Exponential Family Embeddings

Maja Rudolph

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Word embeddings are a powerful approach for capturing semantic similarity among terms in a vocabulary. Exponential family embeddings extend the idea of word embeddings to other types of high-dimensional data. Exponential family embeddings have three ingredients; embeddings as latent variables, a predefined conditioning set for each observation called the context, and a conditional likelihood from the exponential family. The embeddings are inferred with a scalable algorithm. This thesis highlights three advantages of the exponential family embeddings model class: (A) The approximations used for existing methods such as word2vec can be understood as a biased stochastic gradients procedure on a specific type of exponential family embedding model — the Bernoulli embedding. (B) By choosing different likelihoods from the exponential family we can generalize the task of learning distributed representations to different application domains. For example, we can learn embeddings of grocery items from shopping data, embeddings of movies from click data, or embeddings of neurons from recordings of zebrafish brains. On all three applications, we find exponential family embedding models to be more effective than other types of dimensionality reduction. They better reconstruct held-out data and find interesting qualitative structure. (C) Finally, the probabilistic modeling perspective allows us to incorporate structure and domain knowledge in the embedding space. We develop models for studying how language varies over time, differs between related groups of data, and how word usage differs between languages. Key to the success of these method is that the embeddings share statistical information through hierarchical priors or neural networks. We demonstrate the benefits of this approach in empirical studies of Senate speeches, scientific abstracts, and shopping baskets.

Contents

List of Figures

List of Tables

Acknowledgements

Thank you, Dave, for being the best advisor any graduate student could hope for.

Thank you, Fran, for the fruitful collaboration.

Thank you, Allison, Jaan, James, and Stephan, for introducing me to the Blei Lab.

Thank you for being great labmates.

Thank you, Adji, Alp, Dustin, Rajesh, Yixin, and Scott, for being inspiring.

Thank you, Wes, for our daily coffee runs.

Thank you, Victor, for great conversations.

Thank you, Jackson, for breaking into song sometimes.

Thank you, Laurent, for the birthday cake at WWW.

Thank you, Stella, for caring for the Blei Lab.

Thank you, Emti, for hosting me at RIKEN, Tokyo.

Thank you, Shih-Fu, for giving me a chance.

Thank you, Renee, for that life-changing joke.

Thank you, Emily, for being an awesome travel companion.

Thank you, Sky, for the moral support.

Thank you, MIT, for the chance to face the firehose.

Thank you, YTTP, for the room to breathe.

Thank you, 112th street, for feeling like home.

Thank you, New York City, for not driving me crazy.

Thank you, Larisa, Rita, Alona, and Jan for being my backbone.

Thank you, Mom, for the challenge.

Dedicated to Family

Introduction

Word embeddings have become a popular tool for analyzing language. They ingest a large amount of text and produce a low dimensional distributed representation for each word. This is important because text is inherently high dimensional and discrete. In the discrete representation, all words are orthogonal to each other. In contrast, embeddings allow us to compute nuanced distances between all words. The name of the game is to design embedding models such that the distances in the learned embedding space reflect properties of the words we care about such as semantic or syntactic similarities.

Many variants of word embeddings (Bengio et al., 2003; Mikolov et al., 2013a; Mnih and Kavukcuoglu, 2013; Pennington, Socher, and Manning, 2014) use the co-occurrence patterns of the words in text as the main training signal. This is justified by the intuition that a word's meaning is defined by the company it keeps (Harris, 1954). Many existing methods for learning word representations can be seen as log-bilinear models that predict each word from the context of surrounding words.

The embeddings can then be used as input features for downstream NLP tasks, or as the last layer in a word prediction task (Collobert et al., 2011; Weston et al., 2012). They can be used for document retrieval and classification (Taddy, 2015) and as a tool for computational social science (Hamilton, Leskovec, and Jurafsky, 2016a). If word embeddings are so useful

for text, are there other application areas that could benefit from these ideas?

This thesis is about exponential family embeddings (EF-EMBS), a model class that uses ideas from generalized linear models (GLMS) (McCullagh and Nelder, 1989) and exponential family distributions (Harris, 1954) to unlock the power of embeddings in applications beyond language. EF-EMBS generalize continuous bag of words (CBOW) (Mikolov et al., 2013b) in the same way that exponential family principal component analysis (PCA) (Collins, Dasgupta, and Schapire, 2001) generalizes PCA, GLMS (McCullagh and Nelder, 1989) generalize regression, and deep exponential families (Ranganath et al., 2015) generalize sigmoid belief networks (Neal, 1990). This thesis highlights three aspects of the EF-EMB model class.

Highlight 1: Existing Word Embedding Methods as Special Case. Many existing word embedding methods maintain probabilities over words, which are expensive to compute, because they require normalizing over the entire vocabulary (Bengio et al., 2003). For this reason, different approximation methods have been developed to scale embedding methods to large vocabulary sizes, including methods using importance sampling (Bengio, Senécal, et al., 2003; Bengio and Senécal, 2008), tree-based methods like the hierarchical softmax (Morin and Bengio, 2005; Mnih and Hinton, 2009; Mikolov et al., 2011; Mikolov et al., 2013a), and methods based on noise contrastive estimation (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012; Mikolov et al., 2013a; Mnih and Kavukcuoglu, 2013). One of the most popular techniques for learning word embeddings at scale is negative sampling (Mikolov et al., 2013a; Mikolov et al., 2013b). In Chapter 1, we show that it can be understood as a biased stochastic gradients procedure on a specific type of model from the EF-EMB model class — the Bernoulli embedding.

Highlight 2: Generalization to Other Application Domains. A second aspect highlighted in this thesis is that EF-EMBS allow us to generalize the task of learning embeddings to other application domains. All EF-EMB models have the embeddings as latent variables, but depending on the application, different modeling choices can be made. The first choice is the *context*, a function that specifies the conditioning set for each observation and the second choice is a conditional distribution from the exponential family which best fits the data type. For count data from a shopping application for example, we use Poisson distributions. The resulting Poisson embedding model allows us to learn embeddings of grocery items based on shopping data. More examples will follow in Chapter 2. We fit all models in the EF-EMB class using stochastic gradients on the pseudo-likelihood (Arnold, Castillo, Sarabia, et al., 2001). Exponential families ensure we have to derive the gradients only once and have properties that simplify the gradients. EF-EMBS are presented in Chapter 2.

Highlight 3: Latent Structure in the Embedding Space. A third highlight is that treating the embeddings as latent variables in a probabilistic model allows us to impose domain knowledge and structure on the embeddings. For example, we study how to fit embeddings that vary over time or across related groups of data. Rather than fitting a separate embedding for each time point or group and then trying to compare the results to study variation in word meaning, we train a single model. We desing the structured embedding models in a way such that all their embeddings are in one shared space and such that priors (or neural networks) help manage how information is shared between the embeddings. Several structured extensions of EF-EMBS are presented in Chapter 3.

This thesis is structured around these three highlights. After introducing relevant background in the next chapter, we first present Bernoulli embeddings (B-EMBS) in Chapter 1, as a specific example of a model from the EF-EMB model class. The B-EMB is a model of text, and introduces the main ideas of existing word embedding methods while setting the stage for the more general EF-EMBS presented in the following chapter (Chapter 2). In Chapter 3, we study multiple extensions of EF-EMB, including dynamic embeddings for language that changes over time, hierarchical and amortized embeddings for grouped data, deep embeddings, and a hierarchical embedding model for multilingual texts.

Background

Probabilistic modeling. Probabilistic modeling is an approach for uncovering patterns in data. Today, there is an abundance of data out there, and probabilistic modeling let's us formalize our beliefs of the underlying structure in the data, including our uncertainty about the structure and the noise inherent in the data. We proceed as follows; we identify a collection of random variables, both observable and not, and posit how they relate to each other in terms of their probabilities. Without evidence, our model should reflect that we are uncertain about many aspects of the model, but as evidence is collected or observed, we can update our beliefs. This is done using Bayes' rule, which lets us infer the unknown quantities of our model from data. With Bayes' rule we can uncover patterns of interest in the data and we can use it to make predictions about new data.

Let θ be the set of latent variables. Those are the variables we will not be able to observe and that we wish to make inferences about. The observed variables are X, and our observations (the data) are $X = x$. We formalize our prior beliefs about θ in form of a prior distribution $p(\theta)$, and the relationship between the observed variables X and the latents θ is communicated in terms of a likelihood $p(X|\theta)$. Inferences about θ are made using Bayes'

rule (which follows from the chain rule of probability):

$$p(\theta|X = x) \propto p(X = x|\theta)p(\theta) \tag{1}$$

There are many probability distributions we can place on the prior and the likelihood and it is common to use distributions from the exponential family.

Exponential family. The exponential family (Brown, 1986) is a class of probability distributions that are widely used in applied statistics and machine learning. Prominent members include the Gaussian distribution, Bernoulli and Poisson.

All exponential family distributions can be written in a standardized form,

$$p(X = x; \theta) = h(x) \exp\{\eta(\theta)^\top T(x) - A(\theta)\}. \tag{2}$$

The function $h(x)$ is called the base measure and $T(x)$ is a vector of sufficient statistics. The sufficient statistics summarize all relevant information of a dataset to determine its pdf. In other words, even if two dataset are different, but they produce the same sufficient statistics, then they have the same probability under the exponential family distribution with those sufficient statistics.

Exponential family distributions have been extensively studied for almost hundred years because of their nice properties. For one, they arise naturally as the solution to constrained entropy maximization problems. For specific sufficient statistics (those are the constraints in the aforementioned maximization problem), the exponential family distribution with those sufficient statistics has maximal entropy. Another useful property is that the log-partition

function $A(\theta)$ which ensures Equation (2.1) is normalized, is differentiable and convex. In addition, the derivatives of the log-partition function are the expectations of the sufficient statistics,

$$\nabla_\theta A(\theta) = \mathbb{E}[T(X)]. \tag{3}$$

Finally, the product of two exponential family distributions is also in the exponential family, if the resulting product distribution can be normalized. This makes exponential family distributions useful for Bayesian computation. According to Bayes' rule (Equation (1)) the posterior is proportional to the product of the prior and the likelihood term. When both distributions are *conjugate* exponential family distributions, the posterior can be deduced by reading off the parameters of the resulting distribution, which saves the expensive normalization step.

Generalized linear models. Generalized linear models (McCullagh and Nelder, 1989) are a class of models that use exponential family distributions (Brown, 1986) to generalize linear regression to non Gaussian data. They are a model of data that has covariates X, response Y, and parameters θ. The likelihood $p(Y \mid X, \theta)$ of a GLM is an exponential family distribution with natural parameter η. The natural parameter of the exponential family distribution is a linear combination of the covariates X and the model parameters θ, $\eta = \theta^\top X$. The inverse link function maps the natural parameter to the mean response, and so even though η is a linear function of the data X, GLMs allow us to capture nonlinear relationships between covariates and response. One common example is logistic regression, which uses a Bernoulli likelihood, whose inverse link is the logistic function. Linear regression can be recovered by using a Gaussian model. Its link function is the identity, hence linear regression

has a linear relationship between covariates and response.

Given data $\{(x_i, y_i)\}_{i=1}^N$ and a GLM, we can compute the maximum likelihood estimator of θ by maximizing the log likelihood of the data. Due to the form of the exponential family (Equation (2.1)) the log likelihood of a GLM has the form

$$\sum_{i=1}^{N} \log p(Y = y_i | X = x_i, \theta) = \sum_{i=1}^{N} \theta^\top x_i T(y_i) - A(\theta), \tag{4}$$

which, by the convexity of log-normalizers is a convex objective which we can maximize using gradient methods. We can use Equation (3) to simplify the computation of the gradient.

MAP inference. An alternative for the maximum likelihood estimate (MLE) for θ is the maximum a posteriori (MAP) estimate. While the MLE only accounts for the likelihood of the data, MAP also accounts for the prior on θ. MAP estimation consists of finding a mode of the posterior (Equation (1)). The objective (in log space) is

$$\log p(X = x|\theta) + \log p(\theta). \tag{5}$$

The log prior can be seen as a form or regularization. For example, an isotropic Gaussian prior corresponds to ℓ_2-regularization.

Conditional models and pseudo-likelihood. A model of variables $\{X_1, \cdots, X_n\}$ is conditionally specified (Arnold, Castillo, Sarabia, et al., 2001), if the relationship between the variables is given in terms of conditional distributions (for $j = 1 : n$, $X_j \sim p(X_j | X_{-j})$). If one drew a graph with nodes X_j and edges (i, j) whenever X_i is in the conditioning set of X_j, the difficulty of working with conditionally specified models comes from the fact

that the graph can contain cycles. If the graph is acyclic, the joint distribution over the variables X_j can be written as the product of the conditional distributions. The conditional distributions can be understood as a specification of how the joint distribution factorizes. In contrast, when there are cycles, the joint distribution is unknown, and in some cases no joint distribution exists. The joint distribution needs to be consistent with the conditionals specified. Consistency means that manipulating the joint distribution to compute conditional distribution of the variables coincides with the conditionals specified.

One can work with conditional models even without access to the joint distribution. Arnold, Castillo, Sarabia, et al., (2001) suggests estimating the parameters of a conditional model by optimizing a loss called the pseudo-likelihood. It is the sum of the log conditional likelihoods of the observations.

$$\mathcal{L} = \sum_i \log p(x_i | x_{-i}) \tag{6}$$

In the case that the graph is acyclical, maximizing the pseudo-likelihood is equicalent to maximizing the likelihood of the model. In the subsequent Chapters, we work with conditional models where additionally to specifying the conditional distributions, we also specify priors on the model variables. The regularized pseudo-likelihood is the sum of the log conditional distributions plus the log priors. The log priors act as regularizers. The regularized pseuso-likelihood is to MAP inference as pseudo likelihood relates to maximum likelihood.

Neural networks. A feed-forward neural network is a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^o$ with network parameters θ. For any given input $x \in \mathbb{R}^d$, the neural network alternates between

linear projections (in the form of matrix multiplications and additions of bias vectors) and the application of elementwise non-linearities to produce the output $f(x) \in \mathbb{R}^o$. The parameters θ are the weight matrices and bias vectors of the consecutive neural network layers. By changing θ (usually using stochastic gradient descent on an appropriately chosen loss function) the neural network can be trained to approximate a specific function of interest. The neural network's modeling capacity can be increased either by adding layers of by increasing the dimension of the intermediate layers. As the network becomes larger, it has the capacity to approximate a larger class of functions.

Neural language models. A language model is a probability distribution over text. A good language model will ideally assign high probability to plausible sentences that are grammatically correct while placing only little probability mass on sentences that are wrong. In many language models, the joint distribution over words in a sentence is factorized into n-gram probabilities, e.g. for a trigram language model, the probability of a sentence of length n is

$$p(x_1, \cdots x_n, \text{STOP}) = \prod_{i=1}^{n+1} p(x_i | x_{i-1}, x_{i-2}), \tag{7}$$

where $x_0 = *$ and $x_{-1} = *$ are a special start symbol and $x_{n+1} = \text{STOP}$ is the stop symbol. The factors $p(x_i | x_{i-1}, x_{i-2})$ are called the trigram probabilities. A *neural language model* (Bengio et al., 2003) uses neural networks to parameterize the n-gram probabilities of Equation (7). The weight matrices at the input layer and the output layer of the neural network maintain continuous representations for each word. When the neural networks are trained to predict which word comes next, the weight matrices at the input and output layer need to

learn to map the discrete one-hot representation of a word into a lower dimensional feature space. As a result, they learn a continuous representation for each term. The representation can be read off directly from the weight matrices.

The neural approach to language modeling addresses the curse of dimensionality. Rather than maintaining a probability table over all possible n-grams (with a vocabulary size of V that would result in a table of size $O(V^n)$), and needing to observe all combinations that are supposed to have non-zero probability, the neural language model shares information between sentences with a similar representation. According to Bengio et al., (2003), each training sentence informs the model about a combinatorial number of other sentences when the words the sentences contain have similar representations.

Chapter 1

Bernoulli Embeddings

Bernoulli embeddings (B-EMBS) are a conditional model of text, closely related to word2vec (Mikolov et al., 2013a). When we fit B-EMBS to text data, we estimate an embedding for each word. The embeddings are vector representations of the words that capture their co-occurrence statistics and the distances in the embedding space are often interpreted as a measure of semantic similarity. In this chapter, we describe the components of a B-EMB model and an objective that we can optimize to learn the embeddings. We then discuss how existing methods relate to B-EMBS.

The components of a B-EMB model are its latent variables (the embeddings), and a conditional distribution for each observed data point. A B-EMB is an instance of an exponential family embedding model, which will be presented in the next chapter. B-EMBS can be used to model text or other binary data and we first describe the form of the data, before specifying the full model and how we fit it.

The text data is a sequence of words $(x_1, \ldots, x_N)$ from a vocabulary of size V. Each word $x_i \in \{0, 1\}^V$ is an indicator vector (also called a "one-hot" vector). It has one nonzero entry at v, where v is the vocabulary term at position i.

Each data point has a *context*. The context is a modeling choice and must be set in advance. When modeling text data, the context of each word is its neighborhood; Each word

is modelled conditionally on the words that come before and after. Typical context sizes range between 2 and 10 words and are set in advance. Let c_i be the indices of the words before and after position i in the text and let $\boldsymbol{x}_{c_i}$ denote the collection of data points indexed by those positions.

The model has an embedding vector $\rho_v \in \mathbb{R}^K$ and a context vector $\alpha_v \in \mathbb{R}^K$ for each unique term in the vocabulary, $v = 1, \ldots, V$. These vectors encode the semantic properties of words, and they are used to parameterize the conditional probability of a word given its context. These are the embeddings we wish to learn.

B-EMBS parameterize the conditional probability of the target word given its context via a linear combination of the embedding vector and the context vectors,

$$p(x_{iv} \mid \boldsymbol{x}_{c_i}) = \text{Bernoulli}\big(\sigma(\rho_v^\top \Sigma_i)\big), \quad \text{with} \quad \Sigma_i \triangleq \sum_{(j,w)\in c_i} \alpha_w x_{jw}. \tag{1.1}$$

Here, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and we have introduced the notation Σ_i for the sum of the context vectors at location i. The context representation Σ_i captures the latent attributes of the context and the relative orientation of the embedding and the context representation determines the probability of the word occurring in a given context. If they are aligned, the probability will be large, whereas if they are almost orthogonal to each other, the probability will be close to 0. Note that Equation (1.1) does not impose the constraint that the sum over the vocabulary words $\sum_v p(x_{iv} = 1 \mid \boldsymbol{x}_{c_i})$ must be 1. This significantly alleviates the computational complexity (Mikolov et al., 2013a; Rudolph et al., 2016).[1]

[1]Multinomial embeddings (Rudolph et al., 2016) model each indicator vector x_i with a categorical conditional distribution, but this requires expensive normalization in form of a softmax. For computational efficiency, one can replace the softmax with the hierarchical softmax (Morin and Bengio, 2005; Mnih and Hinton, 2009; Mikolov et al., 2013a) or employ noise contrastive estimation (Gutmann and Hyvärinen, 2010;

Our goal is to learn the embedding vectors ρ_v and the context vectors α_v from the text by maximizing the log probability of words given their contexts. The data contains N pairs (x_i, c_i) of words and their contexts, and thus we can form the objective function $\mathcal{L}(\boldsymbol{\rho}, \boldsymbol{\alpha})$ as the sum of $\log p(x_{iv} \mid \boldsymbol{x}_{c_i})$ for all instances and vocabulary words,

$$\mathcal{L}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \sum_{v=1}^{V} \log p(x_{iv} \mid \boldsymbol{x}_{c_i}) + \log p(\boldsymbol{\rho}) + \log p(\boldsymbol{\alpha}). \tag{1.2}$$

This sum over the conditional log-likelihoods is called a pseudo-likelihood (Arnold, Castillo, Sarabia, et al., 2001). We regularize the objective with functions that look like log-priors. A Gaussian prior corresponds to ℓ_2-regularization.

The sum over the log conditional probabilities can be broken down into the likelihood of positive examples ($x_{iv} = 1$) and negative examples ($x_{iv} = 0$),

$$\sum_{i=1}^{N} \sum_{v=1}^{V} \log p(x_{iv} \mid \boldsymbol{x}_{c_i}) = \sum_{i=1}^{N} \left(\sum_{v:\, x_{iv}=1} \log \sigma(\rho_v^\top \Sigma_i) + \sum_{v:\, x_{iv}=0} \log \sigma(-\rho_v^\top \Sigma_i) \right). \tag{1.3}$$

If we hold all the context vectors α_v fixed, then Equation (1.3) is the objective of V independent logistic regressors, each predicting whether a word appears in a given context or not using the context representations Σ_i as covariates. The positive examples are those where word v actually appeared in a given context; the negative examples are those where v did not appear. It is the context vectors that couple the V binary classifiers together.

Mnih and Kavukcuoglu, 2013). B-EMBS relax the one-hot constraint of x_i, and work well in practice. The point of this section is to expose the relationship to negative sampling (Mikolov et al., 2013a).

Figure 1.1: Word embeddings capture the semantic similarities between words. The figure shows a t-SNE projection (Maaten and Hinton, 2008) of the embedding vectors learned by a B-EMB fitted to Wikipedia articles.

The most expensive term in the objective is

$$\mathcal{L}_{\text{neg}} = \sum_{i=1}^{N} \sum_{v:\, x_{iv}=0} \log \sigma(-\rho_v^\top \Sigma_i), \tag{1.4}$$

the contribution of the zeroes to the conditional log likelihood. The objective is cheaper if we subsample the zeros. Rather than summing over all words which are not at position i, we

sum over a subset of n negative samples $\mathcal{S}_i$ drawn at random. Mikolov et al., (2013a) call this negative sampling and recommend sampling from $\hat{p}$, the unigram distribution raised to the power of 0.75.

