupervised word embeddings (Chapter 3 and 4) and representations for hierarchical data (Chapter 5).

The literature of word embeddings has its own rich history. Popular word embeddings are pioneered by Mikolov et al. [2013a,b] and Pennington et al. [2014] who propose efficient training to word semantics from a large amount of text data (summarized in Section 2.1.1). An improved method FastText [Bojanowski et al., 2016] incorporates character information to increase model's ability to estimate the semantics of out-of-vocabulary and rare words [Bojanowski et al., 2016] (Section 2.1.2). Other methods extend beyond single vector representation to learn multiple meanings for each word by using *multi-prototype* embeddings (Section 2.1.3). Our probabilistic model called *multimodal word distribution*, detailed in Chapter 3 and its character-level version 4, unify all these approaches with elegant probabilistic representation that can reflect nuances and possibly multiple meanings.

In Chapter 5, we focus on using probabilistic representation to model hierarchical data such as graphs and Section 2.2 aims to provide necessary background and related work. Our work in Chapter 5 proposes a probabilistic method called *density order embeddings* (DOE) to effectively model data such as WordNet graph or hierarchies of a visual-semantic space. Probabilistic representation is a rapidly growing field, and recent works demonstrate applicability in many areas. For instance, it can

be used as a recommendation system through link prediction in user-product graph [Li et al., 2019] or to predict sentence-level textual entailment [Bowman et al., 2015, Marelli et al., 2014]. We summarize relevant probabilistic methods in the literature in Section 2.2.2. Besides probabilistic embeddings, modeling hierarchical data are studied in other nascent fields of hyperbolic embeddings and graph neural networks, which we outline and draw comparison in Section 2.2.3 and 2.2.4 respectively.

2.1 Word Embeddings

To model language, we must represent words. We can imagine representing every word with a binary one-hot vector corresponding to a dictionary position. But such a representation contains no valuable semantic information: distances between word vectors represent only differences in alphabetic ordering. Modern approaches, by contrast, learn to map words with similar meanings to nearby points in a vector space [Mikolov et al., 2013a,b, Pennington et al., 2014], from large datasets such as Wikipedia. These learned word embeddings have become ubiquitous in predictive tasks.

We describe multiple approaches to learn such embeddings in the literature, starting from the traditional word-level embeddings to character-level embeddings, as well as extensions to multi-sense embeddings. We provide background materials, Gaussian embeddings, and explain how we extend this previous work to Chapter 3 and 4 connect all these approaches into a unifying framework.

2.1.1 Word-Level Embeddings

In the past decade, there has been an explosion of interest in word vector representations. WORD2VEC, arguably the most popular word embedding, uses continuous bag of words and skip-gram models, in conjunction with negative sampling for efficient conditional probability estimation [Mikolov et al., 2013a,b]. The objective is to maximize the likelihood of observing nearby words w_c given current word w in contrast to the likelihood of observing other random words n. The loss we seek to optimize is:

$$\ell(s(w, w_c)) + \sum_{n \in \mathcal{N}_c} \ell(-s(w, n)) \tag{2.1}$$

where ℓ is a logistic loss function $\ell(x) = 1 + e^{-x}$, $\mathcal{N}_c$ is a set of negative samples drawn from the vocabulary, and $s(x, y)$ denotes the similarity score between word x and word y. In this case, $s(x, y)$ is simply the dot product $u_x \cdot u_y$ where u_z is a vector corresponding to word z.

Other popular approaches use feedforward [Bengio et al., 2003] and recurrent neural network language models [Collobert and Weston, 2008, Mikolov et al., 2010, 2011b] to predict missing words in sentences, producing hidden layers that can act as word embeddings that encode semantic information. They employ conditional probability estimation techniques, including hierarchical softmax [Mikolov et al., 2011a, Mnih and Hinton, 2008, Morin and Bengio, 2005] and noise contrastive estimation [Gutmann and Hyvärinen, 2012].

A different approach to learning word embeddings is through factorization of word co-occurrence matrices such as GLOVE embeddings [Pennington et al., 2014]. The matrix factorization approach has been shown to have an implicit connection with skip-gram and negative sampling [Baer et al., 2018, Landgraf and Bellay, 2017,

Levy and Goldberg, 2014]. Bayesian matrix factorization where row and columns are modeled as Gaussians has been explored in Salakhutdinov and Mnih [2008] and provides a different probabilistic perspective of word embeddings.

2.1.2 Character-Level Embeddings

There are multiple approaches that incorporate character information into embeddings. A prominent one, FastText [Bojanowski et al., 2016], learns character n-gram vector representations and define a word representation as the average of its character n-gram components as well as its word-level vector. That is,

$$\mu_w = \frac{1}{|NG_w| + 1} \left(v_w + \sum_{g \in NG_w} z_g \right) \tag{2.2}$$

where v_w is a word-level vector for word w, NG_w is a set of character n-grams for word w and z_g is a vector for an n-gram g. The training process is almost identical to that of word2vec with the same loss function and word sampling process. We find that such representation results in automatic assignment of semantics to character n-grams, allowing better representation of rare and unseen words (see details in Chapter 4.3.5).

This flexibility of character-level embeddings has valuable applications in many end-tasks such as language modeling [Kim et al., 2016], named entity recognition [Kuru et al., 2016], and machine translation [Lee et al., 2017, Zhao and Zhang, 2016], where unseen words are frequent and proper handling of these words can greatly improve the performance. Other character-level embedding methods involve explicitly splitting words into multiple disjoint subwords through BPE encoding [Gage, 1994] and learn those subword vectors directly in end tasks [Sennrich et al., 2016]. Another line of work attempts to learn a method to compose subword

vectors to a final word vector [Kim et al., 2018] as opposed to simple averaging in FastText [Bojanowski et al., 2016].

Applications of character-level embeddings is wide-ranging. Many downstream tasks have adopted character embeddings into their models for improved performance on rare words and the ability to generalize to unseen words. FastText, for example, is a crucial component in a state-of-the-art system such as unsupervised machine translation [Lample et al., 2017, 2018].

2.1.3 Multi-Sense Embeddings

Recent work has also proposed deterministic embeddings that can capture polysemies, for example through a cluster centroid of context vectors [Huang et al., 2012], or an adapted skip-gram model with an EM algorithm to learn multiple latent representations per word [Tian et al., 2014]. [Chen et al., 2014] learns word vectors and sense vectors separately through skip-gram approach [Mikolov et al., 2013a] and uses WordNet synsets [Miller, 1995] as external knowledge in order to learn multiple senses. Neelakantan et al. [2014] also extends skip-gram with multiple prototype embeddings where the number of senses per word is determined by a non-parametric approach. Liu et al. [2015] learns topical embeddings based on latent topic models where each word is associated with multiple topics. Another related work by Nalisnick and Ravi [2015] models embeddings in infinite-dimensional space where each embedding can gradually represent incremental word sense if complex meanings are observed.

2.1.4 Gaussian Embeddings

Vilnis and McCallum [2015] was the first to propose using probability densities as word embeddings. In particular, each word is modeled as a Gaussian distribution, where the mean vector represents the semantics and the covariance describes the uncertainty or nuances in the meanings. These embeddings are trained on a natural text corpus by maximizing the similarity between words that are in the same local context of sentences. Given a word w with a true context word c_p and a randomly sampled word c_n (negative context), Gaussian embeddings are learned by minimizing the rank objective in Equation 2.3, which pushes the similarity of the true context pair $E(w, c_p)$ above that of the negative context pair $E(w, c_n)$ by a margin m.

$$L_m(w, c_p, c_n) = \max(0, m - E(w, c_p) + E(w, c_n)) \tag{2.3}$$

The similarity score $E(u, v)$ for words u, v can be either

$$E(u, v) = -\mathrm{KL}(f_u, f_v) \tag{2.4}$$

or

$$E(u, v) = \log\langle f_u, f_u\rangle_{L_2} \tag{2.5}$$

where f_u, f_v are the distributions of words u and v, respectively.

The Gaussian word embeddings contain rich semantic information and performs competitively in many word similarity benchmarks. However, words with multiple meanings result in overly diffuse probability densities, which inaccurately reflect the semantic uncertainties. In Chapter 3, we provide details of our multimodal word distribution model, a method that represents words with Gaussian mixture probability densities to allow for better estimation of uncertainty and multiple

meanings. Chapter 4 incorporate character information into the multimodal word distribution model, which we call *probabilistic* FastText.

2.2 Modeling Hierarchical Data

In the previous section, we have described approaches for unsupervised word embeddings which are learned from text corpora. These embeddings exhibit semantic traits where similar or related words are often close to each other in the embedding space.

However, there are relationships among words or entities that are not effectively learned through natural text. In fact, key signals to learn word embeddings are co-occurrences of words, which can lack certain information. In some cases, specific words can be replaced by a general word in a similar context, for instance, "I love cats" or "I love dogs" can be replaced with "I love animals". Therefore, the embeddings of "cats" and "dogs" should demonstrate an entailment relationship with respect to that of "animals". In the case probabilistic representation, "dogs" and "cats" become concentrated distributions that are encompassed by a more dispersed distribution of "animals", a word that "cats" and "dogs" entail. The broad distribution of a general word agrees with the *distributional informativeness hypothesis* proposed by Santus et al. [2014], which says that a generic word can occur in more general contexts in place of the specific ones that entail it. However, certain relationships are not likely to be encountered; for instance, a sentence "I love mammals" is highly unlikely and therefore the entailment signal between "cats" and "mammals" do not emerge through unsupervised training. Therefore, it is important to learn these relationships explicitly through hierarchical data rather

than text, which is the subject of Chapter 5.

Besides relationships among words, representations of sentences and images can reflect hierarchical structure where certain entities are abstractions of others in a visual-semantic space. For instance, an image caption "A dog and a frisbee" is an abstraction of many images with possible lower-level details such as a dog jumping to catch a frisbee or a dog sitting with a frisbee (Figure 5.1a). Recent work by Vendrov et al. [2016] proposes learning such asymmetric relationships with *order embeddings* – vector representations of non-negative coordinates with partial order structure. These embeddings are shown to be effective for word hypernym classification, image-caption ranking and textual entailment [Vendrov et al., 2016].

Chapter 5 shows that probability distributions are also natural at modeling hierarchical relationships and proposes effective methods for training. We describe a background subject of order embeddings in Section 2.2.1, a approach to learn embeddings of entities equipped with a partial order. Our approach fuses the order embeddings approach with the probabilistic embeddings, which we call density order embeddings (DOE) and is described thoroughly in Chapter 5. We describe other competing probabilistic embeddings in Section 2.2.2 and highlights the difference and advantages of our DOE model. These embeddings learned can be used directly on downstream tasks such as word or sentence entailment prediction. They can also be used as preprocessed embeddings for models such as density regression networks, described in Section 2.2.2, which take distributions as input and generate distributions as output. This procedure is akin to how standard neural networks propagate vectors but instead the DRN model propagates *distributions*.

We also provide overview of other non-probabilistic methods to model hierarchical data, for instance, through hyperbolic embeddings [Nickel and Kiela, 2017,

2018] which learns the embeddings of entities on skewed hyperbolic space with the advantages of being efficient on the number of dimensions needed. Hyperbolic neural networks [Ganea et al., 2018b, Gülçehre et al., 2019] are specialized networks that take hyperbolic embeddings to perform downstream tasks. We provide overview of hyperbolic methods in Section 2.2.3.

Other related work includes graph neural networks [Defferrard et al., 2016, Henaff et al., 2015, Kipf and Welling, 2017, Wu et al., 2019] whose input is the entire graph object. We give a brief summary in Section 2.2.4.

2.2.1 Order Embeddings and Partial Orders

We describe the concepts of partial orders and vector order embeddings proposed by Vendrov et al. [2016], which we will later consider in the context of our hierarchical density order embeddings in Chapter 5.

A partial order over a set of points X is a binary relation $\preceq$ such that for $a, b, c \in X$, the following properties hold: (1) $a \preceq a$ (reflexivity); (2) if $a \preceq b$ and $b \preceq a$ then $a = b$ (antisymmetry); and (3) if $a \preceq b$ and $b \preceq c$ then $a \preceq c$ (transitivity). An example of a partially ordered set is a set of nodes in a tree where $a \preceq b$ means a is a child node of b. This concept has applications in natural data such as lexical entailment. For words a and b, $a \preceq b$ means that every instance of a is an instance of b, or we can say that a entails b. We also say that (a, b) has a *hypernym* relationship where a is a hyponym of b and b is a hypernym of a. This relationship is asymmetric since $a \preceq b$ does not necessarily imply $(b \preceq a)$. For instance, `aircraft` $\preceq$ `vehicle` but it is not true that `vehicle` $\preceq$ `aircraft`.

An order-embedding is a function $f : (X, \preceq_X) \to (Y, \preceq_Y)$ where $a \preceq_X b$ if

and only if $f(a) \preceq_Y f(b)$. Vendrov et al. [2016] proposes to learn the embedding f on $Y = \mathbb{R}^N_+$ where all coordinates are non-negative. Under $\mathbb{R}^N_+$, there exists a partial order relation called the *reversed product order on* $\mathbb{R}^N_+$: $x \preceq y$ if and only if $\forall i, x_i \geq y_i$. That is, a point x entails y if and only if all the coordinate values of x is higher than y's. The origin represents the most general entity at the top of the order hierarchy and the points further away from the origin become more specific. Figure 5.1b demonstrates the vector order embeddings on $\mathbb{R}^N_+$. We can see that since `insect` $\preceq$ `animal` and `animal` $\preceq$ `organism`, we can infer directly from the embedding that `insect` $\preceq$ `organism` (orange line, diagonal line). To learn the embeddings, Vendrov et al. [2016] proposes a penalty function $E(x, y) = ||\max(0, y - x)||^2$ for a pair $x \preceq y$ which has the property that it is positive if and only if the order is violated.

Other related work includes Li et al. [2017] which extends Vendrov et al. [2016] for knowledge representation on data such as ConceptNet [Speer et al., 2016]. Another related work by Hockenmaier and Lai [2017] embeds words and phrases in a vector space and uses denotational probabilities for textual entailment tasks.

2.2.2 Probabilistic Embeddings

Our work in Chapter 5 is among the first to exploit the geometry of probability densities to model relationships among data. Another recent work by [Hockenmaier and Lai, 2017] extends order embeddings [Vendrov et al., 2016] and place probability measure on the $\mathbb{R}^+$ to form entailment cones, and is termed probabilistic order embeddings (POE). We note that our model uses density objects explicitly to model entity relationships where POE adapts vector order embeddings and models entailment through joint densities. However, POE has difficulties modeling negative

relationships where a pair of entities that do not entail each other but still have high joint probability. Box lattice models [Vilnis et al., 2018] remedies this problem by defining probability measures on box lattices instead of infinite entailment cones. However, sharp edges of box lattices can often inhibit learning due to zero gradients in case of non-overlapping boxes. Smooth box lattice models [Li et al., 2019] allows for better training by representing probability densities with convolutions of box lattices, which give rise to density objects with softer boundaries. We show comparison in our work in Section 5.3.3 where we perform at least on par or outperform all methods on a hypernym prediction problem. Related to probabilistic embeddings are distribution regression networks [Kou et al., 2018a,b] which take a distribution as input and outputs a distribution.

2.2.3 Hyperbolic Embeddings and Hyperbolic Networks

Hyperbolic embedding methods learn to map entities to points on hyperbolic spaces instead of the Euclidean space. Proposed models include Poincaré embeddings [Nickel and Kiela, 2017] which embeds entities onto a Poincare unit ball where the distance from the center grows to infinity as it gets closer to the edge. Poincare embeddings are known for its ability to represent large graphs with much smaller dimensions due to its infinite volume.

Follow-up work by Ganea et al. [2018a] improves empirical results of Poincaré embeddings through their proposal of theoretically grounded entailment cones. Another variant of hyperbolic embeddings is the Lorentz model [Nickel and Kiela, 2018] which is shown to be more efficient than using Poincaré embeddings to learn relationships from large-scale unstructured similar scores. Dhingra et al. [2018] provides experiments hyperbolic sentence embeddings by learning to construct

adjacent sentences given the middle sentence, similar to the skip-thought model [Kiros et al., 2015b].

These hyperbolic embeddings, however, can be suboptimal to use for downstream tasks with traditional neural networks due to indistinguishability among embeddings in the normal L_2 space which neural networks operate on. That is, distant points on the hyperbolic space can be very close to each other in the Euclidean space, which is the representation we store the vectors and weights in typical neural networks. Hyperbolic neural networks [Ganea et al., 2018b] or its attention-based version [Gülçehre et al., 2019] are principled models that incorporate the geometry of hyperbolic spaces and can meaningfully take hyperbolic embeddings as inputs.

2.2.4 Graph Neural Networks

Graph neural networks take different approaches to model hierarchical data compared to probabilistic embeddings or hyperbolic embeddings. As opposed to learning representations of each vertex on the graph structure, graph neural networks attempts to perform the end tasks while taking entire graph objects as inputs [Defferrard et al., 2016, Henaff et al., 2015, Kipf and Welling, 2017, Wu et al., 2019]. Applications include predicting attributes of molecules, which can be modeled as a classification problem where inputs are undirected graphs. It can also be used to predict possible edges in graph data such as social networks.

These graph neural networks take adjacency matrix E and perform end tasks directly through *graph convolution* operation. This operation combines the features of each node's neighbors and feed it to the next layer, which is similar to the convolution operator in CNN which combines features of adjoining pixels. As a

result, these networks are typically called *graph convolutional networks* (GCN). They also output nodes' latent representations which is an alternative approach to learning embeddings. GCNs have been used for a wide variety of supervised tasks, both at node-level such as relational reasoning or graph-level such as classification. GCNs also have applications for generative modeling where the networks, for instance, generate realistic looking images based on a scene graph input.

CHAPTER 3

PROBABILISTIC REPRESENTATION FOR WORD EMBEDDINGS

Word embeddings provide point representations of words containing useful semantic information. We introduce multimodal word distributions formed from Gaussian mixtures, for multiple word meanings, entailment, and rich uncertainty information. To learn these distributions, we propose an energy-based max-margin objective. We show that the resulting approach captures uniquely expressive semantic information, and outperforms alternatives, such as word2vec skip-grams, and Gaussian embeddings, on benchmark datasets such as word similarity and entailment.