With negative sampling, we redefine $\mathcal{L}_{\text{neg}}$ as

$$\mathcal{L}_{\text{neg}} = \sum_{i=1}^{N} \sum_{v \in \mathcal{S}_i} \log \sigma(-\rho_v^\top \Sigma_i). \tag{1.5}$$

This sum has fewer terms and reduces the contribution of the zeros to the objective. In a sense, this incurs a bias—the expectation with respect to the negative samples is not equal to the original objective—but "downweighting the zeros" can improve prediction accuracy (Hu, Koren, and Volinsky, 2008; Liang et al., 2016) and leads to significant computational gains.

An implementation of B-EMBS is available at https://github.com/mariru/exponential_family_embeddings. In Chapter 3, we present multiple extensions of B-EMBS. An empirical study of B-EMBS and its extensions will follow then. Here, we give one qualitative result. Figure 1.1 shows the embedding vectors of a B-EMB that has been fitted to a collection of Wikipedia articles (http://mattmahoney.net/dc/enwik8.zip). The 100 dimensional embedding vectors have been projected into 2-D using t-SNE (Maaten and Hinton, 2008). The embedding captures the semantic similarities between words as similar words are close to each other in the embedding space. For example, on the left is a cluster of common male first names including GEORGE, JAMES, and WILLIAM. Towards the top of the figure, on the right of a cluster of country names (GERMANY, FRANCE, CHINA, etc.) is a small cluster of directions (EAST, WEST, and CENTRAL).

1.1. Connections

In this section, we draw connections between B-EMBS and other word embedding methods. Specifically, we derive the relationship between B-EMBS and flavors of word2vec (Mikolov et al., 2013a), which is currently one of the most popular tools for learning embeddings.

There are multiple ways to implement word2vec. First, there is a choice of the objective. Second, there are several ways to approximate the objective to get a scalable algorithm. In this section, we describe the two objectives, continuous bag of words (CBOW) and skip-gram, and we focus on negative sampling as the method of choice to achieve scalability. We describe the similarities and differences between fitting a B-EMB and these two objectives. In summary, under certain assumptions a B-EMB is equivalent to CBOW with negative sampling, and it is related to skip-gram through Jensen's inequality.

B-EMB $\equiv$ CBOW (negative sampling)

First, we explain how a B-EMB and CBOW with negative sampling are related. Consider the full B-EMB objective without regularization,

$$\mathcal{L}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_i \left(\sum_{v:\, x_{iv}=1} \log \sigma(\rho_v^\top \Sigma_i) + \sum_{v:\, x_{iv}=0} \log \sigma(-\rho_v^\top \Sigma_i) \right). \tag{1.6}$$

In most cases, the summation over negative examples ($x_{iv} = 0$) is computationally expensive to compute. To address that, we form an unbiased estimate of that term by subsampling a

random set $\mathcal{S}_i$ of terms and rescaling by $\frac{V-1}{|\mathcal{S}_i|}$,

$$\widehat{\mathcal{L}}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_i \left(\sum_{v:\, x_{iv}=1} \log \sigma(\rho_v^\top \Sigma_i) + \gamma \frac{V-1}{|\mathcal{S}_i|} \sum_{v \in \mathcal{S}_i} \log \sigma(-\rho_v^\top \Sigma_i) \right). \tag{1.7}$$

Here, we have introduced an auxiliary coefficient γ. The estimate is unbiased only for $\gamma = 1$; however, Rudolph et al., (2016) showed that downweighting the contribution of the zeros works better in practice.[2] In particular, if we set the downweight factor as $\gamma = \frac{|\mathcal{S}_i|}{V-1}$, we recover the objective of CBOW with negative sampling,

$$\widehat{\mathcal{L}}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_i \left(\sum_{v:\, x_{iv}=1} \log \sigma(\rho_v^\top \Sigma_i) + \sum_{v \in \mathcal{S}_i} \log \sigma(-\rho_v^\top \Sigma_i) \right) \equiv \mathcal{L}_{\text{CBOW}}(\boldsymbol{\rho}, \boldsymbol{\alpha}) \tag{1.8}$$

There are two more subtle theoretical differences between both. The first difference is that B-EMBS include a regularization term for the embedding vectors, whereas CBOW does not. The second difference is that, in B-EMBS, we need to draw a new set of negative samples $\mathcal{S}_i$ at each iteration of the gradient ascent algorithm (because we form a noisy estimator of the downweighted objective). In contrast, in CBOW with negative sampling, the samples $\mathcal{S}_i$ are drawn once in advance and then hold fixed. In practice, for large datasets, we have not observed significant differences in the performance of both approaches. For simplicity, we draw the negative samples $\mathcal{S}_i$ only once.

CBOW (negative sampling) $\geq$ skip-gram (negative sampling)

Now we show how CBOW and skip-gram are related (considering negative sampling for

[2] This is consistent with the approaches in recommender systems (Hu, Koren, and Volinsky, 2008).

both). Recall that the objective of CBOW is to predict a target word from its context, while the skip-gram objective is to predict the context from the target word. Negative sampling breaks the constraint that the sum of the probability of each word must equal one, and instead models probabilities of the individual entries of the one-hot vectors representing the words.

When we apply negative sampling, the CBOW objective becomes Equation (1.8). The skip-gram objective is given by

$$\mathcal{L}_{\text{skip-gram}}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_{(i,v):\, w_{iv}=1} \left(\sum_{v' \in c_i} \log \sigma\big(\rho_v^\top \alpha_{v'}\big) + \sum_{v' \in \mathcal{S}_i} \log \sigma\big(-\rho_v^\top \alpha_{v'}\big) \right), \tag{1.9}$$

That is, for each target term x_{iv}, the CBOW objective has one term while the skip-gram objective has $|c_i|$ terms. Consider a term (i, v) for which $x_{iv} = 1$. We take the corresponding CBOW term from Equation (1.8) and we apply Jensen's inequality to obtain the corresponding skip-gram term in Equation (1.9):

$$\log \sigma(\rho_v^\top \Sigma_i) = \log \sigma \left(\rho_v^\top \sum_{v' \in c_i} \alpha_{v'} \right) \geq \sum_{v' \in c_i} \log \sigma \left(\rho_v^\top \alpha_{v'} \right). \tag{1.10}$$

Here, we have made use of the concavity of the $\log \sigma(\cdot)$ function. In general, this is a consequence of the convexity of the log-normalizer of exponential family distributions.

This holds for the "positive" examples x_{iv}. As for the negative examples ($x_{iv} = 0$), the comparison is not as straightforward, because the choice of terms in Equations (1.8) and (1.9) is not exactly the same; Equation (1.8) holds v' fixed and draws v from the noise distribution, while Equation (1.9) holds v fixed and draws v' from the noise distribution.

Chapter 2

Exponential Family Embeddings

In the previous chapter, we have presented a model for learning word embeddings. Word embeddings are lower dimensional, distributed representations of words — objects that are inherently high dimensional and discrete. While in the original space all vocabulary terms are orthogonal to each other, in the embedding space we can compute similarities between words. Many other data domains include discrete objects we might be interested in learning embeddings of. This chapter is about a general model class which extends the task of learning embeddings to other applications.

Consider, for example, data of peoples' shopping behaviour. As the customers check out at the register, the type and the quantity of the items purchased is recorded. The items are discrete objects, and we might want to learn embeddings for them. One way to learn item embeddings, would be to download Wikipedia articles about these items and to run Bernoulli embeddings (B-EMBS) on this text. Can we instead learn the embeddings directly from the shopping data? Intuitively, The data of which items are purchased together in which quantities contains information on item similarities and in this chapter we present a model to distill the purchasing patterns into item embeddings.

Other questions we hope to address with the models presented in this chapter include: Can we learn embeddings of zebrafish neurons from recordings of their neural activity?

Can we learn embeddings of movies from data on how people rated different movies? To address these questions we develop a model class which encapsulates the main ideas of word embeddings, but is general enough to be applicable to the different data described here and beyond. The model class is called exponential family embeddings (EF-EMBS), and the B-EMB model described in the previous chapter is a special case.

An EF-EMB model has three ingredients. The embedding parameters (one embedding and one context vector for each object to be embedded), a context function which defines the conditioning set for each observation, as well as a conditional distribution from the exponential family. They are a class of conditionally specified models.

The conditional models are fit to data using a pseudo-likelihood objective. The exponential family representation ensures that the updates have to be derived only once, for the general model class, and properties of the exponential family distribution simplify the gradients. Monte carlo estimation of the gradients scale the task of learning embeddings to large datasets. For sparse data such as text or count data from a recommendation system, negative sampling provides additional speed up. Negative sampling is a technique for subsampling minibatches for the SGD procedure. The resulting approach aggressively downsamples observations that are 0 and is hence biased towards using mostly nonnegative observations.

After introducing the applications we study using EF-EMBS, we present the generla model class. We then provide two specific example models. A Gaussian embedding model which we use to model the co-firing patters of zebrafish neurons as well as a Poisson embedding model, which we use to model the co-purchasing patterns of grocery items.

In an empirical study, we quantitatively compare EF-EMB to existing methods in terms of their capacity of predicting held-out data. We also report qualitative findings.

Applications. In the presentation of the EF-EMB model class and the specific example models we provide below, we have the following applications in mind.

Brain activity from zebrafish larvae. Zebrafish larvae are see-through which makes them an excellent model organism for whole-brain imaging. Neuro scientist have developed techniques to record the activity of almost all the neurons at very high spatial resolution. The resulting data is the activity of 10K neurons over time (ca. 3000 time slices). The goal is to learn embeddings of the neurons from their activity patterns.

Shopping data. Another dataset we consider is the shopping behaviour of people buying groceries at a major supermarket chain. We observe which items are bought together in each trip and the goal is to use that information to learn embedding of the groceries.

Recommendation systems. We also consider movie ratings. For example, the Netflix 100K dataset contains 100K user ratings of movies. We show that with a model from the EF-EMB class, we can learn embeddings of the movies from such data.

2.1. Model Description of Exponential Family Embeddings

Consider data which is organized in a matrix X with entries x_{iv} with $i \in \{1, \cdots, I\}$, $v \in \{1, \cdots, V\}$. For now, we make no assumption on the type of data. In text it can be binary, while in the other applications we study it can either be real valued or count data.

Assume, our goal is to learn embeddings for each of the objects indexed by v.[1] In the neuro science application for example, v indexes the neurons and $x_{i,v}$ is the activity of

[1]For notational convenience, we fix the index to be embedded to the second index (the columns), but the model derivations below also apply when the object to be embedded correspond to the first index of the data matrix. In that case, the data matrix could simply be transposed.

neuron v at time i. In the movie rating application x_{iv} is the rating user i gave movie v.

For each v, the model has two types of vectors. Each v is associated with a context vector $\alpha_v \in \mathbb{R}^K$ and an embeddings vector $\rho_v \in \mathbb{R}^K$. The goal is to learn these vectors. They are treated as latent variables and are the parameters of the conditional distributions of the data.

Each data-point is modelled conditionally on a *context*. The context is a modeling choice and depends on the application. In language, the context is nearby words. In the neuro science application, the context of the activity of a neuron is the activity of neurons which are nearby in the zebrafish brain. In the shopping application, the context for how many of a specific grocery item were bought, are the other groceries that were bought in the same trip.

The conditional distribution for each observation given its context is an exponential family distribution

$$p(x_{iv} \mid \boldsymbol{x}_{c_{iv}}) = \text{ExpFam}(\eta_{iv}(\boldsymbol{x}_{civ}), t(x_{iv})), \tag{2.1}$$

with natural parameter η_{iv} and sufficient statistics $t(x_{iv})$. The natural parameter is a function of the context and the embedding and context vectors. The observations in the context and the context vectors are combined into a weighted average called the context representation,

$$\Sigma_{iv} = \sum_{(j,w) \in c_{iv}} \alpha_w x_{jw}. \tag{2.2}$$

The context representation captures the latent attributes of the context. It is combined with

the embedding of item v, to produce the natural parameter

$$\eta_{iv} = f\left(\rho_v^\top \Sigma_{iv}\right). \tag{2.3}$$

The link function $f(\cdot)$ is also a modeling choice. It can often be the identity or the logarithm. Choosing it is akin to choosing the link function in a generalized linear model. In chapter 3.3 we develop deep embeddings where η_{iv} is a multi-layer perceptron.

In the previous chapter, we presented the B-EMB, a specific instance from the EF-EMB model class for modeling text. We provide two more example models before describing how to fit these conditional models to data. As a first example, we present a Gaussian embedding (G-EMB) for analyzing real observations from a neuroscience application; we also introduce a nonnegative version, the nonnegative Gaussian embedding (NG-EMB). We then develop two Poisson embedding models, Poisson embedding (P-EMB) and additive Poisson embedding (AP-EMB), for analyzing count data; these have different link functions and we use them to study shopping data as well as movie ratings data. An empirical study of these models will follow in Section 2.1.2 after a description of how to fit EF-EMB in Section 2.1.2.

For convenience, the model name acronyms are in Table 2.1.

EF-EMB	*exponential family embedding*
G-EMB	*Gaussian embedding*
NG-EMB	*nonnegative Gaussian embedding*
P-EMB	*Poisson embedding*
AP-EMB	*additive Poisson embedding*
B-EMB	*Bernoulli embedding*

Table 2.1: Acronyms used for exponential family embeddings.

2.1.1 Example: Gaussian Embeddings for Real Valued Observations

Consider a recording of the neural activity of a large population of zebrafish neurons (specifics are described in Section 2.1.2) and let x_{iv} be the neural activity of neuron v at time slice i. The values of x_{iv} are nonnegtaive real values. The goal is to model the similarity between neurons in terms of their behavior, to embed each neuron in a latent space such that neurons with similar behavior are close to each other.

We use a G-EMB to do so. The model has a context vector α_v and an embedding vector ρ_v for each neuron. These parameters are combined as in Equation (2.3) to form the mean of the conditional distribution of each observation x_{iv}. As conditional distribution we choose a univariate Gaussian with constant variance σ^2.

An important modeling choice is the conditioning set for each observation. We decide to model each neuron's activity in the context of the activity of nearby neurons. We use the 3D coordinates of the neurons to identify the K-nearest neighbours (KNN) of each neuron. This allows us to define a context $c_{iv} = \{(i, w)|w \in \text{KNN}(v)\}$ which varies with each neuron, but is constant over time.

These ingredients, along with a regularizer, combine to form a neural embedding objective. G-EMB uses ℓ_2 regularization (i.e., a Gaussian prior); NG-EMB constrains the vectors to be nonnegative (ℓ_2 regularization on the logarithm. i.e., a log-normal prior).

2.1.2 Example: Poisson Embeddings for Count Data

For a matrix of count data with entries x_{iv}, we develop P-EMB. As examples, x_{iv} could represent how many groceries of type v were bought in shopping trip i, or which rating

person i gave movie v. Again, each item (grocery item or movie) is associated with two types of parameters; an embedding vector $\rho_v \in \mathbb{R}^K$ and a context vector $\alpha_v \in \mathbb{R}^K$.

The context for each observation are the other entries in the same column i. For shopping data, this corresponds to the other items bought on the same trip and for movies, this corresponds to the other movies rated by the same user. Formally, $c_{iv} = \{(i, w)|w \neq v\}$. The data is typically sparse (on each trip a person buys a small fraction of items available at the store), so using this context function and context vectors to form a context representation as in Equation (2.2) will require to sum only over the terms x_{iw} that are nonzero. So even though for each i all items $w \neq v$ are in the context c_{iv}, the effective context of item v (set of items w s.t. $x_{iw} \neq 0$) typically differs between trips where v was bought.

We use conditional Poisson distributions to modelthe count data. The sufficient statistic of the Poisson is $t(x_{iv}) = x_{iv}$, and its natural parameter is the logarithm of the rate (i.e., the mean). We set the natural parameter as in Equation (2.3).

We explore two choices for the link function. P-EMB uses an identity link function. Since the conditional mean is the exponentiated natural parameter, this implies that the context items contribute multiplicatively to the mean. For P-EMB we use a Gaussian prior on the embeddings, which corresponds to ℓ_2 regularization. (We use ℓ_2-regularization on the embeddings.) Alternatively, we can use a lognormal prior which constrains the parameters to be nonnegative and set the link function $f(\cdot) = \log(\cdot)$. This is AP-EMB, a model with an additive mean parameterization. The choice of a lognormal prior corresponds to ℓ_2-regularization in log-space. AP-EMB only captures positive correlations between items.

2.2. Model Training

Since the model is conditionally specified (Arnold, Castillo, Sarabia, et al., 2001), there is no guarantee that a joint, consistent with the specified conditionals exists. Whether a consistent joint exists will depend on the choice of exponential family distribution (for example when it is Bernoulli, a joint exists for sure) or on additional constraints on the parameters (for example for a Poisson embedding for a joint to exist we need to impose additional symmetry constraints on the parameters $\alpha_v = \rho_v$). The choice of context also matters. If there are no cyclical context relationships[2], the joint is simply factorized according to Equation (2.1) and we can proceed with standard inference techniques.

The pseudo-likelihood objective. In practice, we disregard whether a joint exists or not an train using a regularized pseudo-likelihood objective. The objective sums the log conditional probabilities of each data point, adding regularizers for the embeddings and context vectors. We use log probability functions as regularizers, e.g., a Gaussian probability leads to ℓ_2 regularization. We can also use regularizers to constrain the embeddings. For example, a log-normal prior forces the embeddings to be non-negative. The objective is

$$\mathcal{L}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_{i=1}^{I} \sum_{v=1}^{V} \left(\eta_{iv}^{\top} t(x_{iv}) - a(\eta_{iv}) \right) + \log p(\boldsymbol{\rho}) + \log p(\boldsymbol{\alpha}). \tag{2.4}$$

Equation (2.4) can be seen as a likelihood function for a bank of generalized linear models (GLMS) (McCullagh and Nelder, 1989). Each data point is modeled as a response conditional on its "covariates," which are the context representations Equation (2.2); the coefficient for

[2] consider the graph where each observation is a node and draw a directed edge from each observation to all observations in its context. "No cyclical context relationships" means that this graph is acyclic.

each response is the embedding itself. We use properties of exponential families and results around GLMS to derive efficient algorithms for EF-EMB models.

Automatic differentiation and model specific gradients. We maximize the objective Equation (2.4) with respect to the embeddings and context vectors using stochastic gradient descent (SGD). SGD is a gradient-based optimization technique and we can either derive the gradients of Equation (2.4) manually or resort to automatic differentiation.

For the manual derivation we use the identity for exponential family distributions that the derivative of the log-normalizer is equal to the expectation of the sufficient statistics, i.e., $\mathbb{E}[t(X)] = \nabla_\eta a(\eta)$ (Equation (3)). Using this result, the gradient with respect to the embedding ρ_v is

$$\nabla_{\rho_v} \mathcal{L} = \sum_{i=1}^{I} \sum_{w=1}^{V} \big(t(x_{iw}) - \mathbb{E}[t(x_{iw})]\big) \nabla_{\rho_v} \eta_{iw} + \nabla_{\rho_v} \log p(\rho_v). \tag{2.5}$$

The gradient with respect to α_v has the same form.

The gradient in Equation (2.5) can involve a sum of many terms and be computationally expensive to compute. To alleviate this, we follow noisy gradients using SGD. We form a subsample $\mathcal{S}$ of the $I \times V$ terms in the summation, i.e.,

$$\widehat{\nabla}_{\rho_v} \mathcal{L} = \frac{I}{|\mathcal{S}|} \sum_{(i,w) \in \mathcal{S}} \big(t(x_{iw}) - \mathbb{E}[t(x_{iw})]\big) \nabla_{\rho_v} \eta_{iw} + \nabla_{\rho_v} \log p(\rho_v), \tag{2.6}$$

where $|\mathcal{S}|$ denotes the size of the subsample and where we scaled the summation to ensure an unbiased estimator of the gradient. Equation (2.6) reduces computational complexity when $|\mathcal{S}|$ is much smaller than the total number of terms. At each iteration of SGD we compute

noisy gradients with respect to ρ_v and α_v (for each v) and take gradient steps according to a step-size schedule. We use Adagrad (Duchi, Hazan, and Singer, 2011a) to set the step-size.

For the models we presented so far in this chapter (G-EMB, NG-EMB, P-EMB and AP-EMB) the model specific gradients can be derived and implemented with a reasonable amount of effort. As we will see in later chapters, it is automatic differentiation tools that allow us to extend EF-EMB in interesting ways. In we will see variations of EF-EMBs that either use a different parametrization of η_{iv} (e.g. using neural networks) or more structured priors $p(\boldsymbol{\rho})$ and $p(\boldsymbol{\alpha})$. The resulting objectives will still be of the same form as in Equation (2.4). The capabilities of automatic differentiation allow us to extend EF-EMB in ways that achieve desired modeling capacity, without having to consider the burden of deriving model specific gradients when making modeling choices.

Negative Sampling. When the data is sparse (because many x_{iv} are unobserved or 0), we can split the gradient into the summation of two terms: one term corresponding to all data entries (i, v) for which $x_{iv} \neq 0$, and one term corresponding to those data entries $x_{iv} = 0$. We compute the first term of the gradient exactly—when the data is sparse there are not many summations to make—and we estimate the second term by subsampling the zero entries. Compared to computing the full gradient, this reduces the complexity when most of the entries x_{iv} are zero, but it retains the strong information about the gradient that comes from the non-zero entries. This relates to negative sampling, which is used to approximate the skip-gram objective (Mikolov et al., 2013a).

2.3. Empirical Study of Exponential Family Embeddings

We study exponential family embedding (EF-EMB) models on real-valued and count-valued data, and in different application domains—computational neuroscience, shopping behavior, and movie ratings. We present quantitative comparisons to other dimension reduction methods and illustrate how we can glean qualitative insights from the fitted embeddings.

2.3.1 Real Valued Data: Neural Data Analysis

Data. We analyze the neural activity of a larval zebrafish, recorded at single cell resolution for 3000 time frames (Ahrens et al., 2013). Through genetic modification, individual neurons express a calcium indicator when they fire. The resulting calcium imaging data is preprocessed by a nonnegative matrix factorization to identify neurons, their locations, and the fluorescence activity $x_t^* \in \mathbb{R}^N$ of the individual neurons over time (Friedrich et al., 2015). Using this method, our data contains 10,000 neurons (out of a total of 200,000).

We fit all models on the lagged data $x_t = x_t^* - x_{t-1}^*$ to filter out correlations based on calcium decay and preprocessing.[3] The calcium levels can be measured with great spatial resolution but there is heavy aliasing in the temporal dimension; the firing rate is much higher than the sampling rate. Hence we ignore all "temporal structure" in the data and model the simultaneous activity of the neurons. We use the Gaussian embedding (G-EMB) and nonnegative Gaussian embedding (NG-EMB) to model the lagged activity of the neurons conditional on the lags of surrounding neurons. We study context sizes $c \in \{10, 50\}$ and latent dimension $K \in \{10, 100\}$.

[3] We also analyzed unlagged data but, as expected, all methods have better reconstruction performance on the lagged data.

	single neuron held out		25% of neurons held out	
Model	$K = 10$	$K = 100$	$K = 10$	$K = 100$
FA	0.290 ± 0.003	0.275 ± 0.003	0.290 ± 0.003	0.276 ± 0.003
G-EMB (c=10)	0.239 ± 0.006	0.239 ± 0.005	0.246 ± 0.004	0.245 ± 0.003
G-EMB (c=50)	0.227 ± 0.002	$\mathbf{0.222 \pm 0.002}$	0.235 ± 0.003	$\mathbf{0.232 \pm 0.003}$
NG-EMB (c=10)	0.263 ± 0.004	0.261 ± 0.004	0.250 ± 0.004	0.261 ± 0.004

Table 2.2: Analysis of neural data: mean squared error and standard errors of neural activity (on the test set) for different models. Both EF-EMB models significantly outperform FA; G-EMB is more accurate than NG-EMB.

Models. We compare EF-EMB to probabilistic factor analysis (FA), fitting K-dimensional factors for each neuron and K-dimensional factor loadings for each time frame. In FA, each entry of the data matrix is a Gaussian with mean equal to the inner product of the corresponding factor and factor loading.

Evaluation. We train each model on a random sample of 90% of the lagged time frames and hold out 5% each for validation and testing. With the test set, we use two types of evaluation. (1) *Leave one out*: For each neuron x_i in the test set, we use the measurements of the other neurons to form predictions. For FA this means the other neurons are used to recover the factor loadings; for EF-EMB this means the other neurons are used to construct the context. (2) *Leave* 25% *out*: We randomly split the neurons into 4 folds. Each neuron is predicted using the three sets of neurons that are out of its fold. (This is a more difficult task.) Note in EF-EMB, the missing data might change the size of the context of some neurons.

Results. Table 3.9 reports both types of evaluation. The EF-EMB models significantly outperform FA in terms of mean squared error on the test set. G-EMB obtains the best results with 100 components and a context size of 50. Figure 2.1 illustrates how to use the learned embeddings to hypothesize connections between nearby neurons.

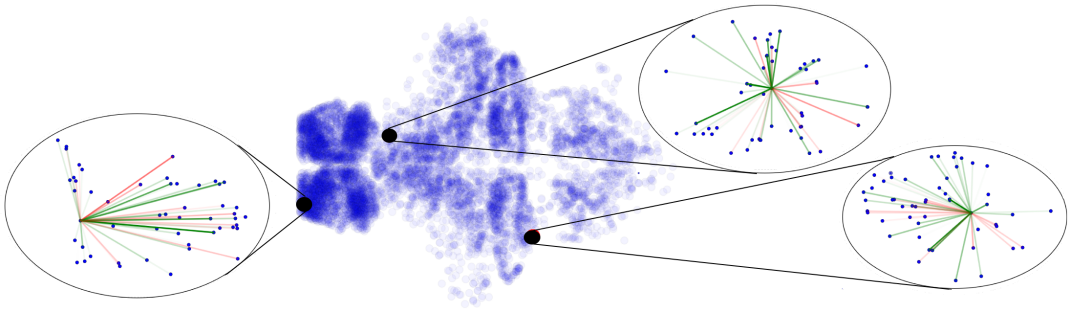


Figure 2.1: Top view of the zebrafish brain, with blue circles at the location of the individual neurons. We zoom on 3 neurons and their 50 nearest neighbors (small blue dots), visualizing the "synaptic weights" learned by a G-EMB model ($K = 100$). The edge color encodes the inner product of the neural embedding vector and the context vectors $\rho_n^\top \alpha_m$ for each neighbor m. Positive values are green, negative values are red, and the transparency is proportional to the magnitude. With these weights we can form hypotheses about how nearby neurons are connected.

(a) Market basket analysis.

Model	$K = 20$	$K = 50$	$K = 100$
P-EMB	-7.497 ± 0.007	-7.284 ± 0.008	-7.199 ± 0.008
P-EMB (DW)	-7.110 ± 0.007	-6.994 ± 0.007	$\mathbf{-6.950 \pm 0.007}$
AP-EMB	-7.868 ± 0.005	-8.191 ± 0.004	-8.414 ± 0.003
HPF	-7.740 ± 0.008	-7.626 ± 0.007	-7.626 ± 0.007
Poisson PCA	-8.314 ± 0.009	-9.51 ± 0.01	-11.01 ± 0.01

(b) Movie ratings.

Model	$K = 20$	$K = 50$	$K = 100$
P-EMB	$\mathbf{-5.691 \pm 0.006}$	-5.726 ± 0.006	-5.726 ± 0.005
P-EMB (DW)	-5.790 ± 0.003	-5.798 ± 0.003	-5.798 ± 0.003
AP-EMB	-5.964 ± 0.003	-6.082 ± 0.002	-6.118 ± 0.002
HPF	-5.787 ± 0.006	-5.844 ± 0.006	-5.859 ± 0.006
Poisson PCA	-5.908 ± 0.006	-6.394 ± 0.008	-7.50 ± 0.01

Table 2.3: Comparison of predictive log-likelihood between P-EMB, AP-EMB, hierarchical Poisson factorization (HPF) (Gopalan, Hofman, and Blei, 2015), and Poisson principal component analysis (PCA) (Collins, Dasgupta, and Schapire, 2001) on held out data. The P-EMB model outperforms the matrix factorization models in both applications. For the shopping data, downweighting the zeros improves the peformance of P-EMB.

2.3.2 Count data: Market Basket Analysis and Movie Ratings

We study the Poisson models Poisson embedding (P-EMB) and additive Poisson embedding (AP-EMB) on two applications: shopping and movies.

Market basket data. We analyze the IRI dataset[4] (Bronnenberg, Kruger, and Mela, 2008), which contains the purchases of anonymous households in chain grocery and drug stores. It contains 137, 632 trips in 2012. We remove items that appear fewer than 10 times, leaving a dataset with 7, 903 items. The context for each purchase is the other purchases from the same trip.

MovieLens data. We also analyze the MovieLens-100K dataset (Harper and Konstan, 2015), which contains movie ratings on a scale from 1 to 5. We keep only positive ratings, defined to be ratings of 3 or more (we subtract 2 from all ratings and set the negative ones to 0). The context of each rating is the other movies rated by the same user. After removing users who rated fewer than 20 movies and movies that were rated fewer than 50 times, the dataset contains 777 users and 516 movies; the sparsity is about 5%.

Models. We fit the P-EMB and the AP-EMB models using number of components $K \in \{20, 50, 100\}$. For each K we select the Adagrad constant based on best predictive performance on the validation set. In these datasets, the distribution of the context size is heavy tailed. To handle larger context sizes we pick a link function for the EF-EMB model which rescales the sum over the context in Equation (2.2) by the context size (the number of terms in the sum). We also fit a P-EMB model that artificially downweights the contribution of the zeros in the objective function by a factor of 0.1, as done by Hu, Koren, and Volinsky, (2008) for matrix factorization. We denote it as "P-EMB (dw)."

We compare the predictive performance with HPF (Gopalan, Hofman, and Blei, 2015) and Poisson PCA (Collins, Dasgupta, and Schapire, 2001). Both HPF and Poisson PCA factorize

[4]We thank IRI for making the data available. All estimates and analysis in this work, based on data provided by IRI, are by the authors and not by IRI.

the data into K-dimensional positive vectors of user preferences, and K-dimensional positive vectors of item attributes. AP-EMB and HPF parameterize the mean additively; P-EMB and Poisson PCA parameterize it multiplicatively. For the EF-EMB models and Poisson PCA, we use stochastic optimization with ℓ_2 regularization. For HPF, we use variational inference.

Evaluation. For the market basket data we hold out 5% of the trips to form the test set, also removing trips with fewer than two purchased different items. In the MovieLens data we hold out 20% of the ratings and set aside an additional 5% of the non-zero entries from the test for validation. We report prediction performance based on the normalized log-likelihood on the test set. For P-EMB and AP-EMB, we compute the likelihood as the Poisson mean of each nonnegative count (be it a purchase quantity or a movie rating) divided by the sum of the Poisson means for all items, given the context. To evaluate HPF and Poisson PCA at a given test observation we recover the factor loadings using the other test entries we condition on, and we use the factor loading to form the prediction.

Predictive performance. Table 2.3 summarizes the test log-likelihood of the four models, together with the standard errors across entries in the test set. In both applications the P-EMB model outperforms HPF and Poisson PCA. On shopping data P-EMB with $K = 100$ provides the best predictions; on MovieLens P-EMB with $K = 20$ is best. For P-EMB on shopping data, downweighting the contribution of the zeros gives more accurate estimates.

Item similarity in the shopping data. Embedding models can capture qualitative aspects of the data as well. Table 2.4 shows four example products and their three most similar items, where similarity is calculated as the cosine distance between embedding vectors. (These vectors are from P-EMB with downweighted zeros and $K = 100$.) For

Maruchan chicken ramen	**Mountain Dew soda**
Maruchan creamy chicken ramen	Mountain Dew orange soda
Maruchan oriental flavor ramen	Mountain Dew lemon lime soda
Maruchan roast chicken ramen	Pepsi classic soda
Dean Foods 1% milk	**Yoplait strawberry yogurt**
Dean Foods 2% milk	Yoplait apricot mango yogurt
Dean Foods whole milk	Yoplait strawberry orange smoothie
Dean Foods chocolate milk	Yoplait strawberry banana yogurt

Table 2.4: Top 3 similar items to several example queries (bold face). The P-EMB model successfuly captures item similarities.

example, the most similar items to a soda are other sodas; the most similar items to a yogurt are (mostly) other yogurts.

The P-EMB model can also identify complementary and substitutable products. To see this, we compute the inner products of the embedding and the context vectors for all item pairs. A high value of the inner product indicates that the probability of purchasing one item is increased if the second item is in the shopping basket (i.e., they are complements). A low value indicates the opposite effect and the items might be substitutes for each other.

Table 2.5 shows some pairs of items with high inner product of embedding vectors and context vector. The items in the first column have higher probability of being purchased if the item in the second column is in the shopping basket. We can observe that they correspond to items that are frequently purchased together (potato chips and beer, potato chips and frozen pizza, two different sodas).

Similarly, Table 2.6 shows some pairs of items with low inner product. The items in the first column have lower probability of being purchased if the item in the second column is in the shopping basket. We can observe that they correspond to items that are rarely purchased together (detergent and toast crunch, milk and toothbrush), or that are substitutes of each

other (two different brands of snacks, soup, or pasta sauce).

Inner product	Item 1	Item 2
2.12	Diet 7 Up lemon lime soda	Diet Squirt citrus soda
2.11	Old Dutch original potato chips	Budweiser Select 55 Lager beer
2.00	Lays potato chips	DiGiorno frozen pizza
2.00	Coca Cola zero soda	Coca Cola soda
1.99	Snoyfield vanilla organic yogurt	La Yogurt low fat mango

Table 2.5: Market basket: List of several of the items with high inner product values. Items from the first column have higher probability of being purchased when the item in the second column is in the shopping basket.

Inner product	Item 1	Item 2
−5.06	General Mills cinammon toast crunch	Tide Plus liquid laundry detergent
−5.00	Doritos chilli pepper	Utz cheese balls
−5.00	Land O Lakes 2% milk	Toothbrush soft adult (private brand)
−5.00	Beef Swanson Broth soup 48oz	Campbell Soup cans 10.75oz
−4.99	Ragu Robusto sautéed onion & garlic pasta sauce	Prego tomato Italian pasta sauce

Table 2.6: Market basket: List of several of the items with low inner product values. Items from the first column have lower probability of being purchased when the item in the second column is in the shopping basket.

We find that items that tend to be purchased together have high value of the inner product (e.g., potato chips and beer, potato chips and frozen pizza, or two different types of soda), while items that are substitutes have negative value (e.g., two different brands of pasta sauce, similar snacks, or soups from different brands). Other items with negative value of the inner product are not substitutes, but they are rarely purchased together (e.g., toast crunch and laundry detergent, milk and a toothbrush).

Topics in the movie embeddings. The embeddings from MovieLens data identify thematically similar movies. For each latent dimension k, we sort the context vectors by the magnitude of the kth component. This yields a ranking of movies for each component. Tables 2.7 and 2.8 show two example clusters of ranked movies that are learned by our P-EMB model. These rankings were generated as follows. For each latent dimension $k \in \{1, \cdots, K\}$ we sorted the context vectors according their value in this dimension. This gives us a ranking

of context vectors for every k. Tables 2.7 and 2.8 show the 10 top items of the ranking for two different values of k. Similar as in topic modeling, the latent dimensions have the interpretation of topics. We see that sorting the context vectors this way reveals thematic structure in the collection of movies. While Table 2.7 gives a table of movies for children, Table 2.8 shows a cluster of science-fiction and action movies (with a few outliers).

#	Movie Name	Year	Rank
1	Winnie the Pooh and the Blustery Day	1968	0.62
2	Cinderella	1950	0.50
3	Toy Story	1995	0.46
4	Fantasia	1940	0.44
5	Dumbo	1941	0.43
6	The Nightmare Before Christmas	1993	0.37
7	Snow White and the Seven Dwarfs	1937	0.37
8	Alice in Wonderland	1951	0.35
9	James and the Giant Peach	1996	0.35

Table 2.7: Movielens: Cluster for "kids movies".

#	Movie Name	Year	Rank
1	Die Hard: With a Vengeance	1995	1.25
2	Stargate	1994	1.19
3	Star Trek IV: The Voyage Home	1986	1.14
4	Manon of the Spring (Manon des sources)	1986	1.14
5	Fifth Element, The	1997	1.14
6	Star Trek VI: The Undiscovered Country	1991	1.13
7	Under Siege	1992	1.11
8	GoldenEye	1995	1.07
9	Supercop	1992	1.07

Table 2.8: Movielens: Cluster for "science-fiction/action movies".

2.4. Discussion of Exponential Family Embeddings

We described exponential family embeddings (EF-EMBS), conditionally specified latent variable models to extract distributed representations from high dimensional data. In Chapter 1, we showed that continuous bag of words (CBOW) (Mikolov et al., 2013a) is a special case of EF-EMB. Here, we provided examples beyond text: the brain activity of zebrafish, shopping data, and movie ratings. We fit the EF-EMB objective using stochastic gradients. Our empirical study demonstrates that an EF-EMB can better reconstruct data than existing dimensionality-reduction techniques based on matrix factorization. Further, the learned embeddings capture interesting semantic structure. In the next chapter, we present several extensions of EF-EMBS.

2.5. Appendix to Exponential Family Embeddings

To specify the gradients in Equation (2.5) for the stochastic gradient descent (SGD) procedure we need the sufficient statistic $t(x)$, the expected sufficient statistic $\mathbb{E}[t(x)]$, the gradient of the natural parameter with respect to the embedding vectors and the gradient of the regularizer on the embedding vectors. In this appendix we specify these quantities for the models we study empirically in Section 2.1.2.

Gradients for Gaussian embeddings (G-EMBS). For G-EMBS, the gradients of Equation (2.4) with respect to each embedding and each context vector are

$$\nabla_{\rho_v} \mathcal{L} = -\lambda \rho_v + \frac{1}{\sigma^2} \sum_{i=1}^{T} \big(x_{iv} - \rho_v^\top \sum_{w \in c_v} x_{iw} \alpha_w\big)\big(\sum_{w \in c_v} x_{iw} \alpha_w\big), \tag{2.7}$$

$$\nabla_{\alpha_v} \mathcal{L} = -\lambda \alpha_v + \frac{1}{\sigma^2} \sum_{i=1}^{T} \sum_{w | v \in c_w} \big(x_{iw} - \rho_w^\top \sum_{r \in c_w} x_{ir} \alpha_r\big)\big(x_{iv} \rho_w\big). \tag{2.8}$$

Gradients for nonnegative Gaussian embeddings (NG-EMBS). By restricting the parameters of the NG-EMB model to be nonnegative we can learn nonnegative synaptic weights between neurons. For notational simplicity we write the parameters as $\exp(\rho)$ and $\exp(\alpha)$ and update them in log-space. The operator $\circ$ stands for element wise multiplication. With this notation, the gradient for the NG-EMB can be easily obtained from Equations 2.7 and 2.8 by applying the chain rule.

$$\begin{aligned} \nabla_{\rho_v} \mathcal{L} = &- \lambda \exp(\rho_v) \circ \exp(\rho_v) \\ &+ \frac{1}{\sigma^2} \sum_{i=1}^{I} \big(x_{iv} - \exp(\rho_v)^\top \sum_{w \in c_v} x_{iw} \exp(\alpha_w)\big)\big(\sum_{w \in c_v} x_{iw} \exp(\rho_v) \circ \exp(\alpha_w)\big) \end{aligned} \tag{2.9}$$

$$\begin{aligned} \nabla_{\alpha_v} \mathcal{L} = &- \lambda \exp(\alpha_v) \circ \exp(\alpha_v) \\ &+ \frac{1}{\sigma^2} \sum_{i=1}^{I} \sum_{w | v \in c_w} \big(x_{iw} - \exp(\rho_w)^\top \sum_{r \in c_w} x_{ir} \exp(\alpha_r)\big)\big(x_{iv} \exp(\rho_w) \circ \exp(\alpha_v)\big) \end{aligned} \tag{2.10}$$

Gradients for Poisson embeddings (P-EMBS). For P-EMBS, the gradients of Equation (2.4) with respect to each embedding and each context vector are

$$\nabla_{\rho_v} \mathcal{L} = -\lambda \rho_v + \sum_{i=1}^{I} \left(x_{iv} - \exp\left(\rho_v^\top \sum_{w \in c_{iv}} x_{iw} \alpha_w \right) \right) \left(\sum_{w \in c_{iv}} x_{iw} \alpha_w \right) \tag{2.11}$$

$$\nabla_{\alpha_v} \mathcal{L} = -\lambda \alpha_v + \sum_{i=1}^{I} \sum_{w | v \in c_{iv}} \left(x_{iw} - \exp\left(\rho_w^\top \sum_{r \in c_{iw}} x_{ir} \alpha_r \right) \right) (x_{iv} \rho_w) \tag{2.12}$$

Gradients for additive Poisson embeddings (AP-EMBS). Here, we proceed in a similar manner as for the NG-EMB model and apply the chain rule to the P-EMB gradients.

$$\nabla_{\rho_v} \mathcal{L} = -\lambda \exp(\rho_v) \circ \exp(\rho_v) + \sum_{i=1}^{I} \left(\frac{x_{iv}}{\rho_v^\top \sum_{w \in c_{iv}} x_{iw} \alpha_w} - 1 \right) \left(\sum_{w \in c_{iv}} x_{iw} \alpha_w \right) \tag{2.13}$$

$$\nabla_{\alpha_v} \mathcal{L} = -\lambda \exp(\alpha_v) \circ \exp(\alpha_v) + \sum_{i=1}^{I} \sum_{w | v \in c_{iw}} \left(\frac{x_{iw}}{\rho_w^\top \sum_{r \in c_{iw}} x_{ir} \alpha_r} - 1 \right) (x_{iv} \rho_w) \tag{2.14}$$

Chapter 3

Structured Embeddings

This chapter is about infusing the embeddings learned by a EF-EMB with more information. With the embeddings as latent variables, we can explicitly impose additional structure in the embedding space. The additional structure we study here is defined a priori and can be domain knowledge or discrete meta-data that is available with the data. Most of the structured embedding models we present below, are extensions of Bernoulli embeddings (B-EMBS) and we use them to analyze text data. Yet, the principles apply to all EF-EMBS.

The modeling extensions we develop incorporate additional information into an embedding model. For each object we want to learn an embedding of (e.g. word) we have additional information such as the time, information about the speaker, syntactic information (part-of-speech), or which language the word belongs to. The goal for structured embedding models is to exploit the structure of such additional information to learn embeddings that can vary according to the meta-data but also share information.