3.1 Introduction

Vilnis and McCallum [2015] recently proposed an alternative view (Section 2.1.4), where words are represented by a whole probability distribution instead of a deterministic point vector. Specifically, they model each word by a Gaussian distribution, and learn its mean and covariance matrix from data. This approach generalizes any deterministic point embedding, which can be fully captured by the mean vector of the Gaussian distribution. Moreover, the full distribution provides much richer information than point estimates for characterizing words, representing probability mass and uncertainty across a set of semantics.

However, since a Gaussian distribution can have only one mode, the learned uncertainty in this representation can be overly diffuse for words with multiple distinct meanings (polysemies), in order for the model to assign *some* density to any plausible semantics. Moreover, the mean of the Gaussian can be pulled in

many opposing directions, leading to a biased distribution that centers its mass mostly around one meaning while leaving the others not well represented.

In this chapter, we propose to represent each word with an expressive multimodal distribution, for multiple distinct meanings, entailment, heavy tailed uncertainty, and enhanced interpretability. For example, one mode of the word 'bank' could overlap with distributions for words such as 'finance' and 'money', and another mode could overlap with the distributions for 'river' and 'creek'. It is our contention that such flexibility is critical for both qualitatively learning about the meanings of words, and for optimal performance on many predictive tasks. We note that the phenomonon of misrepresentation due to a unimodality constraint has been observed in other data domain such as images and rectified with a multimodal density approach as well [Oh et al., 2019].

In particular, we model each word with a mixture of Gaussians (Section 3.2.1). We learn all the parameters of this mixture model using a maximum margin energy-based ranking objective [Joachims, 2002, Vilnis and McCallum, 2015] (Section 3.2.3), where the energy function describes the affinity between a pair of words. For analytic tractability with Gaussian mixtures, we use the inner product between probability distributions in a Hilbert space, known as the expected likelihood kernel [Jebara et al., 2004], as our energy function (Section 3.2.4). Additionally, we propose transformations for numerical stability and initialization 3.2.5, resulting in a robust, straightforward, and scalable learning procedure, capable of training on a corpus with billions of words in days. We show that the model is able to automatically discover multiple meanings for words (Section 3.3.3), and significantly outperform other alternative methods across several tasks such as word similarity and entailment (Section 3.3.4, 3.3.5, 3.3.7). We discover that Chen et al. [2015] proposed a similar

model; however, our setup attains much better results on all evaluation metrics. We have made code available at http://github.com/benathi/word2gm, where we implement our model in Tensorflow [Abadi et al., 2015].

3.2 Methodology

In this section, we introduce our Gaussian mixture (GM) model for word representations, and present a training method to learn the parameters of the Gaussian mixture. This method uses an energy-based maximum margin objective, where we wish to maximize the similarity of distributions of nearby words in sentences. We propose an energy function that compliments the GM model by retaining analytic tractability. We also provide critical practical details for numerical stability, hyperparameters, and initialization.

3.2.1 Word Representation

We represent each word w in a dictionary as a Gaussian mixture with K components. Specifically, the distribution of w, f_w, is given by the density

$$f_w(\vec{x}) = \sum_{i=1}^{K} p_{w,i}\, \mathcal{N}\left[\vec{x}; \vec{\mu}_{w,i}, \Sigma_{w,i}\right] \tag{3.1}$$
$$= \sum_{i=1}^{K} \frac{p_{w,i}}{\sqrt{2\pi|\Sigma_{w,i}|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_{w,i})^\top \Sigma_{w,i}^{-1}(\vec{x}-\vec{\mu}_{w,i})},$$

where $\sum_{i=1}^{K} p_{w,i} = 1$.

The mean vectors $\vec{\mu}_{w,i}$ represent the location of the i^{th} component of word w, and are akin to the point embeddings provided by popular approaches like

`word2vec`. $p_{w,i}$ represents the component probability (mixture weight), and $\Sigma_{w,i}$ is the component covariance matrix, containing uncertainty information. Our goal is to learn all of the model parameters $\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}$ from a corpus of natural sentences to extract semantic information of words. Each Gaussian component's mean vector of word w can represent one of the word's distinct meanings. For instance, one component of a polysemous word such as 'rock' should represent the meaning related to 'stone' or 'pebbles', whereas another component should represent the meaning related to music such as 'jazz' or 'pop'. Figure 3.1 illustrates our word embedding model, and the difference between multimodal and unimodal representations, for words with multiple meanings.

3.2.2 Skip-Gram

The training objective for learning $\theta = \{\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}\}$ draws inspiration from the continuous skip-gram model [Mikolov et al., 2013b], where word embeddings are trained to maximize the probability of observing a word given another nearby word. This procedure follows the *distributional hypothesis* that words occurring in natural contexts tend to be semantically related. For instance, the words 'jazz' and 'music' tend to occur near one another more often than 'jazz' and 'cat'; hence, 'jazz' and 'music' are more likely to be related. The learned word representation contains useful semantic information and can be used to perform a variety of NLP tasks such as word similarity analysis, sentiment classification, modelling word analogies, or as a preprocessed input for complex system such as statistical machine translation.

Figure 3.1: **Top:** A Gaussian Mixture embedding, where each component corresponds to a distinct meaning. Each Gaussian component is represented by an ellipsoid, whose center is specified by the mean vector and contour surface specified by the covariance matrix, reflecting subtleties in meaning and uncertainty. On the left, we show examples of Gaussian mixture distributions of words where Gaussian components are randomly initialized. After training, we see on the right that one component of the word 'rock' is closer to 'stone' and 'basalt', whereas the other component is closer to 'jazz' and 'pop'. We also demonstrate the entailment concept where the distribution of the more general word 'music' encapsulates words such as 'jazz', 'rock', 'pop'. **Bottom:** A Gaussian embedding model [Vilnis and McCallum, 2015]. For words with multiple meanings, such as 'rock', the variance of the learned representation becomes unnecessarily large in order to assign some probability to both meanings. Moreover, the mean vector for such words can be pulled between two clusters, centering the mass of the distribution on a region which is far from certain meanings.

3.2.3 Energy-based Max-Margin Objective

Each sample in the objective consists of two pairs of words, (w, c) and (w, c'). w is sampled from a sentence in a corpus and c is a nearby word within a context window of length ℓ. For instance, a word $w =$ 'jazz' which occurs in the sentence 'I listen to jazz music' has context words ('I', 'listen', 'to' , 'music'). c' is a negative context word (e.g. 'airplane') obtained from random sampling.

The objective is to maximize the energy between words that occur near each other, w and c, and minimize the energy between w and its negative context c'. This approach is similar to negative sampling [Mikolov et al., 2013b,c], which contrasts the dot product between positive context pairs with negative context pairs. The energy function is a measure of similarity between distributions and will be discussed in Section 3.2.4.

We use a max-margin ranking objective [Joachims, 2002], used for Gaussian embeddings in Vilnis and McCallum [2015], which pushes the similarity of a word and its positive context higher than that of its negative context by a margin m:

$$L_\theta(w, c, c') = \max(0, m - \log E_\theta(w, c) + \log E_\theta(w, c')) \tag{3.2}$$

This objective can be minimized by mini-batch stochastic gradient descent with respect to the parameters $\theta = \{\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}\}$ – the mean vectors, covariance matrices, and mixture weights – of our multimodal embedding in Eq. (3.1).

Word Sampling We use a word sampling scheme similar to the implementation in WORD2VEC [Mikolov et al., 2013b,c] to balance the importance of frequent words and rare words. Frequent words such as 'the', 'a', 'to' are not as meaningful as relatively less frequent words such as 'dog', 'love', 'rock', and we are often more interested in learning the semantics of the less frequently observed words. We use subsampling to improve the performance of learning word vectors [Mikolov et al., 2013c]. This technique discards word w_i with probability $P(w_i) = 1 - \sqrt{t/f(w_i)}$, where $f(w_i)$ is the frequency of word w_i in the training corpus and t is a frequency threshold.

To generate negative context words, each word type w_i is sampled according to a distribution $P_n(w_i) \propto U(w_i)^{3/4}$ which is a distorted version of the unigram

distribution $U(w_i)$ that also serves to diminish the relative importance of frequent words. Both subsampling and the negative distribution choice are proven effective in WORD2VEC training [Mikolov et al., 2013c].

3.2.4 Energy Function

For vector representations of words, a usual choice for similarity measure (energy function) is a dot product between two vectors. Our word representations are distributions instead of point vectors and therefore need a measure that reflects not only the point similarity, but also the uncertainty.

Expected Likelihood Kernel

We propose to use the *expected likelihood kernel*, which is a generalization of an inner product between vectors to an inner product between distributions [Jebara et al., 2004]. That is,

$$E(f,g) = \int f(x)g(x)\ dx = \langle f,g\rangle_{L_2} \tag{3.3}$$

where $\langle\cdot,\cdot\rangle_{L_2}$ denotes the inner product in Hilbert space L_2. We choose this form of energy since it can be evaluated in a closed form given our choice of probabilistic embedding in Eq. (3.1).

For Gaussian mixtures f, g representing the words w_f, w_g, $f(x) = \sum_{i=1}^{K} p_i \mathcal{N}(x; \vec{\mu}_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^{K} q_i \mathcal{N}(x; \vec{\mu}_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^{K} p_i = 1$, and $\sum_{i=1}^{K} q_i = 1$, we find that the log energy is

$$\log E_\theta(f,g) = \log \sum_{j=1}^{K} \sum_{i=1}^{K} p_i q_j e^{\xi_{i,j}} \tag{3.4}$$

where

$$\begin{aligned}\xi_{i,j} &\equiv \log \mathcal{N}(0; \vec{\mu}_{f,i} - \vec{\mu}_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}) \\ &= -\frac{1}{2}\log\det(\Sigma_{f,i} + \Sigma_{g,j}) - \frac{D}{2}\log(2\pi) \\ &-\frac{1}{2}(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top(\Sigma_{f,i} + \Sigma_{g,j})^{-1}(\vec{\mu}_{f,i} - \vec{\mu}_{g,j}) \end{aligned} \tag{3.5}$$

The derivation is as follows. Let f, g be Gaussian mixture distributions representing the words w_f, w_g. That is, $f(x) = \sum_{i=1}^{K} p_i \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^{K} q_i \mathcal{N}(x; \mu_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^{K} p_i = 1$, and $\sum_{i=1}^{K} q_i = 1$. The expected likelihood kernel is given by

$$\begin{aligned} E_\theta(f, g) &= \int \left(\sum_{i=1}^{K} p_i \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i})\right) \cdot \\ &\quad \left(\sum_{j=1}^{K} q_j \mathcal{N}(x; \mu_{g,j}, \Sigma_{g,j})\right)\, dx \\ &= \sum_{i=1}^{K}\sum_{j=1}^{K} p_i q_j \int \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i}) \cdot \mathcal{N}(x; \mu_{g,j}, \Sigma_{g,j})\, dx \\ &= \sum_{i=1}^{K}\sum_{j=1}^{K} p_i q_j \mathcal{N}(0; \mu_{f,i} - \mu_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}) \\ &= \sum_{i=1}^{K}\sum_{j=1}^{K} p_i q_j e^{\xi_{i,j}} \end{aligned}$$

where we note that $\int \mathcal{N}(x; \mu_i, \Sigma_i)\mathcal{N}(x; \mu_j, \Sigma_j)\, dx = \mathcal{N}(0, \mu_i - \mu_j, \Sigma_i + \Sigma_j)$ [Vilnis and McCallum, 2015] and we call $\xi_{i,j}$ the (log) partial energy, given by equation 3.5. Figure 3.2 demonstrates the partial energies among the Gaussian components of two words.

Observe that the partial energy term captures the similarity between the i^{th} meaning of word w_f and the j^{th} meaning of word w_g. The total energy in Equation 3.4 is the sum of possible pairs of partial energies, weighted accordingly by the mixture probabilities p_i and q_j.

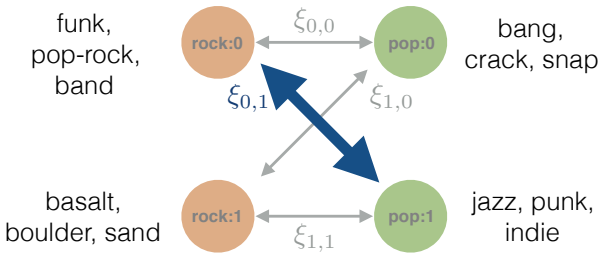


Figure 3.2: The interactions among Gaussian components of word `rock` and word `pop`. The partial energy is the highest for the pair `rock:0` (the zeroth component of rock) and `pop:1` (the first component of pop), reflecting the similarity in meanings.

The term $-(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top (\Sigma_{f,i} + \Sigma_{g,j})^{-1} (\vec{\mu}_{f,i} - \vec{\mu}_{g,j})$ in $\xi_{i,j}$ explains the difference in mean vectors of semantic pair (w_f, i) and (w_g, j). If the semantic uncertainty (covariance) for both pairs are low, this term has more importance relative to other terms due to the inverse covariance scaling. We observe that the loss function L_θ in Section 3.2.3 attains a low value when $E_\theta(w, c)$ is relatively high. High values of $E_\theta(w, c)$ can be achieved when the component means across different words $\vec{\mu}_{f,i}$ and $\vec{\mu}_{g,j}$ are close together (e.g., similar point representations). High energy can also be achieved by large values of $\Sigma_{f,i}$ and $\Sigma_{g,j}$, which washes out the importance of the mean vector difference. The term $-\log\det(\Sigma_{f,i} + \Sigma_{g,j})$ serves as a regularizer that prevents the covariances from being pushed too high at the expense of learning a good mean embedding.

At the beginning of training, $\xi_{i,j}$ roughly are on the same scale among all pairs (i, j)'s. During this time, all components learn the signals from the word occurrences equally. As training progresses and the semantic representation of each mixture becomes more clear, there can be one term of $\xi_{i,j}$'s that is predominantly higher than other terms, giving rise to a semantic pair that is most related.

Probability Product Kernel

In general, the probability product kernel $K_\rho(f,g) = \int f(x)^\rho g(x)^\rho \, dx$ for $\rho > 0$ between two Gaussians are:

$$\begin{aligned} \xi_{i,j}^\rho &\equiv \log K_\rho(f_i, g_j) \\ &= (1-2\rho)\frac{D}{2}\log(2\pi) - \frac{D}{2}\log(\rho) \\ &\quad + \log\det\left[\Sigma_{f,i}^{\rho-1}\Sigma_{g,j}^{\rho} + \Sigma_{f,i}^{\rho}\Sigma_{g,j}^{\rho-1}\right] \\ &\quad -\frac{\rho}{2}(\mu_{f,i}-\mu_{g,j})(\Sigma_{f,i}+\Sigma_{g,j})^{-1}(\mu_{f,i}-\mu_{g,j}) \end{aligned} \tag{3.6}$$

For mixture of Gaussians, we have

$$\log E_\theta^\rho(f,g) = \sum_{i=1}^{K}\sum_{j=1}^{K}(p_i q_j)^\rho e^{\xi_{i,j}^\rho} \tag{3.7}$$

Note that for the case where $\rho = 1$, we recover the expected likelihood kernel in Section 3.2.4

Other Energy Functions

The negative KL divergence is another sensible choice of energy function, providing an asymmetric metric between word distributions. However, unlike the expected likelihood kernel, KL divergence does not have a closed form if the two distributions are Gaussian mixtures.

3.2.5 Implementation

In this section we discuss practical details for training the proposed model.

Reduction to Diagonal Covariance

We use a diagonal Σ, in which case inverting the covariance matrix is trivial and computations are particularly efficient.

Let $\boldsymbol{d}^f, \boldsymbol{d}^g$ denote the diagonal vectors of Σ_f, Σ_g The expression for $\xi_{i,j}$ reduces to

$$\xi_{i,j} = -\frac{1}{2}\sum_{r=1}^{D}\log(d_r^p + d_r^q) - \frac{1}{2}\sum\left[(\boldsymbol{\mu}_{p,i} - \boldsymbol{\mu}_{q,j}) \circ \frac{1}{\boldsymbol{d^p} + \boldsymbol{d^q}} \circ (\boldsymbol{\mu}_{p,i} - \boldsymbol{\mu}_{q,j})\right] \quad (3.8)$$

where $\circ$ denotes element-wise multiplication. The spherical case which we use in all our experiments is similar since we simply replace a vector $\boldsymbol{d}$ with a single value.

Optimization Constraint and Stability

We optimize $\log \boldsymbol{d}$ since each component of diagonal vector $\boldsymbol{d}$ is constrained to be positive. Similarly, we constrain the probability p_i to be in $[0, 1]$ and sum to 1 by optimizing over unconstrained scores $s_i \in (-\infty, \infty)$ and using a softmax function to convert the scores to probability $p_i = \frac{e^{s_i}}{\sum_{j=1}^{K} e^{s_j}}$.

The loss computation can be numerically unstable if elements of the diagonal covariances are very small, due to the term $\log(d_r^f + d_r^g)$ and $\frac{1}{\boldsymbol{d}^q + \boldsymbol{d}^p}$. Therefore, we add a small constant $\epsilon = 10^{-4}$ so that $d_r^f + d_r^g$ and $\boldsymbol{d}^q + \boldsymbol{d}^p$ becomes $d_r^f + d_r^g + \epsilon$ and $\boldsymbol{d^q} + \boldsymbol{d^p} + \epsilon$.

In addition, we observe that $\xi_{i,j}$ can be very small which would result in $e^{\xi_{i,j}} \approx 0$ up to machine precision. In order to stabilize the computation in eq. 3.4, we compute its equivalent form

$$\log E(f, g) = \xi_{i',j'} + \log \sum_{j=1}^{K}\sum_{i=1}^{K} p_i q_j e^{\xi_{i,j} - \xi_{i',j'}} \quad (3.9)$$

where $\xi_{i',j'} = \max_{i,j} \xi_{i,j}$.