With dynamic embeddings, for example, we learn embeddings that vary over the years. Each time slice in the data is associated with a separate set of embeddings. Key to the success of the method is that the embedding parameters share statistical strength across time slices. We exploit the temporal ordering of the data and place a Gaussian random walk prior on the embeddings, which smoothes the trajectory of the embeddings over the years.

For more general group structure, the temporal ordering cannot be exploited and we instead use hierarchical priors to share statistical strength between the embeddings. In Section 3.2 we develop hierarchical embeddings, an embedding model for grouped data.

We then transition from more classical sharing strategies (using priors) to sharing strategies based on neural networks. For example, we develop amortized embeddings. Rather than modeling the relationship between group-specific embedding parameters through a hierarchical prior, the amortized embeddings use a neural network for each group that explicitly learns the mapping from global parameters to group-specific embeddings.

Another neural network based model we present in Section 3.3 is *word2net*. It is a deep extension of exponential family embeddings where the link function in Equation (2.3) is replaced with a multi layer perceptron. As a result, we end up learning a neural network for each word rather than an embedding vector. These neural networks that represent each word are called the *word networks*. We demonstrate that the word networks also provide an opportunity to incorporate side-information. Specifically, we use part-of-speech information to share parameter layers between word networks with the same part-of-speech tag. While this reduces the number of parameters, it increases the representational capacity, since under the parameter sharing scheme each word tag combination has a separate representation.

Finally, in Section 3.4, we present multilingual hierarchical embeddings (MAYHEM) a model for multilingual embeddings. Here, hierarchical priors are placed on the embeddings to tie together words that are known to be translations of each other. As a result, we can learn embeddings of words in 59 languages in one joint embedding space, with low-resource languages benefiting from the embedding structure of languages that have more resources, such as English.

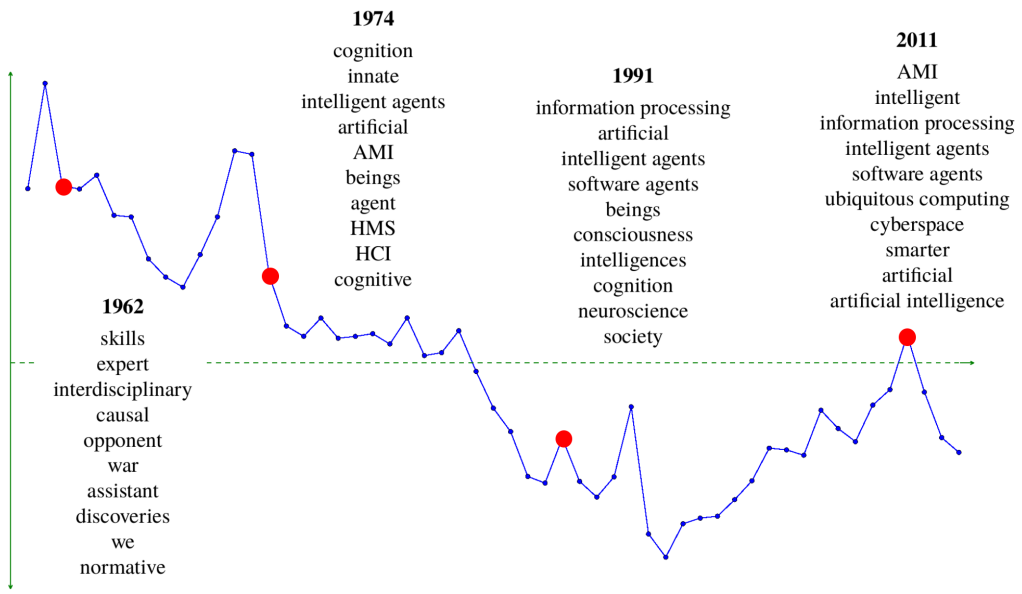
3.1 Dynamic Embeddings

Here we develop a tool for analyzing how word usage changes over time. Dynamic embeddings analyze long-running texts, e.g., documents that span many years, where the way words are used changes over the course of the collection. The goal of dynamic embeddings is to characterize those changes.

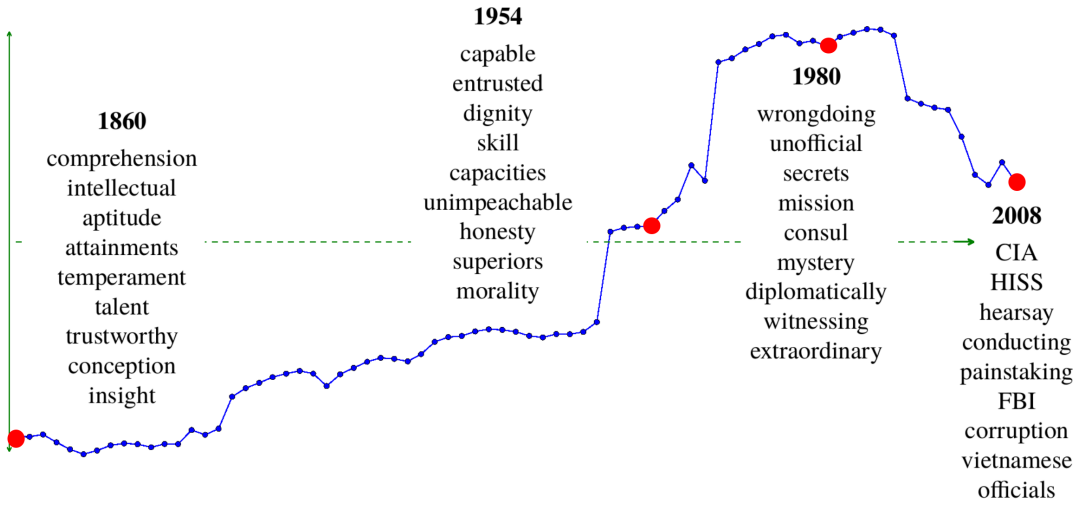
Figure 3.1 illustrates the approach. It shows the changing representation of INTELLIGENCE in two corpora, the collection of computer science abstracts from the ACM 1951–2014 and the U.S. Senate speeches 1858–2009. On the y-axis is "meaning," a proxy for the dynamic representation of the word; in both corpora, its representation changes dramatically over the years. To understand where it is located, the plots also show similar words (according to their changing representations) at various points. Loosely, in the ACM corpus INTELLIGENCE changes from government intelligence to cognitive intelligence to artificial intelligence; in the Congressional record INTELLIGENCE changes from psychological intelligence to government intelligence. Section 3.1.2 gives other examples from these corpora, such as for the terms IRAQ, DATA, and COMPUTER.

In more detail, a word embedding uses representation vectors to parameterize the conditional probabilities of words in the context of other words. Dynamic embeddings divide the documents into time slices, e.g., one per year, and cast the embedding vector as a latent variable that drifts via a Gaussian random walk. When fit to data, the dynamic embeddings capture how the representation of each word drifts from slice to slice.

Section 3.1.1 describes dynamic embeddings and how to fit them. Section 3.1.2 studies this approach on three datasets: 9 years of ArXiv machine learning papers (2007–2015), 64



(a) INTELLIGENCE in ACM abstracts (1951–2014)



(b) INTELLIGENCE in U.S. Senate speeches (1858–2009)

Figure 3.1: The dynamic embedding of INTELLIGENCE reveals how the term's usage changes over the years in a historic corpus of ACM abstracts (a) and U.S. Senate speeches (b) . The y-axis is "meaning," a one dimensional projection of the embedding vectors. For selected years, we list words with similar dynamic embeddings.

years of computer science abstracts (1951–2014), and 151 years of U.S. Senate speeches (1858–2009). Dynamic embeddings give better predictive performance than existing approaches and provide an interesting exploratory window into how language changes.

There have been several lines of research around capturing semantic shifts. Mihalcea and Nastase, (2012) and Tang, Qu, and Chen, (2016) detect semantic changes of words using features such as part-of-speech tags and entropy. Sagi, Kaufmann, and Clark, (2011) and Basile, Caputo, and Semeraro, (2014) employ latent semantic analysis and temporal semantic indexing for quantifying changes in meaning.

Most closely related to our work are methods for dynamic embeddings (Kim et al., 2014; Kulkarni et al., 2015; Hamilton, Leskovec, and Jurafsky, 2016b). These methods train a separate embedding for each time slice of the data. While interesting, this requires enough data in each time slice such that a high quality embedding can be trained for each. Further, because each time slice is trained independently, the dimensions of the embeddings are not comparable across time; they must use initialization (Kim et al., 2014) or ad-hoc alignment techniques (Kulkarni et al., 2015; Hamilton, Leskovec, and Jurafsky, 2016b; Zhang et al., 2016) to stitch them together.

In contrast, the representations of our model for dynamic embeddings are sequential latent variables. This naturally accommodates time slices with sparse data and assures that the dimensions of the embeddings are connected across time. In Section 3.1.2, we show that our method provides improvements over methods that fit each slice independently.

We note that two models similar to ours have been developed independently (Bamler and Mandt, 2017; Yao et al., 2017). Bamler and Mandt, (2017) model both the embeddings and the context vectors using an Uhlenbeck-Ornstein process (Uhlenbeck and Ornstein, 1930). Yao et al., (2017) factorize the pointwise mutual information (PMI) matrix at different time slices. Their regularization also resembles an Uhlenbeck-Ornstein process. Both employ the matrix factorization perspective of embeddings (Levy and Goldberg, 2014), while our work

builds on exponential family embeddings Chapter 2, which generalize embeddings using exponential families. A related perspective is given by Cotterell et al. (Cotterell et al., 2017) who show that exponential family PCA can generalize embeddings to higher order tensors.

Another area of related work is dynamic topic models, which are also used to analyze text data over time (Blei and Lafferty, 2006; Wang and McCallum, 2006; Wang, Blei, and Heckerman, 2008; Gerrish and Blei, 2010; Wijaya and Yeniterzi, 2011; Yogatama et al., 2014; Mitra et al., 2014; Mitra et al., 2015; Frermann and Lapata, 2016). This class of models describes documents in terms of topics, which are distributions over the vocabulary, and then allows the topics to change. As in dynamic embeddings, some dynamic topic models use a Gaussian random walk to capture drift in the underlying language model; for example, see Blei and Lafferty, (2006), Wang, Blei, and Heckerman, (2008), Gerrish and Blei, (2010), and Frermann and Lapata, (2016).

Though topic models and word embeddings are related, they are ultimately different approaches to language analysis. Topic models capture co-occurrence of words at the document level and focus on heterogeneity, i.e., that a document can exhibit multiple topics (Blei, Ng, and Jordan, 2003). Word embeddings capture co-occurrence in terms of proximity in the text, usually focusing on small neighborhoods around each word (Mikolov, Yih, and Zweig, 2013). Combining topic models and word embeddings is an area for future study.

3.1.1 Model Description of Dynamic Embeddings

D-EMBS extend B-EMBS to text data over time. Each observation x_{iv} is associated with a time feature t_i, such as the year of the observation. Context vectors are shared across all positions

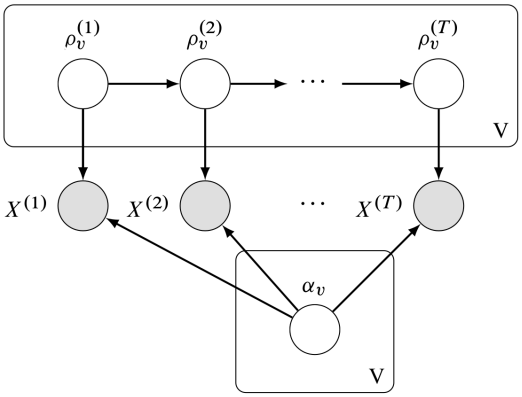


Figure 3.2: Graphical representation of a dynamic embedding (D-EMB) for text data in T time slices, $X^{(1)}, \cdots, X^{(T)}$. The embedding vectors ρ_v of each term evolve over time. The context vectors are shared across all time slices.

in the text but there is a set of embedding vectors per time slice. Dynamic embeddings posit a sequence of embeddings for each term $\rho_v^{(t)} \in \mathbb{R}^K$ while the static context vectors $\alpha_v \in \mathbb{R}^K$ help ensure that consecutive embeddings are grounded in the same semantic space.

The conditional probabilities are defined similar as in a B-EMB (Equation (1.1)) but with the embedding vector ρ_v replaced by the per-time-slice embedding vector $\rho_v^{(t_i)}$,

$$p(x_{iv} \mid \boldsymbol{x}_{c_i}) = \text{Bernoulli}\big(\sigma(\rho_v^{(t_i)\top} \Sigma_i)\big), \quad \text{with} \quad \Sigma_i \triangleq \sum_{(j,w)\in c_i} \alpha_w x_{jw}. \tag{3.1}$$

Dynamic embeddings use a Gaussian random walk as a prior on the embedding vectors,

$$\alpha_v, \rho_v^{(0)} \sim \mathcal{N}(0, \lambda_0^{-1} I), \quad \rho_v^{(t)} \sim \mathcal{N}(\rho_v^{(t-1)}, \lambda^{-1} I). \tag{3.2}$$

Given data, this leads to smoothly changing estimates of each term's embedding.[1]

[1]Because $\boldsymbol{\alpha}$ and $\boldsymbol{\rho}$ appear only as inner products in Equation (3.1), we can capture that their interactions change over time even by placing temporal dynamics on the embeddings $\boldsymbol{\rho}$ only. Exploring dynamics in $\boldsymbol{\alpha}$ is a subject for future study.

Figure 3.2 gives the graphical model for dynamic embeddings. Dynamic embeddings are a conditionally specified model, which in general are not guaranteed to imply a consistent joint distribution. But dynamic Bernoulli embeddings model binary data, and thus a joint exists (Arnold, Castillo, Sarabia, et al., 2001).

Fitting dynamic embeddings. Calculating the joint is computationally intractable. Like with Bernoulli embeddings (Chapter 1), we rather fit dynamic embeddings with the *pseudo log likelihood*, the sum of the log conditionals, a commonly used objective for conditional models (Arnold, Castillo, Sarabia, et al., 2001).

The objective $\mathcal{L}(\boldsymbol{\rho}, \boldsymbol{\alpha})$ has the same form as the objective of Bernoulli embeddings (Equation (1.2)). We regularize the pseudo log likelihood with the log priors and then maximize to obtain a pseudo MAP estimate. The priors (Equation (3.21)) contribute the following expressions to the loss

$$\log p(\boldsymbol{\alpha}) = -\frac{\lambda_0}{2} \sum_v ||\alpha_v||^2 \quad \log p(\boldsymbol{\rho}) = -\frac{\lambda_0}{2} \sum_v ||\rho_v^{(0)}||^2 - \frac{\lambda}{2} \sum_{v,t} ||\rho_v^{(t)} - \rho_v^{(t-1)}||^2.$$

The parameters $\boldsymbol{\rho}$ and $\boldsymbol{\alpha}$ appear in the logit parameters of the log conditionals (Equation (3.1)) and in the log prior. The log conditionals encourage the embeddings to capture the co-occurrences of the words while the random walk prior penalizes consecutive word vectors $\rho_v^{(t-1)}$ and $\rho_v^{(t)}$ for drifting too far apart. It prioritizes parameter settings for which the norm of their difference is small.

We fit the objective using stochastic gradients (Robbins and Monro, 1951) and with adaptive learning rates (Duchi, Hazan, and Singer, 2011b). The negative samples are

	ArXiv ML 2007 – 2015	ACM 1951 – 2014	Senate speeches 1858 – 2009
slices	9	64	76
slice size	1 year	1 year	2 years
vocab size	50k	25k	25k
words	6.5M	21.6M	13.7M

Table 3.1: Time range and size of the three corpora analyzed in Section 3.1.2.

resampled at each gradient step in the same way as in Chapter 1.[2]

3.1.2 Empirical Study of Dynamic Embeddings

This empirical study has two parts. In a quantitative evaluation we benchmark dynamic embeddings against static embeddings (Mikolov et al., 2013b; Mikolov et al., 2013a; Rudolph et al., 2016). We found that dynamic embeddings improve over static embeddings in terms of the conditional likelihood of held-out predictions. Further, dynamic embeddings perform better than embeddings trained on the individual time slices (Hamilton, Leskovec, and Jurafsky, 2016b). In a qualitative evaluation we use fitted dynamic embeddings to extract which word vectors change most and we visualize their dynamics. Dynamic embeddings provide a new window into how language changes.

We study three datasets. Their details are summarized in Table 3.8.

Machine Learning Papers (2007 - 2015) This dataset contains the full text from all machine learning papers (tagged "stat.ML") published on the ArXiv between April 2007 and June 2015. It spans 9 years and we treat each year as a time slice. The number of ArXiv papers about machine learning has increased over the years. There were 101 papers in 2007, while there were $1,573$ papers in 2014.

[2]Code is available at http://github.com/mariru/dynamic_bernoulli_embeddings

Computer Science Abstracts (1951 - 2014) This dataset contains abstracts of computer science papers published by the Association of Computing Machinery (ACM) from 1951 to 2014. We treat each year as a time slice and here too, the amount of data increases over the years. For 1953, there are only around 10 abstracts and their combined length is only 471 words; the total length of the abstracts from 2009 is over 2M.

Senate Speeches (1858 - 2009) This dataset contains all U.S. Senate speeches from 1858 to mid 2009. Here we treat every 2 years as a time slice. Unlike the other datasets, this is a transcript of spoken language. It contains many infrequent words that occur only in a few of the time slices.

Preprocessing We convert the text to lowercase and strip it of all punctuation. Frequent n-grams such as UNITED STATES are treated as a single term. The vocabulary consists of the $25,000$ most frequent terms and all words which are not in the vocabulary are removed. As in Mikolov et al., (2013a), we additionally remove each word with probability $p = 1 - \sqrt{(\frac{10^{-5}}{f_i})}$ where f_i is the frequency of the word. This effectively downsamples especially the frequent words and speeds up training.

Quantitative evaluation We compare dynamic embeddings (D-EMBS) to time-binned embeddings (T-EMBS) (Hamilton, Leskovec, and Jurafsky, 2016b) and static embeddings (S-EMBS). There are many embedding techniques, without dynamics, that enjoy comparable performance. For the S-EMB, we study Bernoulli embeddings (Chapter 1), which are similar to CBOW with negative sampling (Mikolov et al., 2013b; Mikolov et al., 2013a). For time-binned embeddings, Hamilton, Leskovec, and Jurafsky, (2016b) train a separate embedding on each time slice.

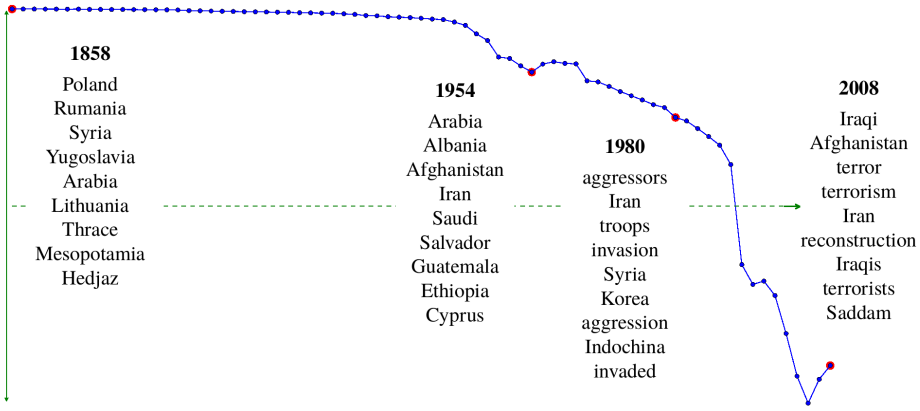


Figure 3.3: The dynamic embedding captures how the usage of the word IRAQ changes over the years (1858-2009). The x-axis is time and the y-axis is a one-dimensional projection of the embeddings using principal component analysis (PCA). We include the embedding neighborhoods for IRAQ in the years 1858, 1954, 1980 and 2008.

Evaluation metric. From each time slice 80% of the words are used for training. A random subsample of 10% of the words is held out for validation and another 10% for testing. We evaluate models by held-out Bernoulli probability. Given a model, each held-out position (validation or testing) is associated with a Bernoulli probability for each vocabulary term. At that position, a better model assigns higher probability to the observed word and lower probability to the others. This metric is straightforward because the competing methods all produce Bernoulli conditional likelihoods (Equation (1.1)). Since we hold out chunks of consecutive words usually both a word and its context are held out. All methods require the words in the context to compute the conditional likelihoods.

We report $\mathcal{L}_{\text{eval}} = \mathcal{L}_{\text{pos}} + \frac{1}{n}\mathcal{L}_{\text{neg}}$, where n is the number of negative samples. Renormalizing with n assures that the metric is balanced. It equally weights the positive and negative examples. To make results comparable, all methods are trained with the same n.

Model training and hyperparameters. Each method takes a maximum of 10 passes over the data. (The corresponding number of stochastic gradient steps depends on the size

of the minibatches.) The parameters of S-EMB are initialized randomly. We initialize both D-EMB and T-EMB from a fit of S-EMB which has been trained from one pass, and then train for 9 additional passes. For all methods, we set the dimension of the embeddings to 100 and the number of negative samples to 20. We study two context sizes, 2 and 8.

Other parameters are set by validation error. All methods use validation error to set the initial learning rate η and minibatch sizes m. The model selects $\eta \in [0.01, 0.1, 1, 10]$ and $m \in [0.001N, 0.0001N, 0.00001N]$, where N is the size of training data. The only parameter specific to D-EMB is the precision of the random drift. To have one less hyper parameter to tune, we fix the precision on the context vectors and the initial dynamic embeddings to $\lambda_0 = \lambda/1000$, a constant multiple of the precision on the dynamic embeddings. We choose $\lambda \in [1, 10]$ by validation error.