Model Hyperparameters and Training Details

In the loss function L_θ, we use a margin $m = 1$ and a batch size of 128. We initialize the word embeddings with a uniform distribution over $[-\sqrt{\frac{3}{D}}, \sqrt{\frac{3}{D}}]$ so that the expectation of variance is 1 and the mean is zero [LeCun et al., 1998]. We initialize each dimension of the diagonal matrix (or a single value for spherical case) with a constant value $v = 0.05$. We also initialize the mixture scores s_i to be 0 so that the initial probabilities are equal among all K components. We use the threshold $t = 10^{-5}$ for negative sampling, which is the recommended value for WORD2VEC skip-gram on large datasets.

We also use a separate output embeddings in addition to input embeddings, similar to WORD2VEC implementation [Mikolov et al., 2013b,c]. That is, each word has two sets of distributions q_I and q_O, each of which is a Gaussian mixture. For a given pair of word and context (w, c), we use the input distribution q_I for w (input word) and the output distribution q_O for context c (output word). We optimize the parameters of both q_I and q_O and use the trained input distributions q_I as our final word representations.

We use mini-batch asynchronous gradient descent with Adagrad [Duchi et al., 2011] which performs adaptive learning rate for each parameter. We also experiment with Adam [Kingma and Ba, 2014] which corrects the bias in adaptive gradient update of Adagrad and is proven very popular for most recent neural network models. However, we found that it is much slower than Adagrad ($\approx$ 10 times). This is because the gradient computation of the model is relatively fast, so a complex gradient update algorithm such as Adam becomes the bottleneck in the

optimization. Therefore, we choose to use Adagrad which allows us to better scale to large datasets. We use a linearly decreasing learning rate from 0.05 to 0.00001.

Word	Co.	Nearest Neighbors
rock	0	basalt:1, boulder:1, boulders:0, stalagmites:0, stalactites:0, rocks:1, sand:0, quartzite:1, bedrock:0
rock	1	rock/:1, ska:0, funk:1, pop-rock:1, punk:1, indie-rock:0, band:0, indie:0, pop:1
bank	0	banks:1, mouth:1, river:1, River:0, confluence:0, waterway:1, downstream:1, upstream:0, dammed:0
bank	1	banks:0, banking:1, banker:0, Banks:1, bankas:1, Citibank:1, Interbank:1, Bankers:0, transactions:1
Apple	0	Strawberry:0, Tomato:1, Raspberry:1, Blackberry:1, Apples:0, Pineapple:1, Grape:1, Lemon:0
Apple	1	Macintosh:1, Mac:1, OS:1, Amiga:0, Compaq:0, Atari:1, PC:1, Windows:0, iMac:0
star	0	stars:0, Quaid:0, starlet:0, Dafoe:0, Stallone:0, Geena:0, Niro:0, Zeta-Jones:1, superstar:0
star	1	stars:1, brightest:0, Milky:0, constellation:1, stellar:0, nebula:1, galactic:1, supernova:1, Ophiuchus:1
cell	0	cellular:0, Nextel:0, 2-line:0, Sprint:0, phones.:1, pda:1, handset:0, handsets:1, pushbuttons:0
cell	1	cytoplasm:0, vesicle:0, cytoplasmic:1, macrophages:0, secreted:1, membrane:0, mitotic:0, endocytosis:1
left	0	After:1, back:0, finally:1, eventually:0, broke:0, joined:1, returned:1, after:1, soon:0
left	1	right-hand:0, hand:0, right:0, left-hand:0, lefthand:0, arrow:0, turn:0, righthand:0, Left:0

Word	Nearest Neighbors
rock	band, bands, Rock, indie, Stones, breakbeat, punk, electronica, funk
bank	banks, banking, trader, trading, Bank, capital, Banco, bankers, cash
Apple	Macintosh, Microsoft, Windows, Macs, Lite, Intel, Desktop, WordPerfect, Mac
star	stars, stellar, brightest, Stars, Galaxy, Stardust, eclipsing, stars., Star
cell	cells, DNA, cellular, cytoplasm, membrane, peptide, macrophages, suppressor, vesicles
left	leaving, turned, back, then, After, after, immediately, broke, end

Table 3.1: Nearest neighbors based on cosine similarity between the mean vectors of Gaussian components for Gaussian mixture embedding (top) (for $K = 2$) and Gaussian embedding (bottom). The notation `w:i` denotes the i^{th} mixture component of the word `w`.

3.3 Experiments

We have introduced a model for multi-prototype embeddings, which expressively captures word meanings with whole probability distributions. We show that our combination of energy and objective functions, proposed in Section 3.2, enables one to learn interpretable multimodal distributions through unsupervised training, for describing words with multiple distinct meanings. By representing multiple distinct meanings, our model also reduces the unnecessarily large variance of a Gaussian embedding model, and has improved results on word entailment tasks.

To learn the parameters of the proposed mixture model, we train on a con-

catenation of two datasets: `UKWAC` (2.5 billion tokens) and `Wackypedia` (1 billion tokens) [Baroni et al., 2009]. We discard words that occur fewer than 100 times in the corpus, which results in a vocabulary size of 314, 129 words. Our word sampling scheme, described at the end of Section 3.3.3, is similar to that of WORD2VEC with one negative context word for each positive context word.

After training, we obtain learned parameters $\{\vec{\mu}_{w,i}, \Sigma_{w,i}, p_i\}_{i=1}^{K}$ for each word w. We treat the mean vector $\vec{\mu}_{w,i}$ as the embedding of the i^{th} mixture component with the covariance matrix $\Sigma_{w,i}$ representing its subtlety and uncertainty. We perform qualitative evaluation to show that our embeddings learn meaningful multi-prototype representations and compare to existing models using a quantitative evaluation on word similarity datasets and word entailment.

We name our model as Word to Gaussian Mixture (W2GM) in constrast to Word to Gaussian (W2G) [Vilnis and McCallum, 2015]. Unless stated otherwise, W2G refers to our implementation of W2GM model with one mixture component.

3.3.1 Hyperparameters

Unless stated otherwise, we experiment with $K = 2$ components for the W2GM model, but we have results and discussion of $K = 3$ at the end of section 3.3.3. We primarily consider the spherical case for computational efficiency. We note that for diagonal or spherical covariances, the energy can be computed very efficiently since the matrix inversion would simply require $\mathcal{O}(d)$ computation instead of $\mathcal{O}(d^3)$ for a full matrix. Empirically, we have found diagonal covariance matrices become roughly spherical after training. Indeed, for these relatively high dimensional embeddings, there are sufficient degrees of freedom for the mean vectors to be

learned such that the covariance matrices need not be asymmetric. Therefore, we perform all evaluations with spherical covariance models.

Models used for evaluation have dimension $D = 50$ and use context window $\ell = 10$ unless stated otherwise. We provide additional hyperparameters and training details in Section (3.2.5).

3.3.2 Similarity Measures

Since our word embeddings contain multiple vectors and uncertainty parameters per word, we use the following measures that generalizes similarity scores. These measures pick out the component pair with maximum similarity and therefore determine the meanings that are most relevant.

Expected Likelihood Kernel

A natural choice for a similarity score is the expected likelihood kernel, an inner product between distributions, which we discussed in Section 3.2.4. This metric incorporates the uncertainty from the covariance matrices in addition to the similarity between the mean vectors.

Maximum Cosine Similarity

This metric measures the maximum similarity of mean vectors among all pairs of mixture components between distributions f and g. That is, $d(f, g) = \max_{i,j=1,\ldots,K} \frac{\langle \boldsymbol{\mu}_{f,i}, \boldsymbol{\mu}_{g,j} \rangle}{||\boldsymbol{\mu}_{f,i}|| \cdot ||\boldsymbol{\mu}_{g,j}||}$, which corresponds to matching the meanings of f and

g that are the most similar. For a Gaussian embedding, maximum similarity reduces to the usual cosine similarity.

Minimum Euclidean Distance

Cosine similarity is popular for evaluating embeddings. However, our training objective directly involves the Euclidean distance in Eq. (3.5), as opposed to dot product of vectors such as in WORD2VEC. Therefore, we also consider the Euclidean metric: $d(f, g) = \min_{i,j=1,\ldots,K}[||\boldsymbol{\mu}_{f,i} - \boldsymbol{\mu}_{g,j}||]$.

3.3.3 Qualitative Evaluation

In Table 3.1, we show examples of polysemous words and their nearest neighbors in the embedding space to demonstrate that our trained embeddings capture multiple word senses. For instance, a word such as 'rock' that could mean either 'stone' or 'rock music' should have each of its meanings represented by a distinct Gaussian component. Our results for a mixture of two Gaussians model confirm this hypothesis, where we observe that the 0^{th} component of 'rock' being related to ('basalt', 'boulders') and the 1^{st} component being related to ('indie', 'funk', 'hip-hop'). Similarly, the word `bank` has its 0^{th} component representing the river bank and the 1^{st} component representing the financial bank.

By contrast, in Table 3.1 (bottom), see that for Gaussian embeddings with one mixture component, nearest neighbors of polysemous words are predominantly related to a single meaning. For instance, 'rock' mostly has neighbors related to rock music and 'bank' mostly related to the financial bank. The alternative meanings of these polysemous words are not well represented in the embeddings.

As a numerical example, the cosine similarity between 'rock' and 'stone' for the Gaussian representation of Vilnis and McCallum [2015] is only 0.029, much lower than the cosine similarity 0.586 between the 0^{th} component of 'rock' and 'stone' in our multimodal representation.

In cases where a word only has a single popular meaning, the mixture components can be fairly close; for instance, one component of 'stone' is close to ('stones', 'stonework', 'slab') and the other to ('carving, 'relic', 'excavated'), which reflects subtle variations in meanings. In general, the mixture can give properties such as heavy tails and more interesting unimodal characterizations of uncertainty than could be described by a single Gaussian.

Embedding Visualization We provide an interactive visualization as part of our code repository: `https://github.com/benathi/word2gm#visualization` that allows real-time queries of words' nearest neighbors (in the `embeddings` tab) for $K = 1, 2, 3$ components. We use a notation similar to that of Table 3.1, where a token `w:i` represents the component `i` of a word `w`. For instance, if in the $K = 2$ link we search for `bank:0`, we obtain the nearest neighbors such as `river:1, confluence:0, waterway:1`, which indicates that the 0^{th} component of 'bank' has the meaning 'river bank'. On the other hand, searching for `bank:1` yields nearby words such as `banking:1, banker:0, ATM:0`, indicating that this component is close to the 'financial bank'. We also have a visualization of a unimodal (W2G) for comparison in the $K = 1$ link.

In addition, the embedding link for our Gaussian mixture model with $K = 3$ mixture components can learn three distinct meanings. For instance, each of the three components of 'cell' is close to ('keypad', 'digits'), ('incarcerated', 'inmate')

Dataset	sg*	w2g*	w2g/mc	w2g/el	w2g/me	w2gm/mc	w2gm/el	w2gm/me
SL	29.39	**32.23**	29.35	25.44	25.43	29.31	26.02	27.59
WS	59.89	65.49	71.53	61.51	64.04	**73.47**	62.85	66.39
WS-S	69.86	76.15	76.70	70.57	72.3	**76.73**	70.08	73.3
WS-R	53.03	58.96	68.34	54.4	55.43	**71.75**	57.98	60.13
MEN	70.27	71.31	72.58	67.81	65.53	**73.55**	68.5	67.7
MC	63.96	70.41	76.48	72.70	**80.66**	79.08	76.75	80.33
RG	70.01	71	73.30	72.29	72.12	**74.51**	71.55	73.52
YP	39.34	41.5	41.96	38.38	36.41	**45.07**	39.18	38.58
MT-287	-	-	64.79	57.5	58.31	**66.60**	57.24	60.61
MT-771	-	-	**60.86**	55.89	54.12	60.82	57.26	56.43
RW	-	-	28.78	32.34	33.16	28.62	31.64	**35.27**

Table 3.2: Spearman correlation for word similarity datasets. The models `sg, w2g, w2gm` denote WORD2VEC skip-gram, Gaussian embedding, and Gaussian mixture embedding (K=2). The measures `mc, el, me` denote maximum cosine similarity, expected likelihood kernel, and minimum Euclidean distance. For each of W2G and W2GM, we underline the similarity metric with the best score. For each dataset, we boldface the score with the best performance across all models. The correlation scores for `sg*, w2g*` are taken from Vilnis and McCallum [2015] and correspond to cosine distance.

or ('tissue', 'antibody'), indicating that the distribution captures the concept of 'cellphone', 'jail cell', or 'biological cell', respectively. Due to the limited number of words with more than 2 meanings, our model with $K = 3$ does not generally offer substantial performance differences to our model with $K = 2$; hence, we do not further display $K = 3$ results for compactness.

3.3.4 Word Similarity

We evaluate our embeddings on several standard word similarity datasets, namely, SimLex [Hill et al., 2014], WS or WordSim-353, WS-S (similarity), WS-R (relatedness) [Finkelstein et al., 2002], MEN [Bruni et al., 2014], MC [Miller and Charles, 1991], RG [Rubenstein and Goodenough, 1965], YP [Yang and Powers,

Model	$\rho \times 100$
Huang	64.2
Huang*	71.3
MSSG 50d	63.2
MSSG 300d	71.2
w2g	70.9
w2gm	**73.5**

Table 3.3: Spearman's correlation (ρ) on WordSim-353 datasets for our Word to Gaussian Mixture embeddings, as well as the multi-prototype embedding by Huang et al. [2012] and the MSSG model by Neelakantan et al. [2014]. Huang* is trained using data with all stop words removed. All models have dimension $D = 50$ except for MSSG 300D with $D = 300$ which is still outperformed by our w2gm model.

2006], MTurk(-287,-771) [Halawi et al., 2012, Radinsky et al., 2011], and RW [Luong et al., 2013]. Each dataset contains a list of word pairs with a human score of how related or similar the two words are.

We calculate the Spearman correlation [Spearman, 1904] between the labels and our scores generated by the embeddings. The Spearman correlation is a rank-based correlation measure that assesses how well the scores describe the true labels.

The correlation results are shown in Table 3.2 using the scores generated from the expected likelihood kernel, maximum cosine similarity, and maximum Euclidean distance.

We show the results of our Gaussian mixture model and compare the performance with that of word2vec and the original Gaussian embedding by Vilnis and McCallum [2015]. We note that our model of a unimodal Gaussian embedding w2g also outperforms the original model, which differs in model hyperparameters and initialization, for most datasets.

Our multi-prototype model w2gm also performs better than skip-gram or

Gaussian embedding methods on many datasets, namely, `WS`, `WS-R`, `MEN`, `MC`, `RG`, `YP`, `MT-287`, `RW`. The maximum cosine similarity yields the best performance on most datasets; however, the minimum Euclidean distance is a better metric for the datasets `MC` and `RW`. These results are consistent for both the single-prototype and the multi-prototype models.

We also compare out results on WordSim-353 with the multi-prototype embedding method by Huang et al. [2012] and Neelakantan et al. [2014], shown in Table 3.3. We observe that our single-prototype model W2G is competitive compared to models by Huang et al. [2012], even without using a corpus with stop words removed. This could be due to the auto-calibration of importance via the covariance learning which decrease the importance of very frequent words such as 'the', 'to', 'a', etc. Moreover, our multi-prototype model substantially outperforms the model of Huang et al. [2012] and the MSSG model of Neelakantan et al. [2014] on the WordSim-353 dataset.

3.3.5 Word Similarity for Polysemous Words

We use the dataset SCWS introduced by Huang et al. [2012], where word pairs are chosen to have variations in meanings of polysemous and homonymous words.

We compare our method with multiprototype models by HUANG [Huang et al., 2012], TIAN [Tian et al., 2014], CHEN [Chen et al., 2014], and MSSG model by [Neelakantan et al., 2014]. We note that CHEN model uses an external lexical source WORDNET that gives it an extra advantage.

We use many metrics to calculate the scores for the Spearman correlation. `MaxSim` refers to the maximum cosine similarity. `AveSim` is the average of cosine

Model	Dimension	$\rho \times 100$
word2vec skip-gram	50	61.7
Huang-S	50	58.6
w2g	50	**64.7**
Chen-S	200	64.2
w2g	200	**66.2**
Huang-M AvgSim	50	62.8
Tian-M MaxSim	50	63.6
w2gm MaxSim	50	62.7
MSSG AvgSim	50	**64.2**
Chen-M AvgSim	200	**66.2**
w2gm MaxSim	200	65.5

Table 3.4: Spearman's correlation ρ on dataset SCWS. We show the results for single prototype (top) and multi-prototype (bottom). The suffix `-(S,M)` refers to single and multiple prototype models, respectively.

similarities with respect to the component probabilities.

In Table 3.4, the model w2g performs the best among all single-prototype models for either 50 or 200 vector dimensions. Our model w2gm performs competitively compared to other multi-prototype models. In SCWS, the gain in flexibility in moving to a probability density approach appears to dominate over the effects of using a multi-prototype. In most other examples, we see w2gm surpass w2g, where the multi-prototype structure is just as important for good performance as the probabilistic representation. Note that other models also use `AvgSimC` metric which uses context information which can yield better correlation [Chen et al., 2014, Huang et al., 2012]. We report the numbers using `AvgSim` or `MaxSim` from the existing models which are more comparable to our performance with `MaxSim`.

Model	Score	Best AP	Best F1
w2g (5)	cos	73.1	76.4
w2g (5)	kl	73.7	76.0
w2gm (5)	cos	73.6	76.3
w2gm (5)	kl	75.7	77.9
w2g (10)	cos	73.0	76.1
w2g (10)	kl	74.2	76.1
w2gm (10)	cos	72.9	75.6
w2gm (10)	kl	74.7	76.3

Table 3.5: Entailment results for models w2g and w2gm with window size 5 and 10 for maximum cosine similarity and the maximum negative KL divergence. We calculate the best average precision and the best F1 score. In most cases, w2gm outperforms w2g for describing entailment.

3.3.6 Reduction in Variance of Polysemous Words

One motivation for our Gaussian mixture embedding is to model word uncertainty more accurately than Gaussian embeddings, which can have overly large variances for polysemous words (in order to assign some mass to all of the distinct meanings). We see that our Gaussian mixture model does indeed reduce the variances of each component for such words. For instance, we observe that the word `rock` in w2g has much higher variance per dimension ($e^{-1.8} \approx 1.65$) compared to that of Gaussian components of `rock` in w2gm (which has variance of roughly $e^{-2.5} \approx 0.82$). We also see, in the next section, that w2gm has desirable quantitative behavior for word entailment.