Results. We train each model on each training set and use each validation set for selecting parameters like the minibatch size and the learning rate. Table 3.9 reports the results on the test set. Dynamic embeddings consistently have higher held-out likelihood.

Qualitative exploration There are different reasons for a word's usage to change over the course of a collection. Words can become obsolete or obtain a new meaning. As society makes progress and words are used to describe that progress, that progress also gradually changes the meaning of words. A word might also have multiple alternative meanings. Over time, one meaning might become more relevant than the other. We now show how to use dynamic embeddings to explore text data and to discover such changes in the usage of words.

A word's *embedding neighborhood* helps visualize its usage and how it changes over time. It is simply a list of other words with similar usage. For a given query word (e.g.,

ArXiv ML

	context size 2	context size 8
S-EMB [Chapter 1]	−2.77	−2.54
T-EMB [Hamilton, Leskovec, and Jurafsky, 2016b]	−2.97	−2.81
D-EMB [this work]	**−2.58**	**−2.44**

Senate speeches

	context size 2	context size 8
S-EMB [Chapter 1]	−2.41	−2.29
T-EMB [Hamilton, Leskovec, and Jurafsky, 2016b]	−2.44	−2.46
D-EMB [this work]	**−2.33**	**−2.28**

ACM

	context size 2	context size 8
S-EMB [Chapter 1]	−2.48	−2.30
T-EMB [Hamilton, Leskovec, and Jurafsky, 2016b]	−2.55	−2.42
D-EMB [this work]	**−2.45**	**−2.27**

Table 3.2: Dynamic embedding (D-EMB) consistently achieve highest held-out $\mathcal{L}_{\text{eval}}$. We compare to static embedding (S-EMB) (Mikolov et al., 2013a), and time-binned embedding (T-EMB) (Hamilton, Leskovec, and Jurafsky, 2016b). The largest standard error on the held-out predictions is 0.002 which means all reported results are significant.

COMPUTER) we take its index v and select the top ten words according to

$$\text{neighborhood}(v, t) = \text{argsort}_w \left(\frac{\text{sign}(\rho_v^{(t)})^\top \rho_w^{(t)}}{||\rho_v^{(t)}|| \cdot ||\rho_w^{(t)}||} \right). \tag{3.3}$$

As an example, we fit a dynamic embedding fit to the Senate speeches. Table 3.3 gives the embedding neighborhoods of COMPUTER for the years 1858 and 1986. Its usage changed dramatically over the years. In 1858, a COMPUTER was a profession, a person who was hired to compute things. Now the profession is obsolete; COMPUTER refers to the electronic device. Another example in Table 3.3 is the word DATA, from a D-EMB of the ACM abstracts. The evolution of the embedding neighborhoods of DATA reflects how it changes meaning.

COMPUTER (Senate)		DATA (ACM)			
1858	**1986**	**1961**	**1969**	**1991**	**2014**
draftsman	software	directories	repositories	voluminous	data streams
draftsmen	computers	files	voluminous	raw data	voluminous
copyist	copyright	bibliographic	lineage	repositories	raw data
photographer	technological	formatted	metadata	data streams	warehouses
computers	innovation	retrieval	snapshots	data sources	dws
copyists	mechanical	publishing	data streams	volumes	repositories
janitor	hardware	archival	raw data	dws	data sources
accountant	technologies	archives	cleansing	dsms	data mining

Table 3.3: Dynamic embeddings reveal how word usage changes. In the Senate speeches from 1858, the word COMPUTER used to describe a profession. By 1986, the digital computer has been invented and the profession of people who compute things has become obsolete. The most similar words to the word DATA over the years, according to a dynamic embedding fitted to abstracts of the ACM journal, also attest to technological development.

Finding changing words with absolute drift. We have highlighted example words whose usage changes. However, not all words have changing usage. We now define a metric to discover which words change most.

words with largest drift (Senate)			
IRAQ	3.09	coin	2.39
tax cuts	2.84	social security	2.38
health care	2.62	FINE	2.38
energy	2.55	signal	2.38
medicare	2.55	program	2.36
DISCIPLINE	2.44	moves	2.35
text	2.41	credit	2.34
VALUES	2.40	UNEMPLOYMENT	2.34

Table 3.4: A list of the top 16 words whose dynamic embedding on Senate speeches changes most. The number represents the absolute drift (Equation (3.4)). The dynamics of the capitalized words are in Table 3.5 and discussed in the text.

One way to find words that change is to use *absolute drift*. For word v, it is

$$\text{drift}(v) = ||\rho_v^{(T)} - \rho_v^{(0)}||. \tag{3.4}$$

DISCIPLINE		VALUES		FINE		UNEMPLOYMENT	
1858	**2004**	**1858**	**2000**	**1858**	**2004**	**1858**	**2000**
hazing	balanced	fluctuations	sacred	luxurious	punished	unemployed	jobless
westpoint	balancing	value	inalienable	finest	penitentiaries	depression	rate
assaulting	fiscal	currencies	unique	coarse	imprisonment	acute	depression

Table 3.5: Embedding neighborhoods extracted from a dynamic embedding fitted to Senate speeches (1858 - 2009). DISCIPLINE, VALUES, FINE, and UNEMPLOYMENT are within the 16 words whose dynamic embedding has largest absolute drift. (Table 3.4).

JOBS			PROSTITUTION				
1858	**1938**	**2008**	**1858**	**1945**	**1962**	**1988**	**1990**
employment	unemployed	job	punishing	indecent	indecent	intimidation	servitude
unemployed	employment	create	immoral	vile	harassment	prostitution	harassment
overtime	job	creating	illegitimate	immoral	intimidation	counterfeit	intimidation

Table 3.6: Using dynamic embeddings we can study a social phenomenon of interest. We pick a target word of interest, such as JOBS or PROSTITUTION and create their embedding neighborhoods (Equation (3.3)).

This is the Euclidean distance between the word's embedding at the last and at the first step.

In the Senate speeches, Table 3.4 shows the 16 words that have largest absolute drift. The word IRAQ has largest drift. Figure 3.3 highlights IRAQ's embedding neighborhood in four time slices: 1858, 1950, 1980, and 2008. At first the neighborhood contains other countries and regions. Later, Arab countries move to the top of the neighborhood, suggesting that the speeches start to use rhetoric more specific to Arab countries. In 1980, Iraq invades Iran and the Iran-Iraq war begins. In these years, words such as TROOPS, and INVASION appear in the embedding neighborhood. By 2008, the neighborhood contains TERROR, and TERRORISM.

Four other words with large drift are DISCIPLINE, VALUES, FINE and UNEMPLOYMENT (Table 3.4). Table 3.5 shows their embedding neighborhoods. Of these words, DISCIPLINE, VALUES and, FINE have multiple meanings. Their neighborhoods reflect how the dominant meaning changes over time. For example, VALUES can be either a numerical quantity or can be used to refer to moral values and principles. In contrast, IRAQ and UNEMPLOYMENT are both words which have always had the same definition. Yet, the evolution of their

neighborhood captures changes in the way they are used.

Changepoint analysis. We use the fitted dynamic embeddings to find instances in time where a word's usage changes drastically. We make no assumption that a word's meaning makes only a single phase transition (Kulkarni et al., 2015). Since in our formulation of D-EMB the context vectors are shared between all time slices, the embeddings are grounded in one semantic space and no postprocessing is needed to align the embeddings. We can directly compute large jumps in word usage on the learned embedding vectors.

For each word, we compute a list of time slices where the word's usage changed most.

$$\text{max change}(v) = \text{argsort}_t \left(\frac{||\rho_v^{(t)} - \rho_v^{(t-1)}||}{\sum_w ||\rho_w^{(t)} - \rho_w^{(t-1)}||} \right). \tag{3.5}$$

The changes in time slice t are normalized by how much all other words changed within the same time slice. The normalization, makes the *max change* ranking sensitive to time slices in which a word's embedding drifted farthest, compared to how far other words drifted within the time slice.

For example, for the word IRAQ the largest change is in the years 1990-1992. Indeed, that year the Gulf war started. Note that this is consistent with Figure 3.3 where we see in the one dimensional projection of the trajectory of the embedding a large jump around the year 1990. The trajectory in the Figure captures only the variation in the first principal component, while Equation 3.5 measures difference of embedding vectors in all the dimensions combined.

Next, we examine in which years many words changed most in terms of their usage. In Figure 3.4 is a histogram of the years in which each word changed the most. For example, IRAQ falls into the 1990-1992 bin, together with almost 300 other words which also had

their largest relative change in 1990 - 1992. We can see that the bin with the most words (marked in red) is 1946-1947 which marks the end of the Second World War. Almost 1000 words had their largest relative change in that time slice.

In Table 3.7 is a list of the 10 words with the largest change in the years 1946-1947. On top of the list is MARSHALL, the middle name of John Marshall Harlan, and John Marshall Harlan II, father and son who both served as U.S. Supreme Court Justices. It is also the last name of George Marshall who became the U.S. Secretary of State in 1947. He conceived and carried out the Marshall plan, an economic relief program to aid post-war Europe.

In Table 3.7 are the embedding neighborhoods for MARSHALL before and after the 1946-1947 time bin. In 1944-1945, MARSHALL is similar to other names with importance to the U.S. judicial system but by 1948-1950 the most similar word is PLAN as the Marshall plan is now frequently discussed in the U.S. Senate Speeches.

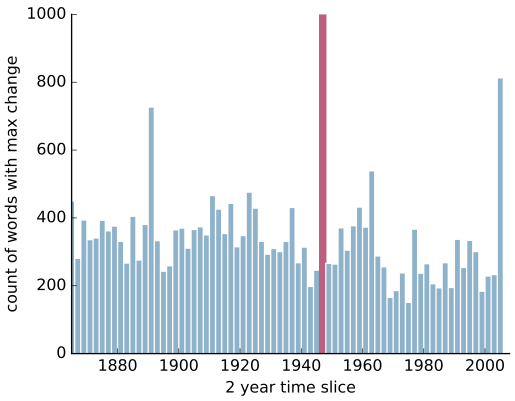


Figure 3.4: According to D-EMB fitted to the Senate Speeches, most words change most in the 1947-1947 time slice. Table 3.7 lists the 10 words that change most during that time.

Table 3.7: D-EMB identifies MARSHALL, as the word changing most in 1946-1947. On the left is a list of the top words with largest change in the time bin marked red in Figure 3.4. On the right, are the embedding neighborhoods of MARSHALL before and after the jump.

top change in 1946	**MARSHALL (Senate)**	
1. marshall	**1944**	**1948**
2. atlantic	wheaton	plan
3. korea	taney	satellites
4. douglas	harlan	britain
5. holland	vs	great britain
6. steam	gibbons	acheson
7. security	mcreynolds	democracies
8. truman	waite	france
9. plan	webster	western europe

Dynamic embeddings as a tool to study a text. Our hope is that dynamic embeddings provide a suggestive tool for understanding change in language. For example, researchers interested in UNEMPLOYMENT can complement their investigation by looking at the embedding neighborhood of related words such as EMPLOYMENT, JOBS or LABOR. In Table 3.6 we list the neighborhoods of JOBS for the years 1858, 1938, and 2008. In 2008 the embedding neighborhood contains words like CREATE and CREATING, suggesting a different outlook on JOBS than in earlier years.

Another interesting example is PROSTITUTION. It used to be IMMORAL and VILE, went to INDECENT, and in modern days it is considered HARASSMENT. We note the word PROSTITUTION is not a frequent word. On average, it is used once per time slice and, in two thirds of the time slices, it is not mentioned at all. Yet, the model is able to learn about PROSTITUTION and the temporal evolution of the embedding neighborhood reveals how over the years a judgemental stance turns into concern over a social issue.

3.1.3 Discussion of Dynamic Embeddings

We described dynamic embeddings, distributed representations of words that drift over the course of the collection. Building on Chapter 1 , we formulate word embeddings with conditional probabilistic models and then incorporate dynamics with a Gaussian random walk prior. We fit dynamic embeddings to language data using stochastic optimization.

We used dynamic embeddings to analyze 3 datasets: 8 years of machine learning papers, 63 years of computer science abstracts, and 151 years of U.S. Senate speeches. Dynamic embeddings provide a better fit than static embeddings and other methods that account for time. In addition, dynamic embeddings can help identify interesting ways in which language changes. A word's meaning can change (e.g., COMPUTER); its dominant meaning can change (e.g., VALUES); or its related subject matter can change (e.g., IRAQ).

3.2 Hierarchical and Amortized Embeddings

We develop hierarchical and amortized embeddings, extensions of EF-EMB for studying how embeddings can vary across groups of related data. Since both hierarchical and amortized embeddings accommodate data divided into some group structure, we call both methods structured embedding (S-EMB). We will study several examples: in U.S. Congressional speeches, word usage can vary across states or party affiliations; in scientific literature, the usage patterns of technical terms can vary across fields; in supermarket shopping data, co-purchase patterns of items can vary across seasons of the year. We will see that S-EMB discovers a per-group embedding representation of objects. While the naïve approach of fitting an individual embedding model for each group would typically suffer from lack of

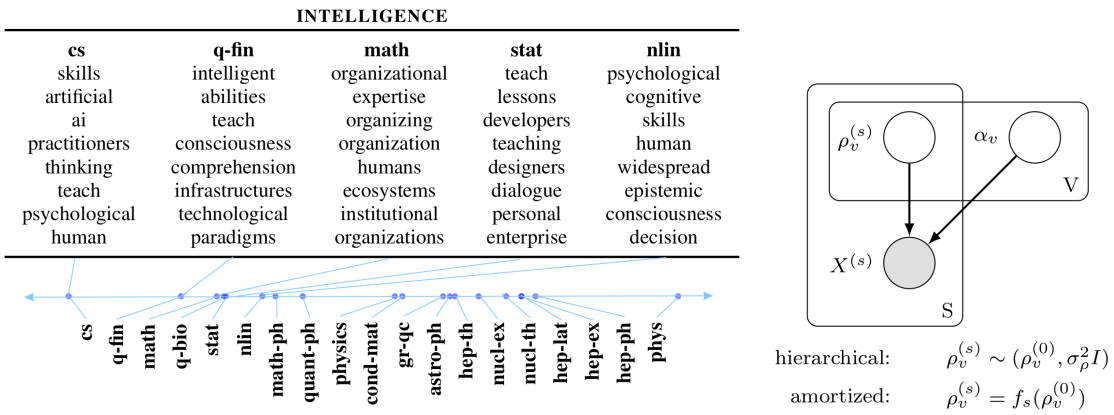
data—especially in groups for which fewer observations are available—we develop two methods that can share information across groups.

Figure 3.5(a) illustrates the kind of variation that we can capture. We fit an S-EMB to ArXiv abstracts grouped into different sections, such as computer science (**cs**), quantitative finance (**q-fin**), and nonlinear sciences (**nlin**). S-EMB results in a per-section embedding of each term in the vocabulary. Using the fitted embeddings, we illustrate similar words to the word INTELLIGENCE. We can see that how INTELLIGENCE is used varies by field: in computer science the most similar words include ARTIFICIAL and AI; in finance, similar words include ABILITIES and CONSCIOUSNESS.

Like the EF-EMB of Chapter 2, S-EMB maintains two types of representation vectors for each term in the vocabulary; embedding vectors and context vectors. (We use the language of text for concreteness; as we mentioned, EF-EMB extend to other types of data.) Each word has a separate embedding vector for each group, but the context vectors are shared; this ensures that the embedding vectors are in the same space.

We propose two methods to share statistical strength among the embedding vectors. The first approach is based on hierarchical modeling (Gelman et al., 2003), which assumes that the group-specific embedding representations are tied through a global embedding. The second approach is based on amortization (Dayan et al., 1995; Gershman and Goodman, 2014), which considers that the individual embeddings are the output of a deterministic function of a global embedding representation. Here too, we use stochastic optimization on a pseudo-likelihood objective to fit large data sets.

Our work relates closely to two threads of research in the embedding literature. One is embedding methods that study how language evolves over time (Kim et al., 2014; Kulkarni



(a) S-EMB uncovers variations in the usage of the word INTELLIGENCE. (b) Graphical repres. of S-EMB.

Figure 3.5: (a) INTELLIGENCE is used differently across the ArXiv sections. Words with the closest embedding to the query are listed for 5 sections. (The embeddings were obtained by fitting an amortized S-EMB.) The method automatically orders the sections along the horizontal axis by their similarity in the usage of INTELLIGENCE. See Section 3.2.2 for additional details. **(b)** Graphical representation of S-EMB for data in S categories. The embedding vectors $\rho_v^{(s)}$ are specific to each group, and the context vectors α_v are shared across all categories.

et al., 2015; Hamilton, Leskovec, and Jurafsky, 2016b; Rudolph and Blei, 2017; Bamler and Mandt, 2017; Yao et al., 2017). Time can be thought of as a type of "group", though with evolutionary structure that we do not consider here. For the D-EMB developed in Section 3.1, we were able to exploit the temporal ordering of the groups through a Gaussian random walk prior on the embeddings. Here we assume no knowledge on how the groups are related to each other. The second thread is multilingual embeddings (Klementiev, Titov, and Bhattarai, 2012; Mikolov, Le, and Sutskever, 2013; Ammar et al., 2016; Zou et al., 2013); our approach is different in that most words appear in all groups and we are interested in the variations of the embeddings across those groups. In Section 3.4, we will consider a hierarchical embedding model for multilingual data.

Our contributions are thus as follows. We introduce the S-EMB model, extending EF-EMB to grouped data. We present two techniques to share statistical strength among the embedding

	data	embedding of	groups	grouped by	size
ArXiv abstracts	text	15k terms	19	subject areas	15M words
Senate speeches	text	15k terms	83	home state/party	20M words
Shopping data	counts	5.5k items	12	months	0.5M trips

Table 3.8: Group structure and size of the three corpora analyzed in Section 3.2.2.

vectors, one based on hierarchical modeling and one based on amortization. We carry out an experimental study on two text databases, ArXiv papers by section and U.S. Congressional speeches by home state and political party. Using Poisson embeddings, we study market basket data from a large grocery store, grouped by season. On all three data sets, S-EMB outperforms EF-EMB in terms of held-out log-likelihood. Qualitatively, we demonstrate how S-EMB discovers which words are used most differently across U.S. states and political parties, and show how word usage changes in different ArXiv disciplines.

3.2.1 Model Description of Hierarchical and Amortized Embeddings

In this section, we develop S-EMB, a model that builds on EF-EMB (Rudolph et al., 2016) to capture semantic variations across groups of data. In embedding models, we represent each object (e.g., a word in text, or an item in shopping data) using two sets of vectors, an embedding vector and a context vector. We are interested in how the embeddings vary across groups of data, and for each object we want to learn a separate embedding vector for each group. Having a separate embedding for each group allows us to study how the usage of a word like INTELLIGENCE varies across categories of the ArXiv, or which words are used most differently by U.S. Senators depending on which state they are from and whether they are Democrats or Republicans.

Model. Here, we describe the S-EMB model for grouped data. In text, some examples

of grouped data are Congressional speeches grouped into political parties or scientific documents grouped by discipline. Our goal is to learn group-specific embeddings from data partitioned into S groups, i.e., each instance i is associated with a group $s_i \in \{1, \ldots, S\}$. The S-EMB model extends EF-EMB to learn a separate set of embedding vectors for each group. Each term v has a single context vector $\alpha_v \in \mathbb{R}^K$ and S embedding vectors, $\rho_v^{(s)} \in \mathbb{R}^K$, one for each group s. These parameters are combined in the conditional likelihoods of an EF-EMB (Equation (2.1)) with the embedding ρ_v in Equation (2.3) replaced by the group-specific embedding $\rho_v^{(s_i)}$. Wich group-specific embedding to use is determined by which group the observation belongs to. We show a graphical representation of the S-EMB in Figure 3.5(b).

Sharing the context vectors α_v has two advantages. First, the shared structure reduces the number of parameters, while the resulting S-EMB model is still flexible to capture how differently words are used across different groups, as $\rho_v^{(s)}$ is allowed to vary. Second, it has the important effect of uniting all embedding parameters in the same space, as the group-specific vectors $\rho_v^{(s)}$ need to agree with the components of α_v. While one could learn a separate embedding model for each group, as has been done for text grouped into time slices (Kim et al., 2014; Kulkarni et al., 2015; Hamilton, Leskovec, and Jurafsky, 2016b), this approach would require ad-hoc postprocessing steps to align the embeddings.[3]

When there are S groups, the S-EMB model has S times as many embedding vectors than the standard embedding model. This may complicate inferences about the group-specific vectors, especially for groups with less data. Additionally, an object v may appear with very low frequency in a particular group. Thus, the naïve approach for building the S-EMB model

[3]Another potential advantage of the proposed parameter sharing structure is that, when the context vectors are held fixed, the resulting objective function is convex, by the convexity properties of exponential families (Wainwright and Jordan, 2008).

without additional structure may be detrimental for the quality of the embeddings, especially for small-sized groups. To address this problem, we propose two different methods to tie the individual $\rho_v^{(s)}$ together, sharing statistical strength among them. The first approach consists in a hierarchical embedding structure. The second approach is based on amortization. In both methods, we introduce a set of *global* embedding vectors $\rho_v^{(0)}$, and impose a particular structure to generate $\rho_v^{(s)}$ from $\rho_v^{(0)}$.

Hierarchical embedding structure. Here, we impose a hierarchical structure that allows sharing statistical strength among the per-group variables. For that, we assume that each $\rho_v^{(s)} \sim \mathcal{N}(\rho_v^{(0)}, \sigma_\rho^2 I)$, where σ_ρ^2 is a fixed hyperparameter. The objective is again a pseudo-lieklihood,

$$\mathcal{L}_{\text{hier}} = \log p(\alpha) + \log p(\rho^{(0)}) + \sum_s \log p(\rho^{(s)} \mid \rho^{(0)}) + \sum_{v,i} \log p\left(x_{vi} \;\middle|\; x_{c_{vi}}; \alpha, \rho\right). \quad (3.6)$$

Fitting the hierarchical model involves maximizing Equation (3.6) with respect to α_v, $\rho_v^{(0)}$, and $\rho_v^{(s)}$. We note that we have not reduced the number of parameters to be inferred; rather, we tie them together through a common prior distribution. We use stochastic gradient ascent to maximize Equation (3.6).

Amortization. The idea of amortization has been applied in the literature to develop amortized inference algorithms (Dayan et al., 1995; Gershman and Goodman, 2014). The main insight behind amortization is to reuse inferences about past experiences when presented with a new task, leveraging the accumulated knowledge to quickly solve the new problem. Here, we use amortization to control the number of parameters of the S-EMB model. In particular, we set the per-group embeddings $\rho_v^{(s)}$ to be the output of a deterministic function

of the global embedding vectors, $\rho_v^{(s)} = f_s(\rho_v^{(0)})$. We use a different function $f_s(\cdot)$ for each group s, and we parameterize them using neural networks, similarly to other works on amortized inference (Korattikara et al., 2015; Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014; Mnih and Gregor, 2014). Unlike standard uses of amortized inference, in S-EMB the input to the functions $f_s(\cdot)$ is unobserved and must be estimated together with the parameters of the functions $f_s(\cdot)$.

Depending on the architecture of the neural networks, the amortization can significantly reduce the number of parameters in the model (as compared to the non-amortized model), while still having the flexibility to model different embedding vectors for each group. The number of parameters in the S-EMB model is $KL(S+1)$, where S is the number of groups, K is the dimensionality of the embedding vectors, and L is the number of objects (e.g., the vocabulary size). With amortization, we reduce the number of parameters to $2KL + SP$, where P is the number of parameters of the neural network. Since typically $L \gg P$, this corresponds to a significant reduction in the number of parameters, even when P scales linearly with K.

In the amortized S-EMB model, we need to introduce a new set of parameters $\phi^{(s)} \in \mathbb{R}^P$ for each group s, corresponding to the neural network parameters. Given these, the group-specific embedding vectors $\rho_v^{(s)}$ are obtained as

$$\rho_v^{(s)} = f_s(\rho_v^{(0)}) = f(\rho_v^{(0)}; \phi^{(s)}). \tag{3.7}$$

We compare two architectures for the function $f_s(\cdot)$: fully connected feed-forward neural networks and residual networks (He et al., 2016). For both, we consider one hidden layer

with H units. Hence, the network parameters $\phi^{(s)}$ are two weight matrices,

$$\phi^{(s)} = \{W_1^{(s)} \in \mathbb{R}^{H\times K}, W_2^{(s)} \in \mathbb{R}^{K\times H}\}, \tag{3.8}$$

i.e., $P = 2KH$ parameters. The neural network takes as input the global embedding vector $\rho_v^{(0)}$, and it outputs the group-specific embedding vectors $\rho_v^{(s)}$. The mathematical expression for $\rho_v^{(s)}$ for a feed-forward neural network and a residual network is respectively given by

$$\rho_v^{(s)} = f_{\text{ffnet}}(\rho_v^{(0)}; \phi^{(s)}) = W_2^{(s)} \tanh\left(W_1^{(s)}\rho_v^{(0)}\right), \tag{3.9}$$

$$\rho_v^{(s)} = f_{\text{resnet}}(\rho_v^{(0)}; \phi^{(s)}) = \rho_v^{(0)} + W_2^{(s)} \tanh\left(W_1^{(s)}\rho_v^{(0)}\right), \tag{3.10}$$

where we have considered the hyperbolic tangent nonlinearity. The main difference between both network architectures is that the residual network focuses on modeling how the group-specific embedding vectors $\rho_v^{(s)}$ differ from the global vectors $\rho_v^{(0)}$. That is, if all weights were set to 0, the feed-forward network would output 0, while the residual network would output the global vector $\rho_v^{(0)}$ for all groups.

The objective function under amortization is given by

$$\mathcal{L}_{\text{amortiz}} = \log p(\alpha) + \log p(\rho^{(0)}) + \sum_{v,i} \log p\left(x_{vi} \mid x_{c_{vi}}; \alpha, \rho^{(0)}, \phi\right). \tag{3.11}$$

We maximize this objective with respect to α_v, $\rho_v^{(0)}$, and $\phi^{(s)}$ using stochastic gradient ascent. We implement the hierarchical and amortized S-EMB models in TensorFlow (Abadi et al.,

2016), which allows us to leverage automatic differentiation.[4]

3.2.2 Empirical Study of Hierarchical and Amortized Embeddings

Data. We apply the S-EMB on three datasets: ArXiv papers, U.S. Senate speeches, and purchases on supermarket grocery shopping data. We describe these datasets below, and we provide a summary of the datasets in Table 3.8.

ArXiv papers: This dataset contains the abstracts of papers published on the ArXiv under the 19 different tags between April 2007 and June 2015. We treat each tag as a group and fit S-EMB with the goal of uncovering which words have the strongest shift in usage. We split the abstracts into training, validation, and test sets, with proportions of 80%, 10%, and 10%, respectively.

Senate speeches: This dataset contains U.S. Senate speeches from 1994 to mid 2009. In contrast to the ArXiv collection, it is a transcript of spoken language. We group the data into state of origin of the speaker and his or her party affiliation. Only affiliations with the Republican and Democratic Party are considered. As a result, there are 83 groups (Republicans from Alabama, Democrats from Alabama, Republicans from Arkansas, etc.). Some of the state/party combinations are not available in the data, as some of the 50 states have only had Senators with the same party affiliation. We split the speeches into training (80%), validation (10%), and testing (10%).

Grocery shopping data: This dataset contains the purchases of 3,206 customers. The data covers a period of 97 weeks. After removing low-frequency items, the data contains

[4]Code is available at https://github.com/mariru/structured_embeddings

5, 590 unique items at the UPC (Universal Product Code) level. We split the data into a training, test, and validation sets, with proportions of 90%, 5%, and 5%, respectively. The training data contains 515, 867 shopping trips and 5, 370, 623 purchases in total.

For the text corpora, we fix the vocabulary to the 15k most frequent terms and remove all words that are not in the vocabulary. Following Mikolov et al., (2013a), we additionally remove each word with probability $1 - \sqrt{10^{-5}/f_v}$, where f_v is the word frequency. This downsamples especially the frequent words and speeds up training. (Sizes reported in Table 3.8 are the number of words remaining after preprocessing.)

Models. Our goal is to fit the S-EMB model on these datasets. For the text data, we use the Bernoulli distribution as the conditional exponential family, while for the shopping data we use the Poisson distribution, which is more appropriate for count data.

On each dataset, we compare four approaches based on S-EMB with two EF-EMB (Rudolph et al., 2016) baselines. All are fit using SGD (Robbins and Monro, 1951). In particular, we compare the following methods:

- A global EF-EMB model, which cannot capture group structure.
- Separate EF-EMB models, fitted independently on each group.
- (this work) S-EMB without hierarchical structure or amortization.
- (this work) S-EMB with hierarchical group structure.
- (this work) S-EMB, amortized with a feed-forward neural network (Equation (3.9)).
- (this work) S-EMB, amortized using a residual network (Equation (3.10)).

Experimental setup and hyperparameters. For text we set the dimension of the embeddings to $K = 100$, the number of hidden units to $H = 25$, and we experiment with

two context sizes, 2 and 8.[5] In the shopping data, we use $K = 50$ and $H = 20$, and we randomly truncate the context of baskets larger than 20 to reduce their size to 20. For both methods, we use 20 negative samples.

For all methods, we subsample minibatches of data in the same manner. Each minibatch contains subsampled observations from all groups and each group is subsampled proportionally to its size. For text, the words subsampled from within a group are consecutive, and for shopping data the observations are sampled at the shopping trip level. This sampling scheme reduces the bias from imbalanced group sizes. For text, we use a minibatch size of $N/10000$, where N is the size of the corpus, and we run 5 passes over the data; for the shopping data we use $N/100$ and run 50 passes. We use the default learning rate setting of TensorFlow for Adam[6] (Kingma and Ba, 2015).

We use the standard initialization schemes for the neural network parameters. The weights are drawn from a uniform distribution bounded at $\pm\sqrt{6}/\sqrt{K+H}$ (Glorot and Bengio, 2010). For the embeddings, we try 3 initialization schemes and choose the best one based on validation error. In particular, these schemes are: (1) all embeddings are drawn from the Gaussian prior implied by the regularizer; (2) the embeddings are initialized from a global embedding; (3) the context vectors are initialized from a global embedding and held constant, while the embeddings vectors are drawn randomly from the prior. Finally, for each method we choose the regularization variance from the set $\{100, 10, 1, 0.1\}$, also based on validation error.

---

[5] To save space we report results for context size 8 only. Context size 2 shows the same relative performance.

[6] Adam needs to track a history of the gradients for each parameter that is being optimized. One advantage from reducing the number of parameters with amortization is that it results in a reduced computational overhead for Adam (as well as for other adaptive stepsize schedules).

Runtime. We implemented all methods in Tensorflow. On the Senate speeches, the runtime of S-EMB is 4.3 times slower than the runtime of global EF-EMB, hierarchical EF-EMB is 4.6 times slower than the runtime of global EF-EMB, and amortized S-EMB is 3.3 times slower than the runtime of global EF-EMB. (The Senate speeches have the most groups and hence the largest difference in runtime between methods.)

Evaluation metric. We evaluate the fits by held-out pseudo (log-)likelihood. For each model, we compute the test pseudo log-likelihood, according to the exponential family distribution used (Bernoulli or Poisson). For each test entry, a better model will assign higher probability to the observed word or item, and lower probability to the negative samples. This is a fair metric because the competing methods all produce conditional likelihoods from the same exponential family.[7] To make results comparable, we train and evaluate all methods with the same number of negative samples (20). The reported held out likelihoods give equal weight to the positive and negative samples.

Quantitative results. We show the test pseudo log-likelihood of all methods in Table 3.9 and report that our method outperforms the baseline in all experiments. We find that S-EMB with either hierarchical structure or amortization outperforms the competing methods based on standard EF-EMB highlighted in bold. This is because the global EF-EMB ignores per-group variations, whereas the separate EF-EMB cannot share information across groups. The results of the global EF-EMB baseline are better than fitting separate EF-EMB (the other baseline), but unlike the other methods the global EF-EMB cannot be used for the exploratory analysis of variations across groups. Our results show that using a hierarchical S-EMB is

[7] Since we hold out chunks of consecutive words usually both a word and its context are held out. For all methods we have to use the words in the context to compute the conditional likelihoods.

	ArXiv papers	Senate speeches	Shopping data
Global EF-EMB (Rudolph et al., 2016)	-2.176 ± 0.005	-2.239 ± 0.002	-0.772 ± 0.000
Separated EF-EMB (Rudolph et al., 2016)	-2.500 ± 0.012	-2.915 ± 0.004	-0.807 ± 0.002
S-EMB	-2.287 ± 0.007	-2.645 ± 0.002	-0.770 ± 0.001
S-EMB (hierarchical)	-2.170 ± 0.003	$\mathbf{-2.217 \pm 0.001}$	-0.767 ± 0.000
S-EMB (amortiz+feedf)	-2.153 ± 0.004	-2.484 ± 0.002	-0.774 ± 0.000
S-EMB (amortiz+resnet)	$\mathbf{-2.120 \pm 0.004}$	-2.249 ± 0.002	$\mathbf{-0.762 \pm 0.000}$

Table 3.9: Test log-likelihood on the three considered datasets. S-EMB consistently achieves the highest held-out likelihood. The competing methods are the global EF-EMB, which can not capture group variations, and the separate EF-EMB, which cannot share information across groups.

always better than using the simple S-EMB model or fitting a separate EF-EMB on each group. The hierarchical structure helps, especially for the Senate speeches, where the data is divided into many groups. Among the amortized S-EMB models we developed, at least amortization with residual networks outperforms the base S-EMB. The advantage of residual networks over feed-forward neural networks is consistent with the results reported by (He et al., 2016).

While both hierarchical S-EMB and amortized S-EMB share information about the embedding of a particular word across groups (through the global embedding $\rho_v^{(0)}$), amortization additionally ties the embeddings of all words within a group (through learning the neural network of that group). We hypothesize that for the Senate speeches, which are split into many groups, this is a strong modeling constraint, while it helps for all other experiments.

Structured embeddings reveal a spectrum of word usage. We have motivated S-EMB with the example that the usage of INTELLIGENCE varies by ArXiv category (Figure 3.5(a)). We now explain how for each term the per-group embeddings place the groups on a spectrum. For a specific term v we take its embeddings vectors $\{\rho_v^{(s)}\}$ for all groups s, and project them onto a one-dimensional space using the first component of PCA. This is a one-dimensional

summary of how close the embeddings of v are across groups. Such comparison is possible because the S-EMB shares the context vectors, which grounds the embedding vectors in a joint space.

The spectrum for the word INTELLIGENCE along its first principal component is the horizontal axis in Figure 3.5(a). The dots are the projections of the group-specific embeddings for that word. (The embeddings come from a fitted S-EMB with feed-forward amortization.) We can see that in an unsupervised manner, the method has placed together groups related to physics on one end on the spectrum, while computer science, statistics and math are on the other end of the spectrum.

To give additional intuition of what the usage of INTELLIGENCE is at different locations on the spectrum, we have listed the 8 most similar words for the groups computer science (**cs**), quantitative finance (**q-fin**), math (**math**), statistics (**stat**), and nonlinear sciences (**nlin**). Word similarities are computed using cosine distance in the embedding space. Even though their embeddings are relatively close to each other on the spectrum, the model has the flexibility to capture high variability in the lists of similar words.

Exploring group variations with structured embeddings. The result of the S-EMB also allows us to investigate which words have the highest deviation from their average usage for each group. For example, in the Congressional speeches, there are many terms that we do not expect the Senators to use differently (e.g., most stopwords). We might however want to ask a question like "which words do Republicans from Texas use most differently from other Senators?" By suggesting an answer, our method can guide an exploratory data analysis. For each group s (state/party combination), we compute the top 3 words in

TEXAS	FLORIDA		IOWA		WASHINGTON
border	medicaid	bankruptcy	agriculture	prescription	washington
our country	prescription	water	farmers	drug	energy
iraq	medicare	waste	food	drugs	oil

Table 3.10: List of the three most different words for different groups for the Congressional speeches. S-EMB uncovers which words are used most differently by Republican Senators (red) and Democratic Senators (blue) from different states. The complete table is in the Appendix.

$\text{argsort}_v \left(||\rho_v^{(s)} - \frac{1}{S} \sum_{t=1}^{S} \rho_v^{(t)}|| \right)$ from within the top 1k words.

Table 3.10 shows a summary of our findings (the full table is in the Appendix). According to the S-EMB (with residual network amortization), Republican Senators from Texas use BORDER and the phrase OUR COUNTRY in different contexts than other Senators.

Some of these variations are probably influenced by term frequency, as we expect Democrats from Washington to talk about WASHINGTON more frequently than other states. But we argue that our method provides more insights than a frequency based analysis, as it is also sensitive to the context in which a word appears. For example, WASHINGTON might in some groups be used more often in the context of PRESIDENT and GEORGE, while in others it might appear in the context of DC and CAPITAL, or it may refer to the state.

3.2.3 Discussion of Hierarchical and Amortized Embeddings

We have presented several structured extensions of EF-EMB for modeling grouped data. Hierarchical S-EMB can capture variations in word usage across groups while sharing statistical strength between them through a hierarchical prior. Amortization is an effective way to reduce the number of parameters in the hierarchical model. The amortized S-EMB model leverages the expressive power of neural networks to reduce the number of parameters,

while still having the flexibility to capture variations between the embeddings of each group. Below are practical guidelines for choosing a S-EMB.

How can I fit embeddings that vary across groups of data? To capture variations across groups, never fit a separate embedding model for each group. We recommend at least sharing the context vectors, as all the S-EMB models do. This ensures that the latent dimensions of the embeddings are aligned across groups. In addition to sharing context vectors, we also recommend sharing statistical strength between the embedding vectors. In this chapter we have presented two ways to do so, hierarchical modeling and amortization.

Should I use a hierarchical prior or amortization? The answer depends on how many groups the data contain. In our experiments, the hierarchical S-EMB works better when there are many groups. With less groups, the amortized S-EMB works better.

The advantage of the amortized S-EMB is that it has fewer parameters than the hierarchical model, while still having the flexibility to capture across-group variations. The global embeddings in an amortized S-EMB have two roles. They capture the semantic similarities of the words, and they also serve as the input into the amortization networks. Thus, the global embeddings of words with similar pattern of across-group variation need to be in regions of the embedding space that lead to similar modifications by the amortization network. As the number of groups in the data increases, these two roles become harder to balance. We hypothesize that this is why the amortized S-EMB has stronger performance when there are fewer groups.

Should I use feed-forward or residual networks? To amortize a S-EMB we recommend residual networks. They perform better than the feed-forward networks in all of our

experiments. While the feed-forward network has to output the entire meaning of a word in the group-specific embedding, the residual network only needs the capacity to model how the group-specific embedding differs from the global embedding.

3.3 Word2Net: Deep Embeddings

The goal in word2net is to learn a neural network representation for each word. Here we present a model for learning the word networks as well as a parameter sharing scheme that allows us to incorporate part-of-speech information into the task of learning distributed representation. We describe how to calculate semantic similarities between the fitted word networks and show that qualitatively the learned embeddings seem reasonable.

Our goal is to learn neural network representations of words, deep representations that capture each word's meaning. To this end, we develop word2net. Given text data, word2net learns a collection of *word networks*, one for each term of the vocabulary. These networks enjoy greater representational capacity than their classical vector counterparts.

Figure 3.7a illustrates the intuition behind word2net. Consider the term INCREASE. The top of the figure shows an observation of this term as well as its surrounding words. (This excerpt is from U.S. Senate speeches.) For this observation, the word2net objective contains the probability of a binary variable $x_{i,\text{INCREASE}} = 1$ conditional on the context, i.e., the sum of the the context vectors of the surrounding words. This variable represents that INCREASE occurred at position i.

For word2net, the conditional probability of $x_{i,\text{INCREASE}} = 1$ is the output of a multilayer network that takes the context as input. Each layer of the network transforms the context

ME (pron)	
CBOW	**word2net**
like	myself pron
senator	my pron
just	himself pron

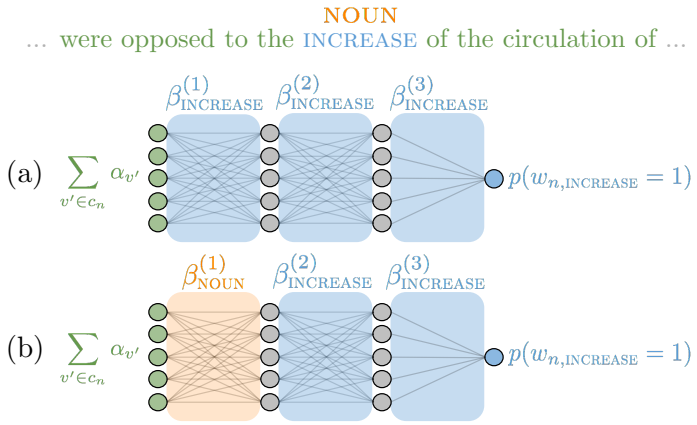
SAY (v)	
CBOW	**word2net**
think	think v
what	know v
just	answer v

Figure 3.6: Word2net can exploit syntactic information to learn semantic similarities. We compare the top 3 most similar words returned by CBOW (left) and word2net (right) to the query words ME and SAY. Additional results and a comparison to CBOW with POS information is in Table 3.15.

into a new hidden representation, reweighting the latent features according to their relevance for predicting the occurrence of INCREASE. In general, the word networks can capture non-linear interactions between co-occurring words; this leads to a better model of language. Note that not illustrated are the 0-variables, i.e., the negative samples, which correspond to terms that are not at position i. In word2net, their probabilities (of being equal to zero) also come from their corresponding word networks.

Another benefit of word2net is that the multilayer architecture of the word networks provides opportunities to share parameters. Here we study word2net models that share parameters based on part-of-speech (POS) tags, where the parameters of certain layers of each network are shared by all terms tagged with the same POS tag.

Figure 3.7b illustrates this idea. The network in this figure is specific to INCREASE as a noun (as opposed to a verb). The parameters of the first layer (orange) are shared among all nouns in the collection; the other layers (blue) are specific to INCREASE. Thus, the networks for INCREASE/NOUN and INCREASE/VERB differ in how the first layer promotes the latent aspects of the context, i.e., according to which context features are more relevant for each POS tag.



INCREASE	
verb (v)	**noun** (n)
decrease v	cut n
doubling v	amount n
increases v	saving n
supply v	limit n
cost v	amounts n
estimates v	decreases n
cut v	increases n
raising v	decrease n
amount v	declines n
half adj	amounting n

(c)

Figure 3.7: An illustration of word2net. **(a)** In word2net, each term v is represented by a neural network with weights $\beta_v^{(\ell)}$. The word network predicts the probability of a target word (INCREASE, shown in blue) from its context (green). The input is the sum of the context vectors and the output is the occurrence probability of the target. **(b)** Word2net can incorporate syntactic information by sharing an entire layer (orange) between words with the same POS tag (NOUN in this case). **(c)** The fitted word2net with POS sharing can be queried for semantic similarities of the word networks for each word/tag pair. In this example, we list the most similar networks to INCREASE/VERB and INCREASE/NOUN.