3.3.7 Word Entailment

We evaluate our embeddings on the word entailment dataset from Baroni et al. [2012]. The lexical entailment between words is denoted by $w_1 \preceq w_2$ which means

that all instances of w_1 are w_2. The entailment dataset contains positive pairs such as *aircraft* $\preceq$ *vehicle* and negative pairs such as *aircraft* $\not\preceq$ *insect*.

We generate entailment scores of word pairs and find the best threshold, measured by Average Precision (AP) or F1 score, which identifies negative versus positive entailment. We use the maximum cosine similarity and the minimum KL divergence, $d(f, g) = \min_{i,j=1,\ldots,K} KL(f||g)$, for entailment scores. The minimum KL divergence is similar to the maximum cosine similarity, but also incorporates the embedding uncertainty. In addition, KL divergence is an asymmetric measure, which is more suitable for certain tasks such as word entailment where a relationship is unidirectional. For instance, $w_1 \preceq w_2$ does not imply $w_2 \preceq w_1$. Indeed, *aircraft* $\preceq$ *vehicle* does not imply *vehicle* $\preceq$ *aircraft*, since all aircraft are vehicles but not all vehicles are aircraft. The difference between $KL(w_1||w_2)$ versus $KL(w_2||w_1)$ distinguishes which word distribution encompasses another distribution, as demonstrated in Figure 3.1.

Table 3.5 shows the results of our W2GM model versus the Gaussian embedding model W2G. We observe a trend for both models with window size 5 and 10 that the KL metric yields improvement (both AP and F1) over cosine similarity. In addition, W2GM generally outperforms W2G.

The multi-prototype model estimates the meaning uncertainty better since it is no longer constrained to be unimodal, leading to better characterizations of entailment. On the other hand, the Gaussian embedding model suffers from overestimatating variances of polysemous words, which results in less informative word distributions and reduced entailment scores.

3.4 Discussion

We introduced a model that represents words with expressive multimodal distributions formed from Gaussian mixtures. To learn the properties of each mixture, we proposed an analytic energy function for combination with a maximum margin objective. The resulting embeddings capture different semantics of polysemous words, uncertainty, and entailment, and also perform favorably on word similarity benchmarks.

Elsewhere, latent probabilistic representations are proving to be exceptionally valuable, able to capture nuances such as face angles with variational autoencoders [Kingma and Welling, 2013] or subtleties in painting strokes with the InfoGAN [Chen et al., 2016]. Moreover, classically deterministic deep learning architectures are actively being generalized to probabilistic deep models, for full predictive distributions instead of point estimates, and significantly more expressive representations [Al-Shedivat et al., 2016, Fortunato et al., 2017b, Gan et al., 2016, Wilson et al., 2016a,b].

Similarly, probabilistic word embeddings can capture a range of subtle meanings, and advance the state of the art. Multimodal word distributions naturally represent our belief that words do not have single precise meanings: indeed, the shape of a word distribution can express much more semantic information than any point representation.

In the future, multimodal word distributions could open the doors to a new suite of applications in language modelling, where whole word distributions are used as inputs to new probabilistic LSTMs, or in decision functions where uncertainty matters. As part of this effort, we can explore different metrics between distributions,

such as KL divergences, which would be a natural choice for order embeddings that model entailment properties. It would also be informative to explore inference over the number of components in mixture models for word distributions. Such an approach could potentially discover an unbounded number of distinct meanings for words, but also distribute the support of each word distribution to express highly nuanced meanings. Alternatively, we could imagine a dependent mixture model where the distributions over words are evolving with time and other covariates. One could also build new types of supervised language models, constructed to more fully leverage the rich information provided by word distributions.

CHAPTER 4

PROBABILISTIC WORD EMBEDDINGS WITH CHARACTER INFORMATION

We introduce PROBABILISTIC FASTTEXT (PFT), a new model for word embeddings that can capture multiple word senses, sub-word structure, and uncertainty information. In particular, we represent each word with a Gaussian mixture density, where the mean of a mixture component is given by the sum of n-grams. This representation allows the model to share statistical strength across sub-word structures (e.g. Latin roots), producing accurate representations of rare, misspelt, or even unseen words. Moreover, each component of the mixture can capture a different word sense. PFT outperforms both FASTTEXT, which has no probabilistic model, and dictionary-level probabilistic embeddings, which do not incorporate subword structures, on several word-similarity benchmarks, including English RareWord and foreign language datasets. We also achieve state-of-art performance on benchmarks that measure ability to discern different meanings. Thus, the proposed model is the first to achieve multi-sense representations while having enriched semantics on rare words.

4.1 Introduction

Word embeddings introduced thus far such as WORD2VEC, GloVe or `word2gm` are dictionary-level embeddings, which rely on a vocabulary lookup table consisting of words only in the training set. One shortcoming with the above approaches to word embedding that are based on a predefined dictionary (termed as dictionary-based embeddings) is their inability to learn representations of rare or out-of-vocabulary words. To overcome this limitation, character-level word embeddings have been

proposed. FastText [Bojanowski et al., 2016] is the state-of-the-art character-level approach to embeddings. Each word is modeled by a sum of vectors, with each vector representing an n-gram. The benefit of this approach is that the training process can then share strength across words composed of common roots. For example, with individual representations for "circum" and "navigation", we can construct an informative representation for "circumnavigation", which would otherwise appear too infrequently to learn a dictionary-level embedding. In addition to effectively modelling rare words, character-level embeddings can also represent slang or misspelled words, such as "dogz", and can share strength across different languages that share roots, e.g. Romance languages share latent roots.

pft provides probabilistic character-level representations of words based on Gaussian mixture representation and FastText subword structure. The resulting word embeddings are highly expressive, yet straightforward and interpretable, with simple, efficient, and intuitive training procedures. pft can model rare words, uncertainty information, hierarchical representations, and multiple word senses. In particular, we represent each word with a Gaussian or a Gaussian mixture density, which we name pft-g and pft-gm respectively. Each component of the mixture can represent different word senses, and the mean vectors of each component decompose into vectors of n-grams, to capture character-level information. We also derive an efficient energy-based max-margin training procedure for pft.

We perform comparison with FastText as well as existing density word embeddings w2g (Gaussian) and w2gm (Gaussian mixture). Our models extract high-quality semantics based on multiple word-similarity benchmarks, including the rare word dataset. We obtain an average weighted improvement of 3.7% over FastText [Bojanowski et al., 2016] and 3.1% over the dictionary-level density-

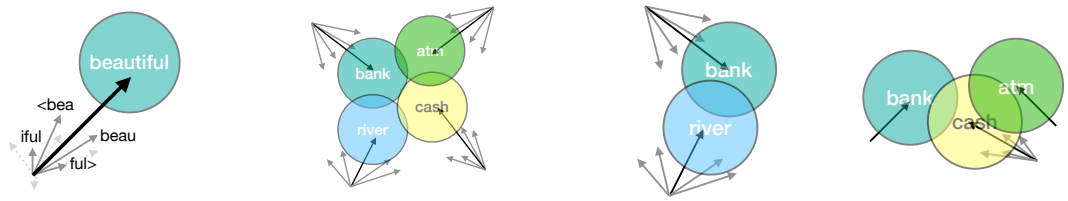
based models. We also observe meaningful nearest neighbors, particularly in the multimodal density case, where each mode captures a distinct meaning. Our models are also directly portable to foreign languages without any hyperparameter modification, where we observe strong performance, outperforming FastText on many foreign word similarity datasets. Our multimodal word representation can also disentangle meanings, and is able to separate different senses in foreign polysemies. In particular, our models attain state-of-the-art performance on SCWS, a benchmark to measure the ability to separate different word meanings, achieving 1.0% improvement over a recent density embedding model w2gm [Athiwaratkun and Wilson, 2017].

To the best of our knowledge, we are the first to develop multi-sense embeddings with high semantic quality for rare words. Our code and embeddings are publicly available.[1]

4.2 Probabilistic FastText

Probabilistic FastText combines a probabilistic word representation with the ability to capture subword structure. We describe the probabilistic subword representation in Section 4.2.1. We describe the loss function as well as other training steps in Section 4.2.2.

[1] https://github.com/benathi/multisense-prob-fasttext



(a) (b) (c)

Figure 4.1: (4.1a) a Gaussian component and its subword structure. The bold arrow represents the final mean vector, estimated from averaging the grey n-gram vectors. (4.1b) PFT-G model: Each Gaussian component's mean vector is a subword vector. (4.1c) PFT-GM model: For each Gaussian mixture distribution, one component's mean vector is estimated by a subword structure whereas other components are dictionary-based vectors.

4.2.1 Probabilistic Subword Representation

We represent each word with a Gaussian mixture with K Gaussian components. That is, a word w is associated with a density function $f(x) = \sum_{i=1}^{K} p_{w,i}\mathcal{N}(x; \vec{\mu}_{w,i}, \Sigma_{w,i})$ where $\{\mu_{w,i}\}_{k=1}^{K}$ are the mean vectors and $\{\Sigma_{w,i}\}$ are the covariance matrices, and $\{p_{w,i}\}_{k=1}^{K}$ are the component probabilities which sum to 1.

The mean vectors of Gaussian components hold much of the semantic information in density embeddings. While these models are successful based on word similarity and entailment benchmarks [Athiwaratkun and Wilson, 2017, Vilnis and McCallum, 2015], the mean vectors are often dictionary-level, which can lead to poor semantic estimates for rare words, or the inability to handle words outside the training corpus. We propose using subword structures to estimate the mean vectors. We outline the formulation below.

For word w, we estimate the mean vector μ_w with the average over n-gram

vectors and its dictionary-level vector. That is,

$$\mu_w = \frac{1}{|NG_w| + 1} \left(v_w + \sum_{g \in NG_w} z_g \right) \tag{4.1}$$

where z_g is a vector associated with an n-gram g, v_w is the dictionary representation of word w, and NG_w is a set of n-grams of word w. Examples of 3,4-grams for a word "beautiful", including the beginning-of-word character '⟨' and end-of-word character '⟩', are:

- 3-grams: ⟨be, bea, eau, aut, uti, tif, ful, ul⟩
- 4-grams: ⟨bea, beau .., iful ,ful⟩

This structure is similar to that of FastText [Bojanowski et al., 2016]; however, we note that FastText uses single-prototype deterministic embeddings as well as a training approach that maximizes the negative log-likelihood, whereas we use a multi-prototype probabilistic embedding and for training we maximize the similarity between the words' probability densities.

Figure 4.1a depicts the subword structure for the mean vector. Figure 4.1b and 4.1c depict our models, Gaussian probabilistic FastText (pft-g) and Gaussian mixture probabilistic FastText (pft-gm). In the Gaussian case, we represent each mean vector with a subword estimation. For the Gaussian mixture case, we represent one Gaussian component's mean vector with the subword structure whereas other components' mean vectors are dictionary-based. This model choice to use dictionary-based mean vectors for other components is to reduce to constraint imposed by the subword structure and promote independence for meaning discovery.

4.2.2 Loss Function

We adopt the loss function and a word sampling scheme as in Section 3.2.3. We use the energy similar to Section 3.2.4 but made a slight simplification for computational efficiency outlined below.

In theory, it can be beneficial to have covariance matrices as learnable parameters. In practice, Athiwaratkun and Wilson [2017] observe that spherical covariances often perform on par with diagonal covariances with much less computational resources. Using spherical covariances for each component, we can further simplify the energy function as follows:

$$\xi_{i,j} = -\frac{\alpha}{2} \cdot ||\mu_{f,i} - \mu_{g,j}||^2 , \tag{4.2}$$

where the hyperparameter α is the scale of the inverse covariance term in Equation 3.5. We note that Equation 4.2 is equivalent to Equation 3.5 up to an additive constant given that the covariance matrices are spherical and the same for all components.

4.3 Experiments

We have proposed a probabilistic FastText model which combines the flexibility of subword structure with the density embedding approach. In this section, we show that our probabilistic representation with subword mean vectors with the simplified energy function outperforms many word similarity baselines and provides disentangled meanings for polysemies.

First, we describe the training details in Section 4.3.1. We provide qualitative evaluation in Section 4.3.2, showing meaningful nearest neighbors for the Gaussian embeddings, as well as the ability to capture multiple meanings by Gaussian mixtures. Our quantitative evaluation in Section 4.3.3 demonstrates strong performance against the baseline models FastText [Bojanowski et al., 2016] and the dictionary-level Gaussian (w2g) [Vilnis and McCallum, 2015] and Gaussian mixture embeddings in Chapter 3 [Athiwaratkun and Wilson, 2017] (w2gm). We train our models on foreign language corpuses and show competitive results on foreign word similarity benchmarks in Section 4.3.4. Finally, we explain the importance of the n-gram structures for semantic sharing in Section 4.3.5.

4.3.1 Training Details

We train our models on both English and foreign language datasets. For English, we use the concatenation of ukWac and WackyPedia [Baroni et al., 2009] which consists of 3.376 billion words. We filter out word types that occur fewer than 5 times which results in a vocabulary size of 2,677,466.

For foreign languages, we demonstrate the training of our model on French, German, and Italian text corpuses. We note that our model should be applicable for other languages as well. We use FrWac (French), DeWac (German), ItWac (Italian) datasets [Baroni et al., 2009] for text corpuses, consisting of 1.634, 1.716 and 1.955 billion words respectively. We use the same threshold, filtering out words that occur less than 5 times in each corpus. We have dictionary sizes of 1.3, 2.7, and 1.4 million words for FrWac, DeWac, and ItWac.

We adjust the hyperparameters on the English corpus and use them for foreign

languages. Note that the adjustable parameters for our models are the loss margin m in Equation 3.2 and the scale α in Equation 4.2. We search for the optimal hyperparameters in a grid $m \in \{0.01, 0.1, 1, 10, 100\}$ and $\alpha \in \{\frac{1}{5\times10^{-3}}, \frac{1}{10^{-3}}, \frac{1}{2\times10^{-4}}, \frac{1}{1\times10^{-4}}\}$ on our English corpus. The hyperpameter α affects the scale of the loss function; therefore, we adjust the learning rate appropriately for each α. In particular, the learning rates used are $\gamma = \{10^{-4}, 10^{-5}, 10^{-6}\}$ for the respective α values.

Other fixed hyperparameters include the number of Gaussian components $K = 2$, the context window length $\ell = 10$ and the subsampling threshold $t = 10^{-5}$. We use character n-grams where $n = 3, 4, 5, 6$ to estimate the mean vectors.

4.3.2 Qualitative Evaluation - Nearest neighbors

We show that our embeddings learn the word semantics well by demonstrating meaningful nearest neighbors. Table 4.1 shows examples of polysemous words such as `rock`, `star`, and `cell`.

Word	Co.	Nearest Neighbors
rock	0	rock:0, rocks:0, rocky:0, mudrock:0, rockscape:0, boulders:0 , coutcrops:0,
rock	1	rock:1, punk:0, punk-rock:0, indie:0, pop-rock:0, pop-punk:0, indie-rock:0, band:1
bank	0	bank:0, banks:0, banker:0, bankers:0, bankcard:0, Citibank:0, debits:0
bank	1	bank:1, banks:1, river:0, riverbank:0, embanking:0, banks:0, confluence:1
star	0	stars:0, stellar:0, nebula:0, starspot:0, stars.:0, stellas:0, constellation:1
star	1	star:1, stars:1, star-star:0, 5-stars:0, movie-star:0, mega-star:0, super-star:0
cell	0	cell:0, cellular:0, acellular:0, lymphocytes:0, T-cells:0, cytes:0, leukocytes:0
cell	1	cell:1, cells:1, cellular:0, cellular-phone:0, cellphone:0, transcellular:0
left	0	left:0, right:1, left-hand:0, right-left:0, left-right-left:0, right-hand:0, leftwards:0
left	1	left:1, leaving:0, leavings:0, remained:0, leave:1, enmained:0, leaving-age:0, sadly-departed:0

Word	Nearest Neighbors
rock	rock, rock-y, rockn, rock-, rock-funk, rock/, lava-rock, nu-rock, rock-pop, rock/ice, coral-rock
bank	bank-, bank/, bank-account, bank., banky, bank-to-bank, banking, Bank, bank/cash, banks.**
star	movie-stars, star-planet, G-star, star-dust, big-star, starsailor, 31-star, star-lit, Star, starsign, pop-stars
cell	cellular, tumour-cell, in-cell, cell/tumour, 11-cell, T-cell, sperm-cell, 2-cells, Cell-to-cell
left	left, left/joined, leaving, left,right, right, left)and, leftsided, lefted, leftside

Table 4.1: Nearest neighbors of PFT-GM (top) and PFT-G (bottom). The notation `w:i` denotes the i^{th} mixture component of the word `w`.

Table 4.1 shows the nearest neighbors of polysemous words. We note that subword embeddings prefer words with overlapping characters as nearest neighbors. For instance, "rock-y", "rockn", and "rock" are both close to the word "rock". For the purpose of demonstration, we only show words with meaningful variations and omit words with small character-based variations previously mentioned. However, all words shown are in the top-100 nearest words.

We observe the separation in meanings for the multi-component case; for instance, one component of the word "bank" corresponds to a financial bank whereas the other component corresponds to a river bank. The single-component case also has interesting behavior. We observe that the subword embeddings of polysemous words can represent both meanings. For instance, both "lava-rock" and "rock-pop" are among the closest words to "rock".

4.3.3 Word Similarity Evaluation

D	50				300				
	W2G	W2GM	PFT-G	PFT-GM	FASTTEXT	W2G	W2GM	PFT-G	PFT-GM
SL-999	29.35	29.31	27.34	**34.13**	38.03	38.84	**39.62**	35.85	39.60
WS-353	71.53	**73.47**	67.17	71.10	73.88	78.25	**79.38**	73.75	76.11
MEN-3K	72.58	73.55	70.61	**73.90**	76.37	78.40	78.76	77.78	**79.65**
MC-30	76.48	79.08	73.54	**79.75**	81.20	82.42	**84.58**	81.90	80.93
RG-65	73.30	74.51	70.43	**78.19**	79.98	80.34	**80.95**	77.57	79.81
YP-130	41.96	**45.07**	37.10	40.91	53.33	46.40	47.12	48.52	**54.93**
MT-287	64.79	66.60	63.96	**67.65**	67.93	67.74	**69.65**	66.41	69.44
MT-771	60.86	60.82	60.40	**63.86**	66.89	70.10	**70.36**	67.18	69.68
RW-2K	28.78	28.62	44.05	**42.78**	48.09	35.49	42.73	**50.37**	49.36
AVG.	42.32	42.76	44.35	**46.47**	49.28	47.71	49.54	49.86	**51.10**

Table 4.2: Spearman's Correlation $\rho \times 100$ on Word Similarity Datasets.