Word2net with POS can consider the two forms of INCREASE separately. Figure 3.7c shows the most similar words to each type of INCREASE; the method correctly picks out tagged words related to the verb and related to the noun. In similar analyses, Figure 3.6 shows the most similar words to ME as a pronoun and SAY as a verb, both for word2net and vector-based embeddings. Word2net with POS provides more interpretable embeddings.

3.3.1 Model Description of Word2Net

We now formalize the definition of the word networks. Then we describe word2net and how to train the word networks in an unsupervised manner. Finally, we introduce parameter sharing across the word networks, which allows us to incorporate side information, such as POS tags, into the task of learning distributed representations.

Word networks. In word2net, each word is represented by a function, parameterized by a feed-forward neural network $f(\cdot\,;\,\beta_v)$ for each term v in the vocabulary. The parameters β_v capture the semantic properties of that term. We refer to the per-term neural networks as word networks.

Each word network maps a K-dimensional input to a scalar output. We study word networks with 3 layers and $\tanh(\cdot)$ nonlinearities; thus the parameters β_v are the weights of each layer, $\beta_v = \{\beta_v^{(1)}, \beta_v^{(2)}, \beta_v^{(3)}\}$. For an input I, the word network $f(\cdot\,;\,\beta_v)$ outputs

$$f\,(I\,;\,\beta_v) = \tanh\left(\tanh\left(I^\top \beta_v^{(1)}\right)^\top \beta_v^{(2)}\right)^\top \beta_v^{(3)}. \tag{3.12}$$

The two hidden layers have dimensions H_1 and H_2, so the dimensions of the weights are $\beta_v^{(1)} \in \mathbb{R}^{K\times H_1}$, $\beta_v^{(2)} \in \mathbb{R}^{H_1\times H_2}$, and $\beta_v^{(3)} \in \mathbb{R}^{H_2}$.

Word2net model. Word2net is a model of text. The text is a sequence of N words with V distinct vocabulary terms. At each position i in the text, we observe a V-dimensional one-hot vector x_i that indicates which word is at that position. That is, each x_i has exactly one nonzero entry x_{iv}.

Word2net models the conditional probability of each binary entry x_{iv}. The model has two types of parameters. For each vocabulary term v, it has a word network $f(\cdot\,;\,\beta_v)$ and a context vector $\alpha_v \in \mathbb{R}^K$.

Like B-EMB word2net posits a conditional model of words given their context. The context of a given target word is defined as the words in a fixed-size window surrounding the target word. Formally, for location i, let the context c_i be the indices of the words before and after word i. Word2net forms a latent representation of each context c_i by combining the

context vectors α_w of words in the context. The representation Σ_i of the context at position i is the sum of the words surrounding the target words as defined in Equation (1.1).

Given a text corpus, the word networks are trained to discriminate the contexts in which a word appears form contexts in which a word does not appear. Word2net models each entry x_{iv} in the corpus as a Bernoulli random variable whose parameter depends on the word network for the v^{th} term and the context Σ_i. Specifically, the log-odds are the output of the word network that takes the context representation Σ_i as input,

$$p(x_{iv} \mid \boldsymbol{x}_{c_i}) = \text{Bernoulli}\Big(\sigma\big(f(\Sigma_i \; ; \; \beta_v)\big)\Big), \tag{3.13}$$

where the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ transforms the output of the word network into a probability.

Modeling individual indicators of the text (Equation (3.13)) does not impose the constraint that the sum over the vocabulary words $\sum_v p(x_{iv} = 1 \mid \boldsymbol{x}_{c_i})$ must be 1. This modeling choice is standard in word embedding models, and it significantly alleviates the computational complexity over multinomial models, as it allows us to use negative sampling (Mikolov et al., 2013a; Rudolph et al., 2016).

Word2net objective. The objective of word2net is the pseudo-likelihood (Arnold, Castillo, Sarabia, et al., 2001; Rudolph et al., 2016), which is the sum of the log conditional

probabilities of the words in the text,

$$\begin{aligned}\mathcal{L}(\beta, \alpha) &= \sum_{v=1}^{V} \sum_{i=1}^{N} \log p(x_{iv} \mid \boldsymbol{x}_{c_i}) \\ &= \sum_{v=1}^{V} \left(\sum_{i:\, x_{iv}=1} \log \sigma\big(f(\Sigma_i;\ \beta_v)\big) \qquad + \sum_{i:\, x_{iv}=0} \log \sigma\big(-f(\Sigma_i;\ \beta_v)\big) \right).\end{aligned} \tag{3.14}$$

On the second and third line, the objective is separated into the positive entries (positive samples) and zero entries of the text corpus (negative examples). Each location i in the text contributes only one positive term but many $(V-1)$ negative examples.

Word2net fits the objective in Equation (3.14) using SGD (Robbins and Monro, 1951). In the SGD procedure, word2net aggressively subsamples the negative examples, resulting in a training procedure analogous to negative sampling (Mikolov et al., 2013a; Rudolph et al., 2016). For each text position i, the set of negative samples is $\mathcal{S}_i$. Effectively, the last term in Equation (3.14) sums only over the negative samples in $\mathcal{S}_i$ rather than over all the 0 entries. Code for fitting word2net is available on Github under https://github.com/mariru/word2net/.

POS-word2net. Word2net can use the multilayer architecture of the word networks to incorporate meta-data about each word. As an example, we develop POS-word2net for incorporating syntactic information, specifically part-of-speech (POS) tags.

Consider POS-tagged text (e.g., annotated by an off-the-shelf POS tagger). Each observed word x_i has a corresponding tag $s_i \in \{0, 1\}^T$ indicating which one of the T tags is assigned to observation i. This groups the observations into T groups.

Different occurrences of a term may be associated with different groups. One example is

FISH, which can be either a NOUN (the animal) or a VERB (trying to catch the animal). On the one hand, both meanings are related. On the other hand, they may appear in different contexts of other words. Ideally, embedding models should be able to capture the difference. However, the simple approach of considering FISH/NOUN and FISH/VERB as separate terms will fail if there are too few occurrences of each individual term/tag pair. (We show this empirically in Section 3.3.2.)

POS-word2net shares some of the word network parameters across all words with the same POS tag. Let β_t be the parameter associated to tag t, and let index $\ell \in \{1, 2, 3\}$ denote the layer at which the parameters are shared. To compute the probability of a word, we use the tag-specific parameters β_t at layer ℓ and the term-specific parameters β_v at the other layers $\neg\ell$. That is, POS-word2net models the observation at position i with a neural network that is specific to that term/tag combination,

$$p(x_{iv} = 1, s_{it} = 1 \mid c_i) = \sigma\left(f\left(\Sigma_i;\ \beta_v^{(\neg\ell)}, \beta_t^{(\ell)}\right)\right). \tag{3.15}$$

Figure 3.7b illustrates parameter sharing at $\ell = 1$.

What does parameter sharing do? For each term/tag pair, POS-word2net updates both the parameters specific to the tag and the parameters specific to the term. For example, when observing FISH as a NOUN at position i, the model is updated to reflect that Σ_i is both a typical context for a NOUN and that it is a typical context for the term FISH. As a result, words share statistical strength with other observations that have either the same tag or the same term.

General parameter sharing for word2net. POS-word2net is an example of parameter

sharing in word2net. It generalizes to any discrete side information, presuming that each observation belongs to one of T groups.

The network parameters at a specific layer ℓ are shared between the word networks of observations in the same group. When there is only one group, we force all word networks to share parameters at one of the layers. Note this type of sharing does not require side information.

In Section 3.3.2 we compare the performance of word2net with parameter sharing at different layers, either sharing according to POS tags or between all networks. Parameter sharing improves the performance of word2net.

Semantic similarity of word networks. In standard word embeddings, the default choice to compute semantic similarities between words is by cosine distances between the word vectors. Since word2net replaces the word vectors with word networks, we can no longer apply this default choice. We next describe the procedure that we use to compute semantic similarities between word networks.

After fitting word2net, each word is represented by a neural network. Given that these networks parameterize functions, we design a metric that accounts for the fact that two functions are similar if they map similar inputs to similar outputs. So the intuition behind our procedure is as follows: we consider a set of K-dimensional inputs, we evaluate the output of each neural network on this set of inputs, and then we compare the outputs across networks. For the inputs, we choose the V context vectors, which we stack together into a matrix $\boldsymbol{\alpha} \in \mathbb{R}^{V \times K}$. We evaluate each network $f(\cdot)$ row-wise on $\boldsymbol{\alpha}$ (i.e., feeding each α_v as a K-dimensional input to obtain a scalar output), obtaining a V-dimensional summary of

where the network $f(\cdot)$ maps the inputs. Finally, we use the cosine distance of the outputs to compare the outputs across networks. In summary, we obtain the similarity of two words w and v as

$$\text{dist}\,(w, v) = \frac{f(\boldsymbol{\alpha};\ \beta_w)^\top f(\boldsymbol{\alpha};\ \beta_v)}{||f(\boldsymbol{\alpha};\ \beta_w)||_2\ \ ||f(\boldsymbol{\alpha};\ \beta_v)||_2}. \tag{3.16}$$

If we are using parameter sharing, we can also compare POS-tagged words; e.g., we may ask how similar is FISH/NOUN to FISH/VERB. The two combinations will have different representations under the word2net method trained with POS-tag sharing. Assuming that layer ℓ is the shared layer, we compute the semantic similarity between the word/tag pair $[w, t]$ and the pair $[v, s]$ as

$$\text{dist}([w, t], [v, s]) = \frac{f(\boldsymbol{\alpha};\ \beta_w^{(\neg\ell)}, \beta_t^{(\ell)})^\top f(\boldsymbol{\alpha};\ \beta_v^{(\neg\ell)}, \beta_s^{(\ell)})}{||f(\boldsymbol{\alpha};\ \beta_w^{(\neg\ell)}, \beta_t^{(\ell)})||_2\ \ ||f(\boldsymbol{\alpha};\ \beta_v^{(\neg\ell)}, \beta_s^{(\ell)})||_2}. \tag{3.17}$$

3.3.2 Empirical Study of Word2Net

In this section we study the performance of word2net on two datasets, Wikipedia articles and Senate speeches. We show that word2net fits held-out data better than existing models and that the learned network representations capture semantic similarities. Our results also show that word2net is superior at incorporating syntactic information into the model, which improves both the predictions and the quality of the word representations.

Data. We use word2net to study two data sets, both with and without POS tags:

Wikipedia: The text8 corpus of Wikipedia articles contains 17M words. We form a vocabulary with the 15K most common terms, replacing less frequent terms with the

	vocabulary	K	p/V	cs 2	cs 4	cs 8
Mikolov et al., 2013a:						
skip-gram	words	20	40	−1.061	−1.062	−1.071
skip-gram	tagged words	20	240	−2.994	−3.042	−3.042
Mikolov et al., 2013a; Rudolph et al., 2016:						
B-EMB/CBOW	words	20	40	−1.023	−0.976	−0.941
B-EMB/CBOW	words	165	330	−1.432	−1.388	−1.381
B-EMB/CBOW	tagged words	20	240	−1.411	−1.437	−1.461
this work:	sharing					
word2net		20	330	−0.940	−0.912	−0.937
word2net	ALL	20	≈ 120	−1.040	−1.003	−0.964
word2net	ALL	20	≈ 230	−1.191	−1.141	−1.111
word2net	ALL	20	≈ 320	−0.863	−0.881	−0.890
word2net	POS	20	≈ 120	−0.918	−0.914	−0.871
word2net	POS	20	≈ 230	−0.844	**−0.801**	**−0.793**
word2net	POS	20	≈ 320	**−0.840**	−0.822	−0.862

Table 3.11: Word2net outperforms existing word embedding models (skip-gram and B-EMB/CBOW) in terms of test log-likelihood on the Wikipedia data, both with and without POS tags. We compare models with the same context dimension K and the same total number of parameters p/V for different context sizes (cs). For word2net, we study different parameter sharing schemes, and the color coding indicates which layer is shared and how, as in Figure 3.7. Parameter sharing improves the performance of word2net, especially with POS tags.

UNKNOWN token. We annotate text8 using the NLTK POS tagger and the universal tagset.[8] We also form a tagged dataset in which each term/tag combination has a unique token, resulting in a vocabulary of 49K tagged terms.

Senate speeches: These are the speeches given in the U.S. Senate in the years 1916-2009. The data is a transcript of spoken language and contains 24M words. As above, we form a vocabulary of 15K terms. We annotate the text using the Stanford CoreNLP POS tagger (Manning et al., 2014), and we map the tags to the universal tagset. We form a tagged dataset with 38K tagged terms.

The Senate speeches contain a lot of boilerplate repetitive language; for this reason,

[8] See http://nltk.org.

	vocabulary	K	p/V	cs 2	cs 4	cs 8
Mikolov et al., 2013a:						
skip-gram	words	20	40	−1.052	−1.080	−1.061
skip-gram	tagged words	20	240	−1.175	−1.199	−1.227
Mikolov et al., 2013a; Rudolph et al., 2016:						
B-EMB/CBOW	words	20	40	−1.274	−1.246	−1.222
B-EMB/CBOW	tagged words	20	240	−1.352	−1.340	−1.339
B-EMB/CBOW	words	165	330	−1.735	−1.734	−1.744
this work:	sharing					
word2net		20	330	−1.406	−1.555	−1.401
word2net	ALL	20	≈ 120	−1.276	−1.256	−1.243
word2net	ALL	20	≈ 230	−1.462	−1.435	−1.413
word2net	POS	20	≈ 120	**−0.873**	**−0.860**	**−0.850**
word2net	POS	20	≈ 230	−1.057	−1.034	−1.015

Table 3.12: Comparison of the test log-likelihood across different models on the Senate speeches. We compare models with the same context dimension K and the same total number of parameters p/V for different context sizes ("cs"). For word2net, we explore different parameter sharing schemes. The color coding of the parameter sharing (same as Figure 3.7) indicates which layer is shared and how.

we tokenize around 350 frequent phrases, such as SENATOR FROM ALABAMA or UNITED STATES, considering the entire phrase an individual vocabulary term. We apply the POS tagger before this tokenization step, and then we assign the NOUN tag to all phrases.

For both datasets, we subsample the frequent words following Mikolov et al., (2013a); i.e., each word x_i in the training set is discarded with probability $1 - \sqrt{\frac{t}{\text{frequency}(x_i)}}$, with $t = 10^{-5}$. This subsampling step has empirically been shown to speed up training and is standard for most embedding approaches.

Table 3.8 summarizes the information about both corpora. We split each dataset into a training set (90% of words), a validation set (5% or words), and a test set (5% or words).

Methods. We compare word2net to its shallow counterpart, the CBOW model (Mikolov et al., 2013a), which is equivalent to B-EMB (Rudolph et al., 2016), and we also compare to skip-gram (Mikolov et al., 2013a). With context vectors α_v and context representations

	vocabulary	K	p/V	cs 2	cs 4	cs 8
Mikolov et al., 2013a:						
skip-gram	words	100	200	−1.107	−1.053	−1.043
skip-gram	tagged words	100	1200	−3.160	−3.151	−3.203
Mikolov et al., 2013a; Rudolph et al., 2016:						
B-EMB/CBOW	words	100	200	−1.212	−1.160	−1.127
B-EMB/CBOW	tagged words	100	1200	−1.453	−3.387	−3.433
B-EMB/CBOW	words	1260	2520	−3.772	−2.397	−2.506
this work:	sharing					
word2net		100	2520	−1.088	−1.049	−1.012
word2net	ALL	100	≈ 520	−1.041	−0.988	−1.001
word2net	ALL	100	≈ 2120	−1.114	−1.059	−1.016
word2net	POS	100	≈ 521	**−0.828**	**−0.807**	**−0.770**
word2net	POS	100	≈ 2120	−0.892	−0.850	−0.822

Table 3.13: Comparison of the test log-likelihood across different models on the Wikipedia dataset. We compare models with the same context dimension K and the same total number of parameters p/V for different context sizes ("cs"). For word2net, we explore different parameter sharing schemes. The color coding of the parameter sharing (same as Figure 3.7) indicates which layer is shared and how.

Σ_i defined as in Equation (1.1), and with one embedding vector $\rho_v \in \mathbb{R}^K$ per term, their objectives are

$$\mathcal{L}_{\text{cbow}}(\rho, \alpha) = \sum_i \left(\sum_{v:\, x_{iv}=1} \log \sigma(\rho_v^\top \Sigma_i) + \sum_{v \in \mathcal{S}_i} \log \sigma(-\rho_v^\top \Sigma_i) \right) \tag{3.18}$$

and

$$\mathcal{L}_{\text{skip-gram}}(\rho, \alpha) = \sum_{\substack{(i,v):\\ x_{iv}=1}} \left(\sum_{v' \in c_i} \log \sigma\big(\rho_v^\top \alpha_{v'}\big) + \sum_{v' \in \mathcal{S}_i} \log \sigma\big(-\rho_v^\top \alpha_{v'}\big) \right) \tag{3.19}$$

We fit B-EMB/CBOW and skip-gram both to the untagged data and to the augmented data of POS-tagged terms. Chapter 1 contains detailed derivations on how these objectives relate to each other through Jensen's inequality as well as additional details on negative sampling.

In summary, the methods we compare are:

- B-EMB/CBOW: Learns vector representations for each word (or tagged word) by optimizing Equation (3.18).
- *Skip-gram*: Learns vector representations for each word (or tagged word) by optimizing Equation (3.19).
- *Word2net*: Learns a neural network representation for each word by optimizing Equation (3.14). We study the following parameter sharing schemes:
  1. : no parameter sharing.
  2.  : layer ℓ shared between all networks.
  3.  : layer ℓ shared between terms with the same part-of-speech (POS).

Algorithm details. We fit word2net with the context dimensions $K \in \{20, 100\}$. The context dimension is also the dimension of the input layer. For $K = 20$, we use $H_1 = 10$ hidden units in the first hidden layer of each word network and $H_2 = 10$ hidden units in the second layer. For $K = 100$, we use $H_1 = H_2 = 20$ hidden units. Without parameter sharing, the number of parameters per word is $K + KH_1 + H_1H_2 + H_2$. The shallow models have $2K$ parameters per term (the entries of the context and word vectors). Since we want to compare models both in terms of context dimension K and in terms of total parameters, we fit the methods with $K \in \{20, 165, 100, 1260\}$.

We set the context sizes $|c_i| \in \{2, 4, 8\}$ and we train all methods using SGD (Robbins and Monro, 1951) with $|\mathcal{S}_i| = 10$ negative samples on the Wikipedia data and with $|\mathcal{S}_i| = 20$ negative samples on the Senate speeches. Following Mikolov et al., (2013a), we draw the negative samples from the unigram distribution raised to the power of 0.75. We use ℓ-2

regularization, corresponding to a Gaussian prior on the parameters with standard deviation 10 for the word vectors, the context vectors, and the network weights.

For the step size, we use Adam (Kingma and Ba, 2015) with Tensorflow's default settings (Abadi et al., 2016) to train all methods for up to 30000 iterations, using a minibatch size of 4069 or 1024. We assess convergence by monitoring the loss on a held-out validation set every 50 iterations, and we stop training when the average validation loss starts increasing.

We initialize and freeze the context vectors of the word2net methods with the context vectors from a pretrained Bernoulli embedding with the same context dimension K. Network parameters are initialized according to standard initialization schemes of feed-forward neural networks (Glorot and Bengio, 2010), i.e., the weights are initialized from a uniform distribution with bounds $\pm\sqrt{6}/\sqrt{H_{\text{in}} + H_{\text{out}}}$.

RATE (3000)	
CBOW	**word2net**
expectancy	capacity
per	amount
increase	energy

COFFEE (500)	
CBOW	**word2net**
bananas	beans
potatoes	juice
pepper	poultry

PARROT (70)	
CBOW	**word2net**
turtle	dolphin
beaver	crow
pratchett	dodo

Table 3.14: The word networks fitted using word2net capture semantic similarities. We compare the top 3 similar words to several query words (shaded in gray) for CBOW/B-EMB and word2net, trained on the Wikipedia dataset. The numbers in parenthesis indicate the frequency of the query words.

Word2net has better predictive performance. We compute the log-likelihood of the words in the test set, $\log p(x_{iv} \mid \boldsymbol{x} c_i)$. For skip-gram, which was trained to predict the context words from the target, we average the context vectors α_v for a fair comparison.[9]

Table 3.11 shows the results for the Wikipedia dataset. We explore different model sizes:

[9]If we do not average, the held-out likelihood of skip-gram becomes worse.

ME (pron)		
CBOW	**CBOW POS**	**word2net**
like	governor n	myself pron
senator	sen. from alabama n	my pron
just	used adj	himself pron

BECAUSE (sc)		
CBOW	**CBOW POS**	**word2net**
but	unemployed n	as sc
reason	annuity n	that sc
that	shelled v	through sc

CAUSES (n)		
CBOW	**CBOW POS**	**word2net**
fatal	pro adj	consequences n
consequences	enough adv	clash n
coupled	positions n	handicaps n

SAY (v)		
CBOW	**CBOW POS**	**word2net**
think	time v	think v
what	best adj	know v
just	favour n	answer v

Table 3.15: Word2net learns better semantic representations by exploiting syntactic information. The top 3 similar words to several queries are listed for different models fitted to the Senate speeches. We compare CBOW trained without POS tags (left), CBOW with POS tags (center), and word2net with POS parameter sharing (right). The POS tags are noted in orange. Parameter sharing helps word2net capture better semantic similarities, while adding the POS information to CBOW hurts its performance.

with the same number of parameters as word2net, and with the same dimensionality K of the context vectors. For word2net, we explore different parameter sharing approaches. Table 3.13 shows the results for other model sizes (including $K = 100$). In both tables, word2net without parameter sharing performs at least as good as the shallow models. Importantly, the performance of word2net improves with parameter sharing, and it outperforms the other methods.