We evaluate our embeddings on several standard word similarity datasets, namely, SL-999 [Hill et al., 2014], WS-353 [Finkelstein et al., 2002], MEN-3k [Bruni et al., 2014], MC-30 [Miller and Charles, 1991], RG-65 [Rubenstein and Goodenough,

1965], YP-130 [Yang and Powers, 2006], MTurk(-287,-771) [Halawi et al., 2012, Radinsky et al., 2011], and RW-2k [Luong et al., 2013]. Each dataset contains a list of word pairs with a human score of how related or similar the two words are. We use the notation DATASET-NUM to denote the number of word pairs NUM in each evaluation set. We note that the dataset RW focuses more on infrequent words and SimLex-999 focuses on the similarity of words rather than relatedness. We also compare PFT-GM with other multi-prototype embeddings in the literature using SCWS [Huang et al., 2012], a word similarity dataset that is aimed to measure the ability of embeddings to discern multiple meanings.

We calculate the Spearman correlation [Spearman, 1904] between the labels and our scores generated by the embeddings. The Spearman correlation is a rank-based correlation measure that assesses how well the scores describe the true labels. The scores we use are cosine-similarity scores between the mean vectors. In the case of Gaussian mixtures, we use the pairwise maximum score:

$$s(f,g) = \max_{i\in 1,\ldots,K} \max_{j\in 1,\ldots,K} \frac{\mu_{f,i}\cdot\mu_{g,j}}{||\mu_{f,i}||\cdot||\mu_{g,j}||}. \tag{4.3}$$

The pair (i,j) that achieves the maximum cosine similarity corresponds to the Gaussian component pair that is the closest in meanings. Therefore, this similarity score yields the most related senses of a given word pair. This score reduces to a cosine similarity in the Gaussian case ($K = 1$).

Comparison Against Dictionary-Level Density Embeddings and FastText

We compare our models against the dictionary-level Gaussian and Gaussian mixture embeddings in Table 4.2, with 50-dimensional and 300-dimensional mean vectors.

The 50-dimensional results for W2G and W2GM are obtained directly from Chapter 3 [Athiwaratkun and Wilson, 2017]. For comparison, we use the public code[2] to train the 300-dimensional W2G and W2GM models and the publicly available FASTTEXT model[3].

We calculate Spearman's correlations for each of the word similarity datasets. These datasets vary greatly in the number of word pairs; therefore, we mark each dataset with its size for visibility. For a fair and objective comparison, we calculate a weighted average of the correlation scores for each model.

Our PFT-GM achieves the highest average score among all competing models, outperforming both FASTTEXT and the dictionary-level embeddings W2G and W2GM. Our unimodal model PFT-G also outperforms the dictionary-level counterpart W2G and FASTTEXT. We note that the model W2GM appears quite strong according to Table 4.2, beating PFT-GM on many word similarity datasets. However, the datasets that W2GM performs better than PFT-GM often have small sizes such as MC-30 or RG-65, where the Spearman's correlations are more subject to noise. Overall, PFT-GM outperforms W2GM by 3.1% and 8.7% in 300 and 50 dimensional models. In addition, PFT-G and PFT-GM also outperform FASTTEXT by 1.2% and 3.7% respectively.

Comparison Against Multi-Prototype Models

In Table 4.3, we compare 50 and 300 dimensional PFT-GM models against the multi-prototype embeddings described in Section 2.1.3 and the existing multimodal density embeddings W2GM. We use the word similarity dataset SCWS [Huang et al.,

[2] https://github.com/benathi/word2gm

[3] https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.en.zip

Model	Dim	$\rho \times 100$
Huang AvgSim	50	62.8
Tian MaxSim	50	63.6
w2gm MaxSim	50	62.7
Neelakantan AvgSim	50	64.2
pft-gm MaxSim	50	63.7
Chen-M AvgSim	200	66.2
w2gm MaxSim	200	65.5
Neelakantan AvgSim	300	**67.2**
w2gm MaxSim	300	66.5
pft-gm MaxSim	300	**67.2**

Table 4.3: Spearman's Correlation $\rho \times 100$ on word similarity dataset SCWS.

2012] which contains words with potentially many meanings, and is a benchmark for distinguishing senses. We use the maximum similarity score (Equation 4.3), denoted as MaxSim. AveSim denotes the average of the similarity scores, rather than the maximum.

We outperform the dictionary-based density embeddings w2gm in both 50 and 300 dimensions, demonstrating the benefits of subword information. Our model achieves state-of-the-art results, similar to that of Neelakantan et al. [2014].

Lang.	Evaluation	FastText	w2g	w2gm	pft-g	pft-gm
FR	WS353	38.2	16.73	20.09	41.0	**41.3**
DE	GUR350	70	65.01	69.26	77.6	**78.2**
	GUR65	81	74.94	76.89	81.8	**85.2**
IT	WS353	57.1	56.02	61.09	60.2	**62.5**
	SL-999	29.3	29.44	**34.91**	29.3	33.7

Table 4.4: Word similarity evaluation on foreign languages.

4.3.4 Evaluation on Foreign Language Embeddings

We evaluate the foreign-language embeddings on word similarity datasets in respective languages. We use Italian WordSim353 and Italian SimLex-999 [Leviant and Reichart, 2015] for Italian models, GUR350 and GUR65 [Gurevych, 2005] for German models, and French WordSim353 [Finkelstein et al., 2002] for French models. For datasets GUR350 and GUR65, we use the results reported in the FastText publication [Bojanowski et al., 2016]. For other datasets, we train FastText models for comparison using the public code[4] on our text corpuses. We also train dictionary-level models w2g, and w2gm for comparison.

Table 4.4 shows the Spearman's correlation results of our models. We outperform FastText on many word similarity benchmarks. Our results are also significantly better than the dictionary-based models, w2g and w2gm. We hypothesize that w2g and w2gm can perform better than the current reported results given proper pre-processing of words due to special characters such as accents.

We investigate the nearest neighbors of polysemies in foreign languages and also observe clear sense separation. For example, *piano* in Italian can mean "floor" or "slow". These two meanings are reflected in the nearest neighbors where one component is close to *piano-piano*, *pianod* which mean "slowly" whereas the other component is close to *piani* (floors), *istrutturazione* (renovation) or *infrastruttre* (infrastructure). Table 4.5 shows additional results, demonstrating that the disentangled semantics can be observed in multiple languages.

[4] https://github.com/facebookresearch/fastText.git

Word	Meaning	Nearest Neighbors
(IT) *secondo*	2nd	Secondo (2nd), terzo (3rd) , quinto (5th), primo (first), quarto (4th), ultimo (last)
(IT) *secondo*	according to	conformit (compliance), attenendosi (following), cui (which), conformemente (accordance with)
(IT) *porta*	lead, bring	portano (lead), conduce (leads), portano, porter, portando (bring), costringe (forces)
(IT) *porta*	door	porte (doors), finestrella (window), finestra (window), portone (doorway), serratura (door lock)
(FR) *voile*	veil	voiles (veil), voiler (veil), voilent (veil), voilement, foulard (scarf), voils (veils), voilant (veiling)
(FR) *voile*	sail	catamaran (catamaran), driveur (driver), nautiques (water), Voile (sail), driveurs (drivers)
(FR) *temps*	weather	brouillard (fog), orageuses (stormy), nuageux (cloudy)
(FR) *temps*	time	mi-temps (half-time), partiel (partial), Temps (time), annualis (annualized), horaires (schedule)
(FR) *voler*	steal	envoler (fly), voleuse (thief), cambrioler (burgle), voleur (thief), violer (violate), picoler (tipple)
(FR) *voler*	fly	airs (air), vol (flight), volent (fly), envoler (flying), atterrir (land)

Table 4.5: Nearest neighbors of polysemies based on our foreign language PFT-GM models.

4.3.5 Qualitative Evaluation - Subword Decomposition

One of the motivations for using subword information is the ability to handle out-of-vocabulary words. Another benefit is the ability to help improve the semantics of rare words via subword sharing. Due to an observation that text corpuses follow Zipf's power law [Zipf, 1949], words at the tail of the occurrence distribution appears much less frequently. Training these words to have a good semantic representation is challenging if done at the word level alone. However, an n-gram such as 'abnorm' is trained during both occurrences of "abnormal" and "abnormality" in the corpus, hence further augments both words's semantics.

Figure 4.2 shows the contribution of n-grams to the final representation. We filter out to show only the n-grams with the top-5 and bottom-5 similarity scores. We observe that the final representations of both words align with n-grams "abno", "bnor", "abnorm", "anbnor", "<abn". In fact, both "abnormal" and "abnormality" share the same top-5 n-grams. Due to the fact that many rare words such as "autobiographer", "circumnavigations", or "hypersensitivity" are composed from many common sub-words, the n-gram structure can help improve the representation quality.

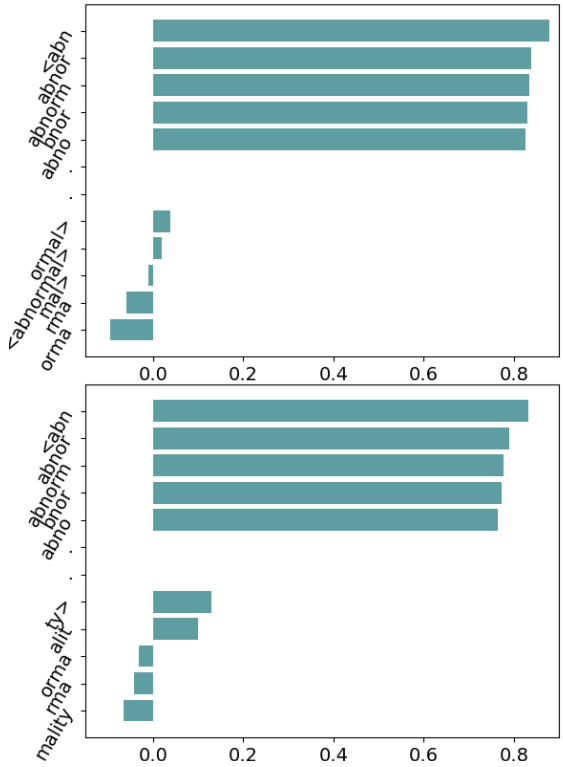


Figure 4.2: Contribution of each n-gram vector to the final representation for word "abnormal" (top) and "abnormality" (bottom). The x-axis is the cosine similarity between each n-gram vector $z_g^{(w)}$ and the final vector μ_w.

4.4 Numbers of Components

It is possible to train our approach with $K > 2$ mixture components; however, Athiwaratkun and Wilson [2017] observe that dictionary-level Gaussian mixtures with $K = 3$ do not overall improve word similarity results, even though these mixtures can discover 3 distinct senses for certain words. Indeed, while $K > 2$ in principle allows for greater flexibility than $K = 2$, most words can be very flexibly modelled with a mixture of two Gaussians, leading to $K = 2$ representing a good balance between flexibility and Occam's razor.

Even for words with single meanings, our PFT model with $K = 2$ often

learns richer representations than a $K = 1$ model. For example, the two mixture components can learn to cluster together to form a more heavy tailed unimodal distribution which captures a word with one dominant meaning but with close relationships to a wide range of other words.

In addition, we observe that our model with K components can capture more than K meanings. For instance, in $K = 1$ model, the word pairs ("cell", "jail") and ("cell", "biology") and ("cell", "phone") will all have positive similarity scores based on $K = 1$ model. In general, if a word has multiple meanings, these meanings are usually compressed into the linear substructure of the embeddings [Arora et al., 2016]. However, the pairs of non-dominant words often have lower similarity scores, which might not accurately reflect their true similarities.

4.5 Discussion

We have proposed models for probabilistic word representations equipped with flexible sub-word structures, suitable for rare and out-of-vocabulary words. The proposed probabilistic formulation incorporates uncertainty information and naturally allows one to uncover multiple meanings with multimodal density representations. Our models offer better semantic quality, outperforming competing models on word similarity benchmarks. Moreover, our multimodal density models can provide interpretable and disentangled representations, and are the first multi-prototype embeddings that can handle rare words.

Future work includes an investigation into the trade-off between learning full covariance matrices for each word distribution, computational complexity, and performance. This direction can potentially have a great impact on tasks where the

variance information is crucial, such as for hierarchical modeling with probability distributions [Athiwaratkun and Wilson, 2018].

Other future work involves co-training PFT on many languages. Currently, existing work on multi-lingual embeddings align the word semantics on pre-trained vectors [Ammar et al., 2016, Conneau et al., 2018, Smith et al., 2017], which can be suboptimal due to polysemies. We envision that the multi-prototype nature can help disambiguate words with multiple meanings and facilitate semantic alignment.

CHAPTER 5

PROBABILISTIC REPRESENTATION FOR HIERARCHICAL DATA

By representing words with probability densities rather than point vectors, probabilistic word embeddings can capture rich and interpretable semantic information and uncertainty. The uncertainty information can be particularly meaningful in capturing *entailment* relationships – whereby general words such as "entity" correspond to broad distributions that encompass more specific words such as "animal" or "instrument". We introduce *density order embeddings*, which learn hierarchical representations through encapsulation of probability densities. In particular, we propose simple yet effective loss functions and distance metrics, as well as graph-based schemes to select negative samples to better learn hierarchical density representations. Our approach provides state-of-the-art performance on the WordNet hypernym relationship prediction task and the challenging HyperLex [Vulić et al., 2017] lexical entailment dataset – while retaining a rich and interpretable density representation.

5.1 Introduction

Entailment patterns can be observed from density word embeddings through *unsupervised* training based on word contexts [Athiwaratkun and Wilson, 2017, Vilnis and McCallum, 2015]. In the unsupervised settings, density embeddings are learned via maximizing the similarity scores between nearby words. In these cases, the density encapsulation behavior arises due to the word occurrence pattern that a general word can often substitute more specific words; for instance, the word "tea"

in a sentence "I like iced tea" can be substituted by "beverages", yielding another natural sentence "I like iced beverages". Therefore, the probability density of a general concept such as "beverages" tends to have a larger variance than specific ones such as "tea", reflecting higher uncertainty in meanings since a general word can be used in many contexts. However, the information from word occurrences alone is not sufficient to train meaningful embeddings of some concepts. For instance, it is fairly common to observe sentences "Look at the cat", or "Look at the dog", but not "Look at the mammal". Therefore, due to the way we typically express natural language, it is unlikely that the word "mammal" would be learned as a distribution that encompasses both "cat" and "dog", since "mammal" rarely occurs in similar contexts.

Rather than relying on the information from word occurrences, one can do *supervised* training of density embeddings on hierarchical data. In this chapter, we propose new training methodology to enable effective supervised probabilistic density embeddings. Despite providing rich and intuitive word representations, with a natural ability to represent order relationships, probabilistic embeddings have only been considered in a small number of pioneering works such as Vilnis and McCallum [2015], and these works are almost exclusively focused on *unsupervised embeddings*. Probabilistic Gaussian embeddings trained directly on labeled data have been briefly considered but perform surprisingly poorly compared to other competing models [Vendrov et al., 2016, Vulić et al., 2017].

Our work reaches a very different conclusion: probabilistic Gaussian embeddings can be *highly effective* at capturing ordering and are suitable for modeling hierarchical structures, and can even achieve state-of-the-art results on hypernym prediction and graded lexical entailment tasks, so long as one uses the right training

procedures.

In particular, we make the following contributions.

(a) We adopt a new form of loss function for training hierarchical probabilistic order embeddings.

(b) We introduce the notion of soft probabilistic encapsulation orders and a thresholded divergence-based penalty function, which do not over-penalize words with a sufficient encapsulation.

(c) We introduce a new graph-based scheme to select negative samples to contrast the true relationship pairs during training. This approach incorporates hierarchy information to the negative samples that help facilitate training and has added benefits over the hierarchy-agnostic sampling schemes previously used in literature.

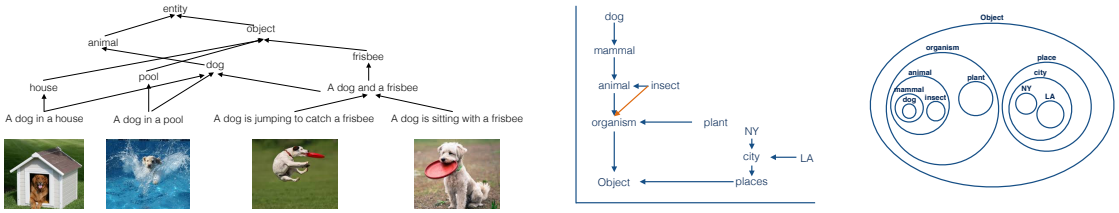
(d) We also demonstrate that initializing the right variance scale is highly important for modeling hierarchical data via distributions, allowing the model to exhibit meaningful encapsulation orders.

The outline of this chapter is as follows. In Section 2.1.4, we introduce the background for Gaussian embeddings [Vilnis and McCallum, 2015] and vector order embeddings [Vendrov et al., 2016]. We describe our training methodology in Section 5.2, where we introduce the notion of soft encapsulation orders (Section 5.2.2) and explore different divergence measures such as the expected likelihood kernel, KL divergence, and a family of Rényi alpha divergences (Section 5.2.3). We describe the experiment details in Section 5.3 and offer a qualitative evaluation of the model in Section 5.3.4, where we show the visualization of the density encapsulation

behavior. We show quantitative results on the WordNet Hypernym prediction task in Section 5.3.3 and a graded entailment dataset HyperLex in Section 5.3.5.

In addition, we conduct experiments to show that our proposed changes to learn Gaussian embeddings contribute to the increased performance. We demonstrate (a) the effects of our loss function in Section 5.3.6, (b) soft encapsulation in Section 5.3.6, (c) negative sample selection in Section 5.3.5], and (d) initial variance scale in Section 5.3.6.

We make our code publicly available.[1]



(a) (b) (c)

Figure 5.1: (a) Captions and images in the visual-semantic hierarchy. (b) Vector order embedding [Vendrov et al., 2016] where specific entities have higher coordinate values. (c) Density order embedding where specific entities correspond to concentrated distributions encapsulated in broader distributions of general entities.

5.2 Methodology

In Section 5.2.1, we describe the ideal partial orders that can be induced by density encapsulation. Note that the concept of partial orders is discussed previously in Section 2.2.1. Section 5.2.2 describes our training approach that softens the notion of strict encapsulation with a viable penalty function.

[1] https://github.com/benathi/density-order-emb

5.2.1 Strict Encapsulation Partial Orders

A partial order on probability densities can be obtained by the notion of encapsulation. That is, a density f is more specific than a density g if f is encompassed in g. The degree of encapsulation can vary, which gives rise to multiple order relations. We define an order relation $\preceq_\eta$ for $\eta \geq 0$ where η indicates the degree of encapsulation required for one distribution to entail another. More precisely, for distributions f and g,

$$f \preceq_\eta g \Leftrightarrow \{x : f(x) > \eta\} \subseteq \{x : g(x) > \eta\}. \tag{5.1}$$

Note that $\{x : f(x) > \eta\}$ is a set where the density f is greater than the threshold η. The relation in Equation 5.1 says that f entails g if and only if the set of g contains that of f. In Figure 5.2, we depict two Gaussian distributions with different mean vectors and covariance matrices. Figure 5.2 (left) shows the density values of distributions f (narrow, blue) and g (broad, orange) and different threshold levels. Figure 5.2 (right) shows that different η's give rise to different partial orders. For instance, we observe that neither $f \preceq_{\eta_1} g$ nor $g \preceq_{\eta_1} f$ but $f \preceq_{\eta_3} g$.

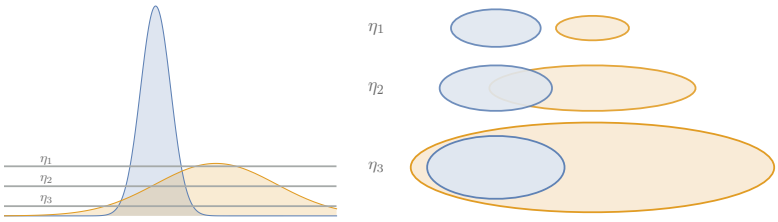


Figure 5.2: Strict encapsulation orders induced by different η values.

5.2.2 Soft Encapsulation Orders

A plausible penalty function for the order relation $f \preceq_\eta g$ is a set measure on $\{x : f(x) > \eta\} - \{x : g(x) > \eta\}$. However, this penalty is difficult to evaluate for

most distributions, including Gaussians. Instead, we use simple penalty functions based on asymmetric divergence measures between probability densities. Divergence measures $D(\cdot||\cdot)$ have a property that $D(f||g) = 0$ if and only if $f = g$. Using $D(\cdot||\cdot)$ to represent order violation is undesirable since the penalty should be 0 if $f \neq g$ but $f \preceq g$. Therefore, we propose using a thresholded divergence

$$d_\gamma(f, g) = \max(0, D(f||g) - \gamma), \tag{5.2}$$

which can be zero if f is properly encapsulated in g. We discuss the effectiveness of using divergence thresholds in Section 5.3.6.

We note that by using $d_\gamma(\cdot, \cdot)$ as a violation penalty, we no longer have the strict *partial order*. In particular, the notion of transitivity in a partial order is not guaranteed. For instance, if $f \preceq g$ and $g \preceq h$, our density order embeddings would yield $d_\gamma(f, g) = 0$ and $d_\gamma(g, h) = 0$. However, it is not necessarily the case that $d_\gamma(f, h) = 0$ since $D(f||h)$ can be greater than γ. This is not a drawback since a high value of $D(f||h)$ reflects that the hypernym relationship is not direct, requiring many edges from f to h in the hierarchy. The extent of encapsulation contains useful entailment information, as demonstrated in Section 5.3.5 where our model scores highly correlate with the annotated scores of a challenging lexical entailment dataset and achieves state-of-the-art results.

Another property, antisymmetry, does not strictly hold since if $d_\gamma(f, g) = 0$ and $d_\gamma(g, f) = 0$ does not imply $f = g$. However, in this situation, it is necessary that f and g overlap significantly if γ is small. Due to the fact that the $d_\gamma(\cdot, \cdot)$ does not strictly induce a partial order, we refer to this model as *soft density order embeddings* or simply *density order embeddings*.

5.2.3 Divergence Measures

Asymmetric Divergence

Kullback-Leibler (KL) Divergence The KL divergence is an asymmetric measure of the difference between probability distributions. For distributions f and g, $\mathrm{KL}(g||f) \equiv \int g(x) \log \frac{g(x)}{f(x)}\, dx$ imposes a high penalty when there is a region of points x such that the density $f(x)$ is low but $g(x)$ is high. An example of such a region is the area on the left of f in Figure 5.2. This measure penalizes the situation where f is a concentrated distribution relative to g; that is, if the distribution f is encompassed by g, then the KL yields high penalty. For d-dimensional Gaussians $f = \mathcal{N}_d(\mu_f, \Sigma_f)$ and $g = \mathcal{N}_d(\mu_g, \Sigma_g)$,

$$2D_{KL}(f||g) = \log(\det(\Sigma_g)/\det(\Sigma_f)) - d + \mathrm{tr}(\Sigma_g^{-1}\Sigma_f) + (\mu_f - \mu_g)^T \Sigma_g^{-1}(\mu_f - \mu_g) \quad (5.3)$$

Rényi α-Divergence is a general family of divergence with varying scale of zero-forcing penalty [Renyi, 1961]. Equation 5.4 describes the general form of the α-divergence for $\alpha \neq 0, 1$ [Liese and Vajda, 1987]. We note that for $\alpha \to 0$ or 1, we recover the KL divergence and the reverse KL divergence; that is, $\lim_{\alpha \to 1} D_\alpha(f||g) = \mathrm{KL}(f||g)$ and $\lim_{\alpha \to 0} D_\alpha(f||g) = \mathrm{KL}(g||f)$ [Pardo, 2006]. The α-divergences are asymmetric for all α's, except for $\alpha = 1/2$.

$$D_\alpha(f||g) = \frac{1}{\alpha(\alpha - 1)} \log \left(\int \frac{f(x)^\alpha}{g(x)^{\alpha - 1}}\, dx \right) \quad (5.4)$$

For two multivariate Gaussians f and g, we can write the Rényi divergence as [Pardo, 2006]:

$$2D_\alpha(f||g) = -\frac{1}{\alpha(\alpha - 1)} \log \frac{\det(\alpha\Sigma_g + (1-\alpha)\Sigma_f)}{\left(\det(\Sigma_f)^{1-\alpha} \cdot \det(\Sigma_g)^\alpha\right)} + (\mu_f - \mu_g)^T (\alpha\Sigma_g + (1-\alpha)\Sigma_f)^{-1} (\mu_f - \mu_g). \quad (5.5)$$

The parameter α controls the degree of *zero forcing* where minimizing $D_\alpha(f||g)$ for high α results in f being more concentrated to the region of g with high density. For low α, f tends to be *mass-covering*, encompassing regions of g including the low density regions. Recent work by Li and Turner [2016] demonstrates that different applications can require different degrees of zero-forcing penalty.

Symmetric Divergence

Expected Likelihood Kernel The expected likelihood kernel (ELK) [Jebara et al., 2004] is a symmetric measure of affinity, define as $K(f,g) = \langle f, g\rangle_{\mathcal{H}}$. For two Gaussians f and g,

$$2\log\langle f,g\rangle_{\mathcal{H}} = -\log\det(\Sigma_f+\Sigma_g) - d\log(2\pi) - (\mu_f-\mu_g)^T(\Sigma_f+\Sigma_g)^{-1}(\mu_f-\mu_g) \quad (5.6)$$

Since this kernel is a similarity score, we use its negative as our penalty. That is, $D_{\text{ELK}}(f||g) = -2\log\langle f,g\rangle_{\mathcal{H}}$. Intuitively, the asymmetric measures should be more successful at training density order embeddings. However, a symmetric measure can result in a correct encapsulation order as well, since a general entity often has to minimize the penalty with many specific elements and consequently ends up having a broad distribution to lower the average loss. The expected likelihood kernel is used to train Gaussian and Gaussian Mixture word embeddings on a large text corpus [Athiwaratkun and Wilson, 2017, Vilnis and McCallum, 2015] where the model performs well on the word entailment dataset [Baroni et al., 2012].

5.2.4 Loss Function

To learn our density embeddings, we use a loss function similar to that of Vendrov et al. [2016]. Minimizing this function (Equation 5.7) is equivalent to minimizing

the penalty between a true relationship pair (u, v) where $u \preceq v$, but pushing the penalty to be above a margin m for the negative example (u', v') where $u' \not\preceq v'$:

$$\sum_{(u,v)\in\mathcal{D}} d(u, v) + \max\{0, m - d(u', v')\} \tag{5.7}$$

We note that this loss function is different than the rank-margin loss introduced in the original Gaussian embeddings (Equation 2.3). Equation 5.7 aims to reduce the dissimilarity of a true relationship pair $d(u, v)$ with no constraint, unlike in Equation 2.3, which becomes zero if $d(u, v)$ is above $d(u', v')$ by margin m.

5.2.5 Selecting Negative Samples

In many embedding models such as WORD2VEC [Mikolov et al., 2013a] or Gaussian embeddings [Vilnis and McCallum, 2015], negative samples are often used in the training procedure to contrast with true samples from the dataset. For flat data such as words in a text corpus, negative samples are selected randomly from a unigram distribution. We propose new graph-based methods to select negative samples that are suitable for hierarchical data, as demonstrated by the improved performance of our density embeddings. In our experiments, we use various combinations of the following methods.

Method **S1**: A simple negative sampling procedure used by Vendrov et al. [2016] is to replace a true hypernym pair (u, v) with either (u, v') or (u', v) where u', v' are randomly sampled from a uniform distribution of vertices. Method **S2**: We use a negative sample (v, u) if (u, v) is a true relationship pair, to make $D(v||u)$ higher than $D(u||v)$ in order to distinguish the directionality of density encapsulation. Method **S3**: It is important to increase the divergence between neighbor entities that do not entail each other. Let $A(w)$ denote all descendants of w in the training

set $\mathcal{D}$, including w itself. We first randomly sample an entity $w \in \mathcal{D}$ that has at least 2 descendants and randomly select a descendant $u \in A(w) - \{w\}$. Then, we randomly select an entity $v \in A(w) - A(u)$ and use the random neighbor pair (v, u) as a negative sample. Note that we can have $u \preceq v$, in which case the pair (v, u) is a reverse relationship. Method **S4**: Same as **S3** except that we sample $v \in A(w) - A(u) - \{w\}$, which excludes the possibility of drawing (w, u).

5.3 Experiments

We have introduced density order embeddings (DOE) to model hierarchical data via encapsulation of probability densities. We propose using a new loss function, graph-based negative sample selections, and a penalty relaxation to induce soft partial orders. In this section, we show the effectiveness of our model on WordNet hypernym prediction and a challenging graded lexical entailment task, where we achieve state-of-the-art performance.

First, we provide the training details in Section 5.3.1 and describe the hypernym prediction experiment in 5.3.3. We offers insights into our model with the qualitative analysis and visualization in Section 5.3.4. We evaluate our model on HyperLex, a lexical entailment dataset in Section 5.3.5.

5.3.1 Training Details

We have a similar data setup to the experiment by Vendrov et al. [2016] where we use the transitive closure of WordNet noun hypernym relationships which contains $82,115$ synsets and $837,888$ hypernym pairs from $84,427$ direct hypernym

edges. We obtain the data using the WordNet API of NLTK version 3.2.1 [Loper and Bird, 2002].

The validation set contains 4000 true hypernym relationships as well as 4000 false hypernym relationships where the false hypernym relationships are constructed from the **S1** negative sampling described in Section 5.2.5. The same process applies for the test set with another set of 4000 true hypernym relationships and 4000 false hypernym relationships.

We use d-dimensional Gaussian distributions with diagonal covariance matrices. We use $d = 50$ as the default dimension and analyze the results using different d's in Section 5.3.6. We initialize the mean vectors to have a unit norm and normalize the mean vectors in the training graph. We initialize the diagonal variance components to be all equal to β and optimize on the unconstrained space of $\log(\Sigma)$. We discuss the important effects of the initial variance scale in Section 5.3.6.

We use a minibatch size of 500 true hypernym pairs and use varying number of negative hypernym pairs, depending on the negative sample combination proposed in Section 5.2.5. We discuss the results for many selection strategies in Section 5.3.5. We also experiment with multiple divergence measures $D(\cdot||\cdot)$ described in Section 5.2.3. We use $D(\cdot||\cdot) = D_{KL}(\cdot||\cdot)$ unless stated otherwise. Section 5.3.6 considers the results using the α-divergence family with varying degrees of zero-forcing parameter α's. We use the Adam optimizer [Kingma and Ba, 2014] and train our model for at most 20 epochs. For each energy function, we tune the hyperparameters on grids. The hyperparameters are the loss margin m, the initial variance scale β, and the energy threshold γ. We evaluate the results by computing the penalty on the validation set to find the best threshold for binary classification, and use this threshold to perform prediction on the test set. Section 5.3.2 describes

the hyperparameters for all our models.

5.3.2 Model Hyperparameters

In Section 5.3.4, the 2−dimensional Gaussian model is trained with **S-1** method where the number of negative samples is equal to the number of positive samples. The best hyperparameters for $d = 2$ model is $(m, \beta, \gamma) = (100.0, 2 \times 10^{-4}, 3.0)$.

In Section 5.3.3, the best hyperparameters (m, β, γ) for each of our model are as follows: For Gaussian with KL penalty: $(2000.0, 5 \times 10^{-5}, 500.0)$, , Gaussian with reversed KL penalty: $(1000.0, 1 \times 10^{-4}, 1000.0)$, Gaussian with ELK penalty $(1000, 1 \times 10^{-5}, 10)$.

In Section 5.3.5, we use the same hyperparameters as in 5.3.3 with KL penalty, but a different negative sample combination in order to increase the distinguishability of divergence scores. For each positive sample in the training set, we use one sample from each of the methods **S1**, **S2**, **S4**. We note that the model from Section 5.3.3, using **S1** with the KL penalty obtains a Spearman's correlation of 0.527.

5.3.3 Hypernym Prediction

We show the prediction accuracy results on the test set of WordNet hypernyms in Table 5.1. We compare our results with **vector order-embeddings** (VOE) by Vendrov et al. [2016] (VOE model details are in Section 2.2.1). Another important baseline is the **transitive closure**, which requires no learning and classifies if a held-out edge is a hypernym relationship by determining if it is in the union of the training edges. **word2gauss** and **word2gauss†** are the Gaussian embeddings

trained using the loss function in Vilnis and McCallum [2015] (Equation 2.3) where **word2gauss** is the result reported by Vendrov et al. [2016] and **word2gauss†** is the best performance of our replication (see Section 5.3.6 for more details). Our density order embedding (DOE) outperforms the implementation by Vilnis and McCallum [2015] significantly; this result highlights the fact that our different approach for training Gaussian embeddings can be crucial to learning hierarchical representations.

We observe that the symmetric model (ELK) performs quite well for this task despite the fact that the symmetric metric cannot capture directionality. In particular, ELK can accurately detect pairs of concepts with no relationships when they're far away in the density space. In addition, for pairs that are related, ELK can detect pairs that overlap significantly in density space. The lack of directionality has more pronounced effects in the graded lexical entailment task (Section 5.3.5) where we observe a high degradation in performance if ELK is used instead of KL.

We find that our method outperforms vector order embeddings (VOE) [Vendrov et al., 2016]. We also find that DOE is very strong in a 2-dimensional Gaussian embedding example, trained for the purpose of visualization in Section 5.3.4, despite only having only 4 parameters: 2 from 2-dimensional μ and another 2 from the diagonal Σ. The results of DOE using a symmetric measure also outperforms the baselines on this experiment, but has a slightly lower accuracy than the asymmetric model.

Figure 5.3 offers an explanation as to why our density order embeddings might be easier to learn, compared to the vector counterpart. In certain cases such as fitting a general concept `entity` to the embedding space, we simply need to adjust the distribution of `entity` to be broad enough to encompass all other concepts. In

Table 5.1: Classification accuracy on hypernym relationship test set from WordNet.

Method	Test Accuracy (%)
transitive closure	88.2
word2gauss [Vilnis and McCallum, 2015]	86.6
word2gauss†	88.6
VOE (symmetric) [Vendrov et al., 2016]	84.2
VOE [Vendrov et al., 2016]	90.6
POE [Lai and Hockenmaier, 2017]	91.6
Li et al. [2017]	91.3
Box Embeddings [Li et al., 2019]	92.2
Smoothed Box Embeddings [Li et al., 2019]	92.0
DOE (ELK)	92.1
DOE (KL, reversed)	83.2
DOE (KL)	**92.3**
DOE (KL, $d = 2$)	89.2

the vector counterpart, it might be required to shift many points further from the origin to accommodate `entity` to reduce cascading order violations.

Figure 5.3: (**Left**) Adding a concept `entity` to vector order embedding (**Right**) Adding a concept `entity` to density order embedding

5.3.4 Qualitative Analysis

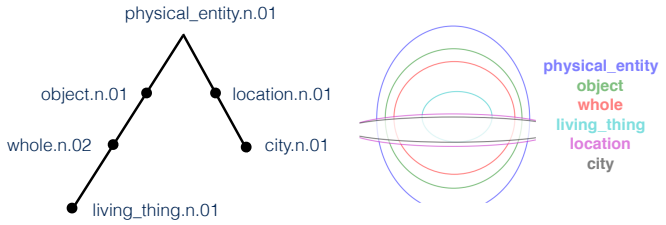
For qualitative analysis, we additionally train a 2-dimensional Gaussian model for visualization. Our qualitative analysis shows that the encapsulation behavior can be observed in the trained model. Figure 5.4 demonstrates the ordering of synsets in the density space. Each ellipse represents a Gaussian distribution where the center is given by the mean vector μ and the major and minor axes are given by the diagonal standard deviations $\sqrt{\Sigma}$, scaled by 300 for the x axis and 30 for the y

axis, for visibility.

Most hypernym relationships exhibit encapsulation behavior where the hypernym encompasses the synset that entails it. For instance, the distribution of `whole.n.02` is subsumed in the distribution of `physical_entity.n.01`. Note that `location.n.01` is not entirely encapsulated by `physical_entity.n.01` under this visualization. However, we can still predict which entity should be the hypernym among the two since the KL divergence of one given another would be drastically different. This is because a large part of `physical_entity.n.01` has considerable density at the locations where location.n.01 has very low density. This causes KL(`physical_entity.n.01` || `location.n.01`) to be very high (5103) relative to KL(`location.n.01` || `physical_entity.n.01`) (206). Table 5.2 shows the KL values for all pairs where we note that the numbers are from the full model ($d = 50$).

Another interesting pair is `city.n.01` $\preceq$ `location.n.01` where we see the two distributions have very similar contours and the encapsulation is not as distinct. In our full model $d = 50$, the distribution of `location.n.01` encompasses `city.n.01`'s, indicated by low KL(`city.n.01`||`location.n.01`) but high KL(`location.n.01`||`city.n.01`).

Figure 5.4 (**Right**) demonstrates the idea that synsets on the top of the hypernym hierarchy usually have higher "volume". A convenient metric that reflects this quantity is $\log\det(\Sigma)$ for a Gaussian distribution with covariance Σ. We can see that the synset, `physical_entity.n.01`, being the hypernym of all the synsets shown, has the highest $\log\det(\Sigma)$ whereas entities that are more specific such as `object.n.01`, `whole.n.02` and `living_thing` have decreasingly lower volume.



Synset	$\log \lvert\Sigma\rvert$
physical_entity.n.01	-219.7
object.n.01	-224.0
whole.n.02	-233.4
living_thing.n.01	-238.3
location.n.01	-261.8
city.n.01	-261.9

Figure 5.4: [best viewed electronically] (**Left**) Synsets and their hypernym relationships from WordNet. (**Middle**) Visualization of our 2-dimensional Gaussian order embedding. (**Right**) The Gaussian "volume" ($\log \det\Sigma$) of the 50-dimensional Gaussian model.

Table 5.2: KL(column||row). Cells in boldface indicate true WordNet hypernym relationships (column $\preceq$ row). Our model predicts a synset pair as a hypernym if the KL less than 1900, where this value is tuned based on the validation set. Most relationship pairs are correctly predicted except for the underlined cells.

	city	location	living_thing	whole	object	physical_entity
city	0	1025	4999	4673	23673	4639
location	**159**	0	4324	4122	26121	5103
living_thing	3623	6798	0	1452	2953	5936
whole	3033	6367	**66**	0	6439	6682
object	138	80	**125**	**77**	0	6618
physical_entity	**232**	**206**	**193**	**166**	**152**	0

5.3.5 Graded Lexical Entailment

HyperLex is a lexical entailment dataset which has fine-grained human annotated scores between concept pairs, capturing varying degrees of entailment [Vulić et al., 2017]. Concept pairs in HyperLex reflect many variants of hypernym relationships, such as `no-rel` (no lexical relationship), `ant` (antonyms), `syn` (synonyms), `cohyp` (sharing a hypernym but not a hypernym of each other), `hyp` (hypernym), `rhyp` (reverse hypernym). We use the noun dataset of HyperLex for evaluation, which contains 2,163 pairs.

We evaluate our model by comparing our model scores against the annotated

scores. Obtaining a high correlation on a fine-grained annotated dataset is a much harder task compared to a binary prediction, since performing well requires meaningful model scores in order to reflect nuances in hypernymy. We use negative divergence as our score for hypernymy scale where large values indicate high degrees of entailment.

We note that the concepts in our trained models are WordNet synsets, where each synset corresponds to a specific meaning of a word. For instance, `pop.n.03` has a definition "a sharp explosive sound as from a gunshot or drawing a cork" whereas `pop.n.04` corresponds to "music of general appeal to teenagers; ...". For a given pair of words (u, v), we use the score of the synset pair (s'_u, s'_v) that has the lowest KL divergence among all the pairs $S_n \times S_v$ where S_u, S_v are sets of synsets for words u and v, respectively. More precisely, $s(u, v) = -\min_{s_u \in S_u, s_v \in S_v} D(s_u, s_v)$. This pair selection corresponds to choosing the synset pair that has the highest degree of entailment. This approach has been used in word embeddings literature to select most related word pairs [Athiwaratkun and Wilson, 2017]. For word pairs that are not in the model, we assign the score equal to the median of all scores. We evaluate our model scores against the human annotated scores using Spearman's rank correlation.

Table 5.3 shows HyperLex results of our models **DOE-A** (asymmetric) and **DOE-S** (symmetric) as well as other competing models. The model **DOE-A** which uses KL divergence and negative sampling approach **S1**, **S2** and **S4** outperforms all other existing models, achieving state-of-the-art performance for the HyperLex noun dataset. (See Section 5.3.2 for hyperparameter details) The model **DOE-S** which uses expected likelihood kernel attains a lower score of 0.455 compared to the asymmetric counterpart (**DOE-A**). This result underscores the importance of

asymmetric measures which can capture relationship directionality.

We provide a brief summary of competing models: **FR** scores are based on concept word frequency ratio [Weeds et al., 2004]. **SLQS** uses entropy-based measure to quantify entailment [Santus et al., 2014]. **Vis-ID** calculates scores based on visual generality measures [Kiela et al., 2015]. **WN-B** calculates the scores based on the shortest path between concepts in WN taxonomy [Miller, 1995]. **w2g** Guassian embeddings trained using the methodology in Vilnis and McCallum [2015]. **VOE** Vector order embeddings [Vendrov et al., 2016]. **Euc** and **Poin** calculate scores based on the Euclidean distance and Poincaré distance of the trained Poincaré embeddings [Nickel and Kiela, 2017]. **HypV** integrates skip-gram to learns hierarchical hypernymy model [Nguyen et al., 2017]. The models **FR** and **SLQS** are based on word occurrences in text corpus, where **FR** is trained on the British National Corpus and **SLQS** is trained on ukWac, WackyPedia [Bailey and Thompson, 2006, Baroni et al., 2009] and annotated BLESS dataset [Baroni and Lenci, 2011]. Other models **Vis-ID, w2g, VOE, Euc, Poin** and ours are trained on WordNet, with the exception that **Vis-ID** also uses Google image search results for visual data. The reported results of **FR**, **SLQS**, **Vis-ID**, **WN-B**, **w2g** and **VOE** are from Vulić et al. [2017].

We note that an implementation of Gaussian embeddings model (**w2g**) reported by Vulić et al. [2017] does not perform well compared to previous benchmarks such as **Vis-ID, FR, SLQS**. Our training approach yields the opposite results and outperforms other highly competitive methods such as Poincaré embeddings and Hypervec. This result highlights the importance of the training approach, even if the concept representation of our work and Vilnis and McCallum [2015] both use Gaussian distributions. In addition, we observe that vector order embeddings

Table 5.3: Spearman's correlation for HYPERLEX nouns.

	FR	**SLQS**	**Vis-ID**	**WN-B**	**w2g**	**VOE**	**Poin**	**HypV**	**DOE-S**	**DOE-A**
ρ	0.283	0.229	0.253	0.240	0.192	0.195	0.512	0.540	0.455	**0.590**

(VOE) do not perform well compared to our model, which we hypothesize is due to the "soft" orders induced by the divergence penalty that allows our model scores to more closely reflect hypernymy degrees.

We note another interesting observation that a model trained on a symmetric divergence (ELK) from Section 5.3.3 can also achieve a high HYPERLEX correlation of 0.532 if KL is used to calculate the model scores. This is because the encapsulation behavior can arise even though the training penalty is symmetric (more explanation in Section 5.3.3). However, using the symmetric divergence based on ELK results in poor performance on HYPERLEX (0.455), which is expected since it cannot capture the directionality of hypernymy.

We note that another model LEAR obtains an impressive score of 0.686 [Vulić and Mrkšić, 2014]. However, LEAR use pre-trained word embeddings such as WORD2VEC or GLOVE as a pre-processing step, leveraging a large vocabulary with rich semantic information. To the best of our knowledge, our model achieves the highest HYPERLEX Spearman's correlation among models without using large-scale pre-trained embeddings.

Table 5.4 shows the effects of negative sample selection described in Section 5.2.5. We note again that **S1** is the technique used in literature [Socher et al., 2013, Vendrov et al., 2016] and **S2**, **S3**, **S4** are the new techniques we proposed. The notation, for instance, $k \times \mathbf{S1} + \mathbf{S2}$ corresponds to using k samples from **S1** and 1 sample from **S2** per each positive sample. We observe that our new selection

Table 5.4: Spearman's correlation for HyperLex nouns for different negative sample schemes.

Negative Samples	ρ	Negative Samples	ρ
1×**S1**	0.527	1×**S1** + **S2** + **S4**	**0.590**
2×**S1**	0.529	2×**S1** + **S2** + **S4**	0.580
5×**S1**	0.518	5×**S1** + **S2** + **S4**	0.582
10×**S1**	0.517	1×**S1** + **S2** + **S3**	0.570
1×**S1** + **S2**	0.567	2×**S1** + **S2** + **S3**	0.581
2×**S1** + **S2**	0.567	**S1** + 0.1×**S2** +0.9×**S3**	0.564
3×**S1** + **S2**	0.584	**S1** + 0.3×**S2** +0.7×**S3**	0.574
5×**S1** + **S2**	0.561	**S1** + 0.7×**S2** +0.3×**S3**	0.555
10×**S1** + **S2**	0.550	**S1** + 0.9×**S2** +0.1×**S3**	0.533

methods offer strong improvement from the range of $0.51 - 0.52$ (using **S1** alone) to 0.55 or above for most combinations with our new selection schemes.

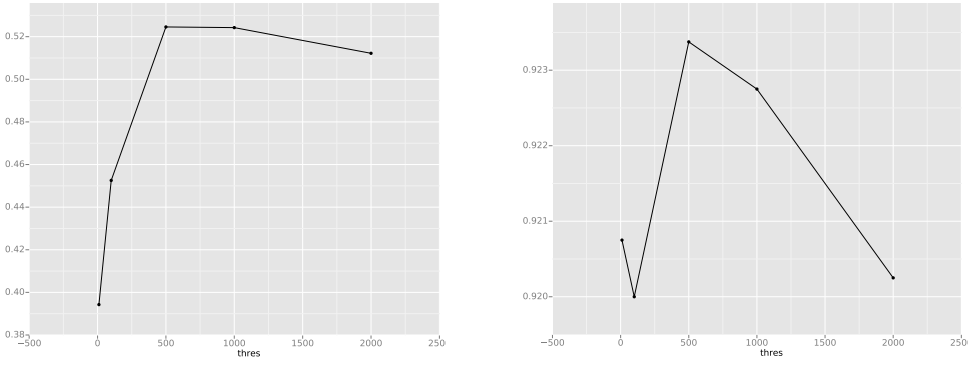
5.3.6 Ablation Study and Analysis

We emphasize that Gaussian embeddings have been used in the literature, both in the unsupervised settings where word embeddings are trained with local contexts from text corpus, and in supervised settings where concept embeddings are trained to model annotated data such as WordNet. The results in supervised settings such as modeling WordNet have been reported to compare with competing models but often have inferior performance [Vendrov et al., 2016, Vulić et al., 2017]. This chapter reaches the opposite conclusion, showing that a different training approach using Gaussian representations can achieve state-of-the-art results.

Divergence Threshold

Consider a relationship $f \preceq g$ where f is a hyponym of g or g is a hypernym of f. Even though the divergence $D(f||g)$ can capture the extent of encapsulation, a density f will have the lowest divergence with respect with g only if $f = g$. In addition, if f is a more concentrated distribution that is encompassed by g, $D(f||g)$ is minimized when f is at the center of g. However, if there any many hyponyms f_1, f_2 of g, the hyponyms can compete to be close to the center, resulting in too much overlapping between f_1 and f_2 if the random sampling to penalize negative pairs is not sufficiently strong. The divergence threshold γ is used such that there is no longer a penalty once the divergence is below a certain level.

We demonstrate empirically that the threshold γ is important for learning meaningful Gaussian distributions. We fix the hyperparameters $m = 2000$ and $\beta = 5 \times 10^{-5}$, with **S1** negative sampling. Figure 5.5 shows that there is an optimal non-zero threshold and yields the best performance for both WordNet Hypernym prediction and HyperLex Spearman's correlation. We observe that using $\gamma = 0$ is detrimental to the performance, especially on HyperLex results.

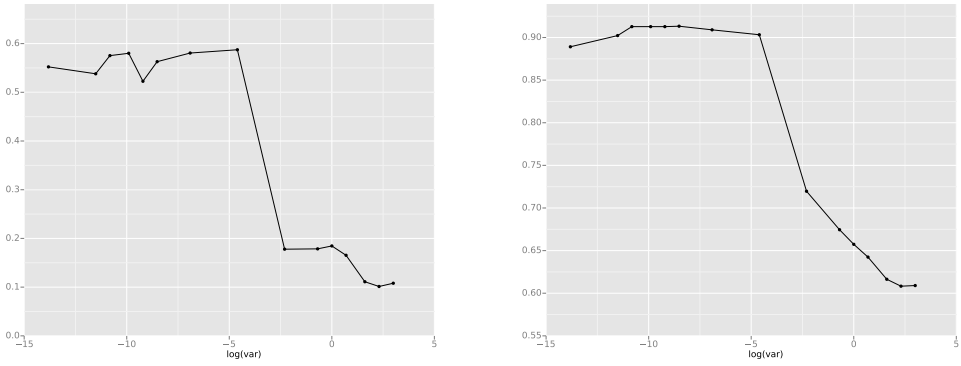


(a) (b)

Figure 5.5: (a) Spearman's correlation on HyperLex versus γ (b) Test Prediction Accuracy versus γ.

Initial Variance Scale

As opposed to the mean vectors that are randomly initialized, we initialize all diagonal covariance elements to be the same. Even though the variance can adapt during training, we find that different initial scales of variance result in drastically different performance. To demonstrate, in Figure 5.6, we show the best test accuracy and HyperLex Spearman's correlation for each initial variance scale, with other hyperparameters (margin m and threshold γ) tuned for each variance. We use **S1** + **S2** + **S4** as a negative sampling method. In general, a low variance scale β increases the scale of the loss and requires higher margin m and threshold γ. We observe that the best prediction accuracy is obtained when $\log(\beta) \approx -10$ or $\beta = 5 \times 10^{-5}$. The best HyperLex results are obtained when the scales of β are sufficiently low. The intuition is that low β increases the scale of divergence $D(\cdot||\cdot)$, which increases the ability to capture relationship nuances.



(a) (b)

Figure 5.6: (a) Spearman's correlation on HyperLex versus $\log(\beta)$ (b) Test Prediction Accuracy versus $\log(\beta)$.

Loss Function

We verify that for this task, our loss function in Equation 5.7 in superior to Equation 2.3 originally proposed by Vilnis and McCallum [2015]. We use the exact same setup with new negative sample selections and KL divergence thresholding and compare the two loss functions. Table 5.5 verifies our claim.

Table 5.5: Best results for each loss function for two negative sampling setups: **S1** (**Left**) and **S1** + **S2** + **S4** (**Right**)

Eq.	Test Accuracy	HyperLex
5.7	0.923	0.527
2.3	0.886	0.524

Eq.	Test Accuracy	HyperLex
5.7	0.911	0.590
2.3	0.796	0.489

Dimensionality

Table 5.6 shows the results for many dimensionalities for two negative sample strategies: **S1** and **S1** + **S2** + **S4** .

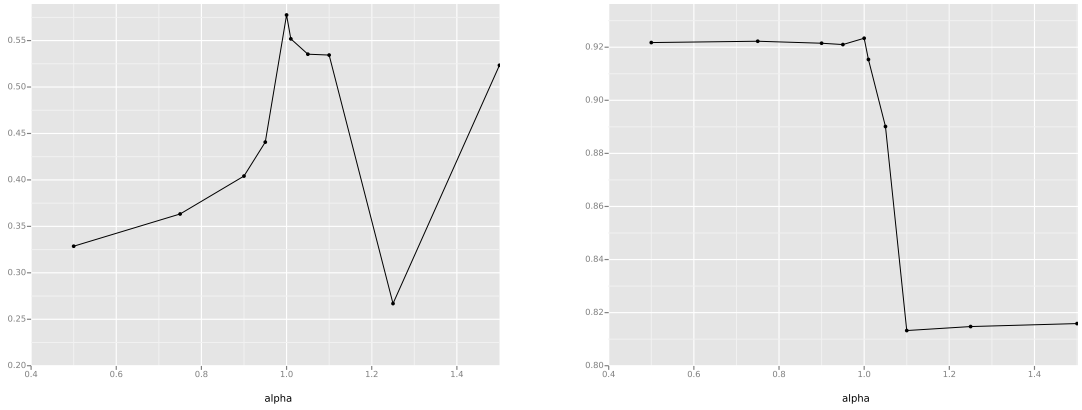
Table 5.6: Best results for each dimension with negative samples **S1** (**Left**) and **S1** + **S2** + **S4** (**Right**)

d	Test Accuracy	HyperLex
5	0.909	0.437
10	0.919	0.462
20	0.922	0.487
50	0.923	0.527
100	0.924	0.526
200	0.918	0.526

d	Test Accuracy	HyperLex
5	0.901	0.483
10	0.909	0.526
20	0.914	0.545
50	0.911	0.590
100	0.913	0.573
200	0.910	0.568

α-divergences

Table 5.7 show the results using models trained and evaluated with $D(\cdot||\cdot) = D_\alpha(\cdot||\cdot)$ with negative sampling approach **S1**. Interestingly, we found that $\alpha \to 1$ (KL) offers the best result for both prediction accuracy and HyperLex. It is possible that $\alpha = 1$ is sufficiently asymmetric enough to distinguish hypernym directionality, but does not have as sharp penalty as in $\alpha > 1$, which can help learning.



(a) (b)

Figure 5.7: (a) Spearman's correlation on HyperLex versus α (b) Test Prediction Accuracy versus α.

5.4 Caption Image Retrieval

We use the task of caption-image retrieval to evaluate the effectiveness of our density order embedding model. The task entails ranking the matching captions given a query image (Caption Retrieval) and ranking the matching images given a query caption (Image Retrieval). The goal is to learn the score function $S(c, i)$ for caption c and image i to be used for ranking. Traditionally these scores involve vector features of caption c and image i. We use the density order embedding

model with the following similarity score $S(c, i) = -E(f_i(i), f_c(c))$ where E is a penalty function described in Section 5.2.3. For the asymmetric measure E (Section 5.2.3), this affinity function implies that we view captions to be above images in the visual-semantic hierarchy. We use the KL divergence in our experiments.

5.4.1 Related Work

Multimodal learning has received much attention over the past decade. We compare our results with the following models: **MNLM** [Kiros et al., 2014], **m-RNN** [Mao et al., 2014], **DVSA** [Karpathy and Li, 2015], **STV** [Kiros et al., 2015a], **FV** [Klein et al., 2015], **m-CNN** and **m-CNN$_{\mathbf{ENS}}$** [Ma et al., 2015], **VQA** [Lin and Parikh, 2016], **Vector Order-Embedding (VOE)** [Vendrov et al., 2016], Representation-level Fusion **(RLF) VQA-aware**, **RLF VQA-agnostic** [Lin and Parikh, 2016]. All prior works learn vector embeddings unlike our approach which uses probability distributions. Most works except for VOE map images and captions to a symmetric visual-semantic space, often with cosine similarity. We note that the RLF VQA-aware uses an external corpus (Visual Question Answering) to help with training and therefore could not compare directly to our results. However, we include the number for completeness.

5.4.2 Image and Caption Density Model

For the image density function $f(x|i)$, we use a linear layer (weight W_μ^{image} and bias b_μ^{image}) after a pre-trained CNN to compute mean vector $\mu(i)$ and another linear layer (weight W_Σ^{image} and bias b_Σ^{image}) to compute $\log \text{diag}(\Sigma)(i)$ given an image i.

That is,

$$f(x|i) = \mathcal{N}\left[x; \mu = W_{\mu}^{\text{image}} \cdot \text{CNN}(i) + b_{\mu}^{\text{image}}, \Sigma = \text{diag}(e^{W_{\Sigma}^{\text{image}} \cdot \text{CNN}(i) + b_{\Sigma}^{\text{image}}})\right] \tag{5.8}$$

Our feature extractor, $\text{CNN}(i)$ is a 19-layer VGG pretrained on ImageNet [Simonyan and Zisserman, 2014]. We obtain the image features by feeding 10 of 224×224 crops from corners, center, and the horizontal reflection to the VGG network and average the 4096-dimensional $fc7$ features. We fix the weights of VGG during training. This setup is similar to that of Klein et al. [2015] and Vendrov et al. [2016]. Similarly, we use linear layers after Gated Recurrent Unit (GRU) features [Cho et al., 2014] which is trained jointly with other weights to compute the sentence's μ and $\log \text{diag}(\Sigma)$.

$$g(x|i) = \mathcal{N}\left[x; \mu = W_{\mu}^{\text{caption}} \cdot \text{GRU}(i) + b_{\mu}^{\text{caption}}, \Sigma = \text{diag}(e^{W_{\Sigma}^{\text{caption}} \cdot \text{GRU}(i) + b_{\Sigma}^{\text{caption}}})\right] \tag{5.9}$$

5.4.3 Loss Function

We adopt a pairwise ranking loss used by Karpathy and Li [2015], Kiros et al. [2014], Socher et al. [2014] with our proposed score function. The first part of the loss function in Equation 5.10 aims to maximize the score $S(c, i)$ between a true caption c and the corresponding image i to be above $S(c, i')$, the score between caption c and a negative image i'. The second part is similar but aims to push the score $S(c, i)$ higher than the score $S(c', i)$ for a negative caption c':

$$\sum_{(c,i)} \left(\sum_{c'} \max\{0, \alpha - S(c, i) + S(c', i)\} + \sum_{i'} \max\{0, \alpha - S(c, i) + S(c, i')\} \right) \tag{5.10}$$

5.4.4 Training Details

We use Microsoft COCO dataset for training and evaluation [Lin et al., 2014] which consists of up to five text captions per image. We use the data setup similar to Karpathy and Li [2015] where the train, validation, and test splits are 113287, 5000 and 5000 images. We use a minibatch size of 128 random image-caption pairs. To obtain a negative pair of image-caption for each sample, we use the other 127 contrastive images or texts. We train using the Adam optimizer for 30 to 50 epochs and use the model with the best validation score for test set evaluation. The dimension of our Gaussian distributions is 1024. We use a word embedding size of 300 for GRU and the GRU hidden state size of 1024. We also constrain the mean vector μ to have a unit norm. We perform a grid search for optimal hyperparameters. The hyperparameters for the best model (DOE) are $\alpha = 10.0$ and $\gamma = 0.05$, $\alpha = 10.0$ and $\gamma = 0.0$ for the DOE-reverse model, and $\alpha = 10.0$ and $\gamma = 100$ for the DOE-elk model.

5.4.5 Results

Table 5.7 shows the results where our model performs competitively with state-of-the-art benchmarks. Similar to the hypernym task, our asymmetric measure is more suitable for caption-image retrieval task, providing better performance than the symmetric model. This observation agrees with the vector order embedding counterpart where the asymmetric model performs slightly better than the symmetric one. We also experiment with the reversed order, with images above text in the hierarchy and observe slightly inferior performance. We include a re-implementation of the vector order embeddings (VOE-reverse) where also reverse the order which

Table 5.7: Caption-image retrieval results on Microsoft COCO. The figures reported are the mean metrics on five-fold splits, where each has 1k test images. For each metric, the best performing model is in boldface. Our density order embedding model with text above image in the visual-semantic hierarchy performs near state-of-the-art. We note that the model RLF VQA-aware† has an extra advantage since it uses an external dataset from visual question answering. Our mean ranking performs the best out of all models shown for caption retrieval task. The numbers from VOE‡ and VOE-symmetric‡ is taken from [Vendrov et al., 2016].

Model	Caption Retrieval				Image Retrieval			
	R@1	R@10	Med r	Mean r	R@1	R@10	Med r	Mean r
MNLM	43.4	85.8	2.0	*	31.0	79.9	3	*
m-RNN	41.0	83.5	2.0	*	29.0	77.0	3	*
DVSA	38.4	80.5	**1.0**	*	27.4	74.8	3	*
STV	33.8	82.1	3.0	*	25.9	74.6	4	*
FV	39.4	80.9	2.0	10.4	25.1	76.6	4	11.1
m-CNN	38.3	81.0	2.0	*	27.4	79.5	3	*
m-CNN$_{\text{ENS}}$	42.8	84.1	2.0	*	32.6	82.8	3	*
RLF VQA-agnostic	45.8	86.1	*	*	33.6	81.0	*	*
RLF VQA-aware†	**50.5**	**89.7**	*	*	37.0	82.9	*	*
VOE‡	46.7	88.9	2.0	5.7	**37.9**	**85.9**	**2.0**	**8.1**
VOE-reverse	42.7	87.8	2.0	6.2	34.3	85.0	2.6	7.8
VOE-symmetric‡	45.4	88.7	2.0	5.8	36.3	85.8	2.0	9.0
DOE	46.7	88.8	2.0	**5.6**	36.8	84.5	2.2	8.4
DOE-reverse	43.9	87.8	2.0	5.8	36.0	84.8	2.2	8.0
DOE-symmetric	41.3	86.7	2.0	7.0	32.9	82.9	2.2	8.4

shows a similar trend.

5.5 Discussion

Analogous to recent work by Vulić and Mrkšić [2014] which post-processed word embeddings such as GloVe or word2vec, our future work includes using the WordNet hierarchy to impose encapsulation orders when training probabilistic embeddings.

In the future, the distribution approach could also be developed for encoder-decoder based models for tasks such as caption generation where the encoder

represents the data as a distribution, containing semantic and visual features with uncertainty, and passes this distribution to the decoder which maps to text or images. Such approaches would be reminiscent of variational autoencoders [Kingma and Welling, 2013], which take *samples* from the encoder's distribution.

CHAPTER 6

CONCLUSION

The goal of this thesis is to demonstrate the effectiveness of probability representations for many types of applications such as words and hierarchical data including taxonomy graphs, sentences and images, . We lay out concrete steps for future work below.

Disambiguation of Multi-Lingual Embeddings Recent model MUSE by [Conneau et al., 2018] successfully constructs bi-lingual embeddings based on independently trained embeddings of two languages. These two embeddings are aligned through an adversarial loss to match the distributions of vector points. However, there can be misalignment due to polysemies. For instance, an Italian word *porta* can either mean "bring" or "door" but there is no polysemy in English that captures the two meanings. For embeddings that do not allow for multimodality such as WORD2VEC or FASTTEXT, which MUSE adopts, the trained embeddings of *porta* is likely to have nearest neighbors that correspond to one predominant meaning (this phenomena are seen in Table 3.1 and 4.1). The alignment learning can be overly constrained and some of the meanings misrepresented.

Disentangled representations such as our PFT model can help disambiguate. Instead of matching vector points, we can match the entire Gaussian mixture distributions and allow of meaning matching through on individual Gaussian components. Extensions of MUSE such as multilingual embeddings [Alaux et al., 2019, Ammar et al., 2016, Chen and Cardie, 2018] or multilingual sentence embeddings [Artetxe and Schwenk, 2018] can also benefit from such disambiguation.

Downstream Tasks We foresee the use of our embeddings for many important downstream tasks such as machine translation [Artetxe et al., 2017, Bahdanau et al., 2014, Lample et al., 2017, 2018, Sutskever et al., 2014]. For instance, in the training pipeline of a current state-of-the-art unsupervised machine translation model by Lample et al. [2017], the system pre-trains FastText embeddings on mono-lingual corpora of two languages and learn bi-lingual embeddings with MUSE [Conneau et al., 2018]. The bi-lingual embeddings can be further improved by the method outlined above where we disambiguate bi-lingual polysemies through our model PFT. We can also extend it to multi-lingual embeddings in the case of multi-lingual translations. In addition, there are advantages to having multimodality in embeddings which are demonstrated in Chapter 4 through superior scores on word similarity evaluation. These two components, (1) disambiguation on word alignment and (2) superior quality through disentangled multimodal representations, can be key to improve challenging tasks such as neural machine translation further.

BIBLIOGRAPHY

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *arXiv preprint arXiv:1610.08936*, 2016.

Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. Unsupervised hyper-alignment for multilingual word embeddings. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJe62s09tX.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. Massively multilingual word embeddings. *CoRR*, 2016.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *CoRR*, abs/1601.03764, 2016. URL http://arxiv.org/abs/1601.03764.

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *CoRR*, 2018.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised

neural machine translation. *CoRR*, 2017.

Ben Athiwaratkun and Andrew Gordon Wilson. Multimodal word distributions. In *ACL*, 2017.

Ben Athiwaratkun and Andrew Gordon Wilson. On modeling hierarchical data via probabilistic order embeddings. *ICLR*, 2018.

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. *CoRR*, 2018.

Benjamin R Baer, Skyler Seto, and Martin T Wells. Exponential family word embeddings: An iterative approach for learning word vectors. *NIPS Workshop: IRASL*, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, 2014.

Steve Bailey and Dave Thompson. UKWAC: building the uk's first public web archive. *D-Lib Magazine*, 12(1), 2006.

Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, 2011.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3), 2009.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *EACL*, pages 23–32, 2012.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard,

Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, 2018.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016. URL http://arxiv.org/abs/1607.04606.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1), January 2014. ISSN 1076-9757.

Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

Xilun Chen and Claire Cardie. Unsupervised multilingual word embeddings. *EMNLP*, 2018.

Xinchi Chen, Xipeng Qiu, Jingxiang Jiang, and Xuanjing Huang. Gaussian mixture embeddings for multiple word prototypes. *CoRR*, 2015.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *EMNLP*, 2014.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations

using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NIPS*, 2018.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167, 2008.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *ICLR*, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016. URL http://arxiv.org/abs/1606.09375.

Bhuwan Dhingra, Christopher J. Shallue, Mohammad Norouzi, Andrew M. Dai, and George E. Dahl. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing, TextGraphs@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 6, 2018*, 2018.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1), 2002.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *CoRR*, abs/1704.02798, 2017a.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017b.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788. URL http://dl.acm.org/citation.cfm?id=177910.177914.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034*, 2016.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. In *ACL*, 2017.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *ICML*, 2018a.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *NIPS*, 2018b.

Çaglar Gülçehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. *ICLR*, 2019.

Iryna Gurevych. Using the structure of a conceptual network in computing semantic relatedness. In *Natural Language Processing - IJCNLP 2005, Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005, Proceedings*, pages 767–778, 2005.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, 2012.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale

learning of word relatedness with constraints. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1406–1414, 2012.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015. URL http://arxiv.org/abs/1506.05163.

Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456, 2014.

Julia Hockenmaier and Alice Lai. Learning to predict denotational probabilities for modeling entailment. In *EACL*, 2017.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, 2012.

Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *JMLR*, 2004.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142, 2002.

Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. Exploiting image generality for lexical entailment detection. In *ACL*, 2015.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. Learning to generate

word representations using subword information. In *COLING*, 2018.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NIPS*, 2015a.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NIPS*, 2015b.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. Associating neural word embeddings with deep image representations using fisher vectors. In *CVPR*, 2015.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

Connie Kou, Hwee Kuan Lee, and Teck Khim Ng. Distribution regression network, 2018a. URL https://openreview.net/forum?id=ByYPLJA6W.

Connie Kou, Hwee Kuan Lee, and Teck Khim Ng. Distribution regression network.

CoRR, 2018b.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. Charner: Character-level named entity recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 911–921, 2016. URL http://aclweb.org/anthology/C/C16/C16-1087.pdf.

Alice Lai and Julia Hockenmaier. Learning to predict denotational probabilities for modeling entailment. In *EACL*, 2017.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/D18-1549.

Andrew J Landgraf and Jeremy Bellay. word2vec skip-gram with negative sampling is a weighted logistic pca. *arXiv preprint arXiv:1705.09755*, 2017.

Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *TACL*, 5:365–378, 2017. URL https://transacl.org/ojs/index.php/tacl/article/view/1051.

Ira Leviant and Roi Reichart. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR*, abs/1508.00106, 2015. URL http://arxiv.org/abs/1508.00106.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

Xiang Li, Luke Vilnis, and Andrew McCallum. Improved representation learning for predicting commonsense ontologies. *NIPS Workshop*, 2017.

Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. In *ICLR*, 2019.

Yingzhen Li and Richard E. Turner. Rényi divergence variational inference. In *NIPS*, 2016.

Friedrich Liese and Igor Vajda. *Convex Statistical Distances*. Leipzig : Teubner, 1987.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Xiao Lin and Devi Parikh. Leveraging visual question answering for image-caption ranking. In *ECCV*, 2016.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2418–2424, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9314.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *ACL workshop*, 2002.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, 2013.

Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. Multimodal convolutional neural networks for matching image and sentence. In *ICCV*, 2015.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. Deep captioning

with multimodal recurrent neural networks (m-rnn). *CoRR*, abs/1412.6632, 2014.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 216–223, 2014.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukás Burget, and Jan Cernocký. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011*, pages 196–201, 2011a.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531, 2011b.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013b.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In

NIPS, 2013c.

George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38 (11), November 1995.

George A. Miller and Walter G. Charles. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes*, 6(1):1–28, 1991. ISSN 01690965. doi: 10.1080/01690969108406936.

Andriy Mnih and Geoffrey E. Hinton. A scalable hierarchical distributed language model. In *NIPS*, 2008.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, 2005.

Eric T. Nalisnick and Sachin Ravi. Infinite dimensional word embeddings. *CoRR*, abs/1511.05392, 2015.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1059–1069, 2014. URL http://aclweb.org/anthology/D/D14/D14-1113.pdf.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 233–243, 2017.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical

representations. *NIPS*, 2017.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*, 2018.

Seong Joon Oh, Andrew C. Gallagher, Kevin P. Murphy, Florian Schroff, Jiyan Pan, and Joseph Roth. Modeling uncertainty with hedged instance embeddings. In *ICLR*, 2019.

Leandro Pardo. *Statistical Inference Based on Divergence Measures*, chapter 1, pages 1–54. Chapman & Hall/CRC, 2006.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1532–1543, 2014.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, 2011. ISBN 978-1-4503-0632-4.

Alfred Renyi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, Berkeley, Calif., 1961.

Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10), October 1965.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, 2008.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. Chasing hypernyms in vector spaces with entropy. In *EACL*, 2014.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation

of rare words with subword units. In *ACL*, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR*, abs/1702.03859, 2017. URL http://arxiv.org/abs/1702.03859.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, 2013.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2, 2014.

C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:88–103, 1904.

Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975, 2016. URL http://arxiv.org/abs/1612.03975.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. A probabilistic model for learning multi-prototype word embeddings. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 151–160, 2014.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings

of images and language. *ICLR*, 2016.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *ICLR*, 2015.

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *ACL*, 2018.

Ivan Vulić and Nikola Mrkšić. Specialising word vectors for lexical entailment. *CoRR*, abs/1710.06371, 2014.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 2017.

Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *CVPR*, 2018.

Julie Weeds, David J. Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *COLING*, 2004.

Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016b.

Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. 2019.

Dongqiang Yang and David M. W. Powers. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*, 2006.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor

Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Deep reinforcement learning with relational inductive biases. In *ICLR*, 2019.

Shenjian Zhao and Zhihua Zhang. An efficient character-level neural machine translation. *CoRR*, abs/1608.04738, 2016. URL http://arxiv.org/abs/1608.04738.

G.K. Zipf. *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press, 1949. URL https://books.google.com/books?id=1tx9AAAAIAAJ.