Tables 3.11 and 3.13 also show that B-EMB/CBOW and skip-gram perform poorly when we incorporate POS information by considering an augmented vocabulary of tagged words. The reason is that each term becomes less frequent, so these approaches would require more data to capture the co-occurrence patterns of tagged words. In contrast, word2net with POS parameter sharing provides the best predictions across all methods (including other versions of word2net).

Finally, Table 3.12 shows the predictive performance for the U.S. Senate speeches. On

	ALL	POS	POS	[Huang et al., 2012]
Spearman ρ	45%	44%	38%	66%

Table 3.16: Contextual similarities of different word2net architectures.

this corpus, skip-gram performs better than B-EMB/CBOW and word2net without parameter sharing; however, word2net with POS sharing also provides the best predictions across all methods.

Can POS sharing contextualize the word similarities of word2net? We study word2net's performance on an external word similarity task. The Stanford's Contextual Word Similarities (SCWS) corpus (Huang et al., 2012), contains human ratings of word similarities in context. SCWS contains pairs of words as well as two sentences that give the words a context. The human rating reflects how similar the meaning of the two words are to each. Especially for ambiguous words, the sentences provided with each example provide context for which meaning of the word is intended. Each pair of words is additionally associated with their POS tags. In our study of how word2net performs on this task, we ignore the sentences, but use the POS tags to compute similarity scores for each example in the corpus. Performance is reported in terms of the Spearman correlation coefficient which measures how correlated the similarity scores of word2net are with the human scores.

Table 3.16 shows how well different word2net fits perform. The reported numbers are the Spearman correlation coefficients. All three word2net fits have input dimension $K = 20$, and context size 2, and have been trained on the Senate speeches.

Word2net captures similarities and leverages syntactic information. Table 3.14 displays the similarity between word networks (trained on Wikipedia with parameter sharing

at layer $\ell = 1$), compared to the similarities captured by word embeddings (B-EMB/CBOW). For each query word, we list the three most similar terms, according to the learned representations. The word vectors are compared using cosine similarity, while the word networks are compared using Equation (3.16). The table shows that word2net can capture latent semantics, even for less frequent words such as PARROT.

Table 3.15 shows similarities of models trained on the Senate speeches. In particular, the table compares: B-EMB/CBOW without POS information, B-EMB/CBOW trained on the augmented vocabulary of tagged words, and word2net with POS parameter sharing at the input layer ($\ell = 1$). We use Equation (3.17) to compute the similarity across word networks with POS sharing. We can see that word2net is superior at incorporating syntactic information into the learned representations. For example, the most similar networks to the pronoun ME are other pronouns such as MYSELF, MY, and HIMSELF. Word networks are often similar to other word networks with the same POS tag, but we also see some variation. One such example is in Figure 3.7c, which shows that the list of the 10 most similar words to the verb INCREASE contains the adjective HALF.

3.3.3 Discussion of Word2Net

We have presented word2net, a method for learning neural network representations of words. The word networks are used to predict the occurrence of words in small context windows and improve prediction accuracy over existing log-bilinear models. We combine the context vectors additively, but this opens the door for future research directions in which we explore other ways of combining the context information, such as accounting for the order of the

context words and their POS tags.

We have also introduced parameter sharing as a way to share statistical strength across groups of words and we have shown empirically that it improves the performance of word2net. Another opportunity for future work is to explore other types of parameter sharing besides POS sharing, such as sharing layers across documents or learning a latent group structure together with the word networks.

3.4 MayHEM: Multilingual Hierarchical Embeddings

The idea of using hierarchical priors to share information between embeddings is also useful for multilingual embeddings. Imagine we would like to learn embeddings from multiple languages. We have two desiderata. First, we want the language specific embeddings to capture semantic similarities between the words in that language and we want them to be of high quality whether the language is resource-rich or resource-lean. Ideally, useful patterns from the embeddings of the resource-rich languages will transfer to the other embeddings. Second, we would like the embeddings of all the languages to be in one shared embedding space. The shared embedding space can be useful for specific cross-lingual tasks such as translation or for further crosslingual study of semantics.

Here we present MAYHEM a model that extends B-EMB to text data from multiple languages. The text data does not need to be aligned, in other words the text data for the different languages can be from arbitrary independent sources. To model the relationship between the languages MAYHEM requires a dictionary for each language that specifies the English translation for a fraction of the vocabulary terms. (We study the sensitivity of

MAYHEM to the size of the dictionary in Section 3.4.2).

The survey of cross-lingual embeddings by Ruder, Vulić, and Søgaard, (2017) contains a typology of many existing methods for embeddings from multiple languages. There are many methods for learning bilingual embeddings, including (Mikolov, Le, and Sutskever, 2013; Faruqui and Dyer, 2014; Luong, Pham, and Manning, 2015; Smith et al., 2017). The approach of Mikolov, Le, and Sutskever, (2013) is to separately learn embeddings in two languages (e.g. English and Spanish) and then use a small bilingual dictionary to learn a linear mapping that projects the two embeddings into the same space. Faruqui and Dyer, (2014) and Smith et al., (2017) respectively use correlated component analysis (CCA) and orthogonal projections as mapping between the pretrained monolingual embeddings. MultiCCA (Ammar et al., 2016) extends the idea around using CCA to relate monolingual embeddings to each other to multiple languages. Like MAYHEM, these mapping based methods require text in each language to learn the monolingual embeddings as well as small dictionaries that relate the languages to each other. Unlike MAYHEM the embeddings in each language are pretrained separately whereas for MAYHEM we will train all the embeddings jointly.

There are also methods that use no parallel data and no dictionaries to learn multilingual embeddings. The models of Conneau et al., (2017) and Lample, Denoyer, and Ranzato, (2017) rely on pretrained monolingual embeddings to have a similar topology and then infer a dictionary that maps the two monolingual embeddings onto each other in a way that preserves that structure. However, like the mapping based approaches (Mikolov, Le, and Sutskever, 2013; Faruqui and Dyer, 2014; Ammar et al., 2016) the embeddings are pretrained separately from each other and so there is no hope that the embeddings of low-resource

languages can levarage any structural information from the embeddings of a resource-rich language such as English.

Luong, Pham, and Manning, (2015) jointly train bilingual embeddings on parallel corpora (i.e. sentences that are known to be translations of each other) that have been aligned (Dyer, Chahuneau, and Smith, 2013). Both embeddings come from an augmented skipgram objective (Mikolov et al., 2013b) which encourages the embeddings to be predictive both of the surrounding context and of the context surrounding the corresponding word in the aligned translation.

While MAYHEM also learns the multilingual embeddings jointly, it does not require parallel data and hence can handle more than two languages. Instead it uses ideas from hierarchical embedding models (Section 3.2) to leverage unrelated texts in different languages and the weak supervision of small dictionaries. The text in each language can be seen as its own group and the dictionaries are used to define a hierarchical prior which both grounds all the languages in one joint embedding space and manages how information is shared between the languages. The model is presented in Section 3.4.1 and in Section 3.4.2 we study the performance of MAYHEM.

3.4.1 Model Description of MayHEM

MAYHEM is a model for learning multilingual embeddings. As input, it requires text data in different languages. These texts do not necessarily need to be translations of each other. Instead, the method requires a dictionary for each language, that specifies how a subset of the vocabulary translates to another language. MAYHEM processes the text data and the

dictionaries and produces word embeddings for each of the languages in one joint embedding space. MAYHEM is a probabilistic model of the multilingual text. The latent variables of the model are the word embeddings. Key to the success of MAYHEM is that the embeddings in the different languages share statistical information through hierarchical priors that are defined using the dictionaries.

In this section, we first formalise the format of the input text data and dictionaries, as well as the desired output – multilingual word embeddings. We then develop MAYHEM, an extension of B-EMB for learning multilingual embeddings from such data. After describing how to fit MAYHEM, we present an empirical study of its multilingual embeddings in Section 3.4.2.

Input data: We assume we have L text corpora $X^{(1)}, \cdots, X^{(L)}$, each in a different language. Each language ℓ, has its own vocabulary $\mathcal{V}^{(\ell)}$, truncated to the $V = |\mathcal{V}^{(\ell)}|$ most frequent words. While each corpus is truncated to the same vocabulary size V, the lengths in terms of number of words $N^{(\ell)}$ can differ from language to language. For English, the text contains millions of words, while for low-resource languages such as Latin, we have much fewer words.

Each text is encoded as binary matrix $X^{(\ell)} \in \{0, 1\}^{N^{(\ell)} \times V}$, with the entries $x_{iv}^{(\ell)}$ indicating whether vocabulary term v is at position i. Text is a sequence of words with exactly one term at each position. Hence the matrix rows are constrained to sum to one (i.e. for all text positions $1 \leq i \leq N^{(\ell)}$, $\sum_v x_{iv}^{(\ell)} = 1$).

For each language, we also have a partial English dictionary $\mathcal{D}^{(\ell)}$ which maps a subset of the terms in the vocabulary $\mathcal{S}^{(\ell)} \subset \mathcal{V}^{(\ell)}$ to their English translation [10]. The subset as well as

[10] Assuming that all words have an English translation simplifies the notation, but MAYHEM can also be

its size differs from language to language. Specifically, for each term $v \in \mathcal{S}^{(\ell)}$, the dictionary return a list of English translations, $\mathcal{D}^{(\ell)}(v)$. In the dictionaries we work with, a word has typically one translation, but many words also have two or three English translations.

Multilingual word embeddings: The goal of MAYHEM is to process such data and learn a set of word embeddings for each language. We would like the multilingual embeddings to live in the same space such that cosine similarities reflect both semantic similarities between words in the same language and across different languages. For each language, each vocabulary term $v \in \mathcal{V}^{(\ell)}$ is associated with two types of vectors. An embedding vector $\rho_v^{(\ell)} \in \mathbb{R}^K$, and a context vector $\alpha_v^{(\ell)} \in \mathbb{R}^K$. These parameters are combined in the conditional co-occurrence probabilities of the words.

Model: MAYHEM is an extension of B-EMB (Chapter 1). Each observation $x_{iv}^{(\ell)}$ is modeled conditionally on the words that appear before it and after in a context window of fixed size, typically between 2 and 10 words. Let c_i be the indices of the words before and after position i in the text and let $\boldsymbol{x}_{c_i}^{(\ell)}$ denote the context words.

In MAYHEM, the conditional probability of the target word given its context is parameterized via a linear combination of the embedding vector and the context vectors,

$$p(x_{iv}^{(\ell)} \mid \boldsymbol{x}_{c_i}^{(\ell)}) = \text{Bernoulli}\big(\sigma(\rho_v^{(\ell)\top} \Sigma_i^{(\ell)})\big), \qquad \text{with} \qquad \Sigma_i^{(\ell)} \triangleq \sum_{(j,w)\in c_i} \alpha_w^{(\ell)} x_{jw}^{(\ell)}. \quad (3.20)$$

Here, $\sigma(x) = \frac{1}{1+e^{-x}}$ is again the sigmoid function, and we have introduced the notation $\Sigma_i^{(\ell)}$ for the sum of the context vectors at location i.

The different languages share statistical information through a regularization scheme on

---

trained with other types of dictionaries.

the embeddings that resembles hierarchical modeling in Bayesian statistics (Gelman et al., 2003). We introduce an additional set of parameters $\rho_v^{(0)} \in \mathbb{R}^K$, and $\alpha_v^{(0)} \in \mathbb{R}^K$ for each word in the English vocabulary $v \in \mathcal{V}^{(\text{english})}$.

The dictionaries define the structure of the prior. For each language ℓ, for each term $v \in \mathcal{S}^{(\ell)}$ and its translation $w = \mathcal{D}^{(\ell)}(v)$, the model assumes

$$\rho_v^{(\ell)} \sim \mathcal{N}(\rho_w^{(0)}, \lambda^{-1}\mathbb{I}) \tag{3.21}$$

$$\alpha_v^{(\ell)} \sim \mathcal{N}(\alpha_w^{(0)}, \lambda^{-1}\mathbb{I}). \tag{3.22}$$

As an example, consider the German word EI which translates to EGG. Equation (3.21) stipulates that the prior of the German embedding for EI is centered at the prior parameter associated with EGG. When a word has multiple translations (e.g. the German word UMFRAGE translates to both POLL and SURVEY), we instead set the prior mean to the average of the parameter vectors associated with the different translations [11].

Fitting the model: Since the model is conditionally specified, we learn the embeddings from the data by optimizing a regularized pseudo-likelihood (Arnold, Castillo, Sarabia, et al., 2001; Rudolph et al., 2016). The regularized pseudo-likelihood is the sum of the log conditionals of the observed data, plus a regularization term in the form of a log prior

$$\mathcal{L}(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_{\ell} \sum_{i=1}^{N^{(\ell)}} \sum_{v=1}^{V} \log p(x_{iv}^{(\ell)} \mid \boldsymbol{x}_{c_i}^{(\ell)}) + \mathcal{L}_{\text{prior}}. \tag{3.23}$$

[11] Another possibility would be to pick the more frequent translation for the prior parameter or to reweight the averages by the word frequencies. In the dictionaries we worked with, most words had a single translation, so we expect the effect of these design decisions to be negligible. Comparing them is kept for future work.

The regularization term $\mathcal{L}_{\text{prior}}$ is determined by Equation (3.21) and has the form

$$\mathcal{L}_{\text{prior}} = \sum_{\ell} \sum_{v \in \mathcal{S}^{(\ell)}} -\frac{\lambda}{2}||\alpha_v^{(\ell)} - \alpha_w^{(0)}||^2 - \frac{\lambda}{2}||\alpha_v^{(\ell)} - \alpha_w^{(0)}||^2, \tag{3.24}$$

where $w = \mathcal{D}^{(\ell)}(v)$. This regularization scheme encourages words which are known to be translations of each other to be close in the joint embedding space. The regularization parameter λ determines the strength of how much the log prior competes with the likelihood term in Equation (3.23).

Some words, even though they are translations of each other can be used very differently in different languages for various reasons. For example, a grammatical rule might prevent two words from ever co-occurring in one language, while in another language with a different grammar, the words co-occur frequently. Other reasons for the embeddings of a word to differ between languages could be cultural. Consider the word FOOD, and its translation into the different languages. While in principle it refers to the same thing, namely "nutritious sustenance that people or animals eat or drink", food culture (i.e. what people eat, when and why) varies drastically around the globe and so it is natural to assume that the contexts in which the word FOOD will occur in different languages will vary.

The hierarchical prior let's the data determine whether the different multilingual embeddings of food should be close to their prior or whether the data supports some variation.

We implement mayhem in Tensorflow (Abadi et al., 2016) and the objective Equation (3.23) is optimized using stochastic gradients (Robbins and Monro, 1951) with adaptive gradients (Adam) (Kingma and Ba, 2015). To draw a minibatch to estimate the gradient at each gradient step, we draw an equal amount of consecutive text from every language.

We obtain further speed-ups by negative sampling (Mikolov et al., 2013a), which biases the gradient estimates. It aggressively subsamples terms of the gradients which correspond to likelihood terms of zero data without rescaling their contribution.

3.4.2 Empirical Study of MayHEM

We study the multilingual embeddings learned by MAYHEM. After a description of the data sources and how the data is preprocessed, we address a number of experimental questions. Do the low-resource languages benefit from being trained jointly with the other languages? How does MAYHEM compare to existing methods such as multiCCA in terms of translation accuracy? How sensitive is MAYHEM to the size and quality of the dictionaries?

The quantitative evaluation is broken down into two parts. Part 1 assesses the model fitness of MAYHEM under different experimental conditions. Model fitness is evaluated in terms of log-likelihood of held-out texts. The second part of our quantitative study investigates the translation quality of embeddings learned by MAYHEM. How well does it reflect the dictionary relations it has been trained on and what happens when parts of the dictionary are held-out? We also compare its translation accuracy to multiCCA (Ammar et al., 2016)..Finally, we present qualitative results and show that MAYHEM successfully captures semantic similarities across languages.

Multilingual data and data preprocessing. We study MAYHEM on a set of text corpora in different languages. We investigate the performance of mayhem by training it on language-pairs as well as larger sets of languages (10 and 59). For the language pairs, English is always one of the languages and we pair it with 9 different other languages. The

	vocabulary	dictionary	single	bilingual	10-languages	10-l-d-75	10-l-d-50
en	15K		-1.20		-1.24	-1.24	-1.23
af	15K	8326	-1.37	-1.32	-1.34	-1.34	-1.34
de	15K	7022	-1.26	-1.26	-1.26	-1.26	-1.25
es	15K	8297	-1.24	-1.23	-1.23	-1.23	-1.23
fr	15K	8618	-1.28	-1.27	-1.24	-1.25	-1.26
ka	15K	5649	-2.58	-2.56	-2.17	-2.28	-2.38
la	15K	2681	-5.80	-6.92	-7.07	-7.00	-7.00
ne	15K	6136	-2.14	-1.93	-1.88	-1.94	-2.01
zh	15K	9225	-1.20	-1.20	-1.20	-1.22	-1.21
zu	15K	3831	-5.92	-5.16	-5.50	-5.33	-5.54

Table 3.17: The model fitness of MAYHEM increases as more languages are added.

corpora come from Ammar et al., (2016), and are a combination of the Leipzig Corpora Collection (Goldhahn, Eckart, and Quasthoff, 2012), and (for 12 of the languages) includes also Europarl (Koehn, 2005). Even though Europarl are parallel corpora, mayhem ignores this information. In principal, it can be trained on any monolingual corpus, as long as a dictionary is available for parts of the vocabulary. With an empty dictionary, mayhem reduces to fitting standard word embeddings without alignment between the languages.

Each of the collections is tokenized, and we restrict each language to its $V = 15K$ most frequent tokens. The dictionaries between any two languages are then restricted to include only pairs where both words are still included the resulting vocabulary. We subsample the frequent words following Mikolov et al., (2013a); i.e., each word x_i in the training set is discarded with probability $\text{Prob}(x_i \text{ is discarded}) = 1 - \sqrt{\frac{t}{\text{frequency}(x_i)}}$, where $\text{frequency}(x_i)$ denotes the frequency of word x_i, and $t = 10^{-5}$. We split the monolingual data into training (80%), validation (10%), and test (10%) sets.We use 20 negative samples.

Since one of our objectives is to understand if there is an advantage from including as many languages as possible, we experiment with different language sets (of different size).

ml-59: The multilingual-59 (ml-59) version of the dataset contains all 59 languages of

			dict 100%		dict 75%		dict 50%	
	K	dict	prec @1	prec @10	prec @1	prec @10	prec @1	prec @10
MAYHEM	100	100%	91.78%	99.00%				
MAYHEM	100	75%			35.98%	51.22%		
MAYHEM	100	50%					28.77%	44.70%
multiCCA	40	100%	28.15%	49.71%	44.83%	65.94%	46.47%	66.78%
multiCCA	512	100%	48.97%	65.16%	58.36%	76.14%	52.52%	71.92%

Table 3.18: Translation quality of MAYHEM and multiCCA. MAYHEM captures its training dictionary (dict 100%) almost perfectly, while its translation accuracy decays as larger parts of the dictionary are held out. For reference, we include the translation accuracy of multiCCA. In the main text we discuss why the results are difficult to compare.

the empirical study of (Ammar et al., 2016).

ml-10: This version contains a subset of the languages of ml-59 including English, German, Spanish, French, Afrikaans, Georgian, Latin, Nepalese, Chinese, and Zulu.

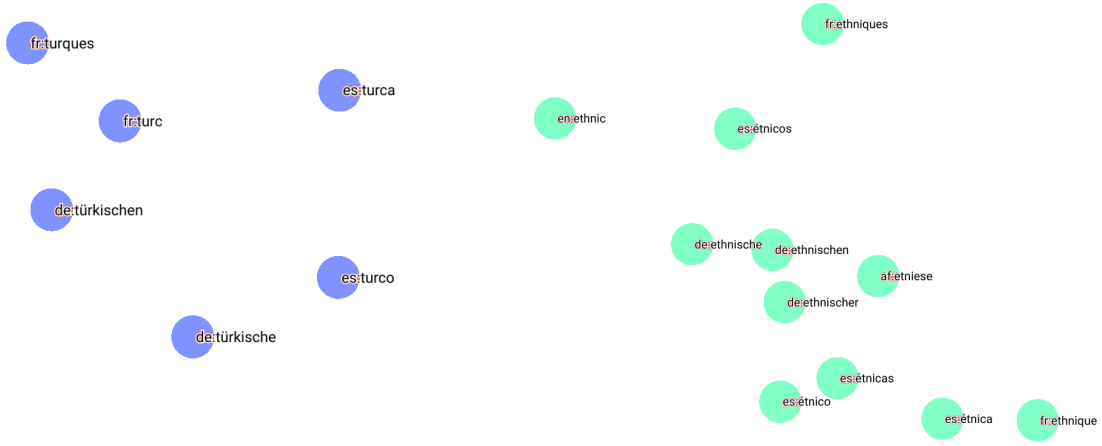
bl: We also run mayhem on 11 bilingual (bl) language pairs. English is always included as one of the languages and the second language is one of the other ml-12 languages.

Evaluation Metrics We evaluate mayhem on two quantitative metrics, model fitness measured in held-out likelihood, as well as translation accuracy on held-out dictionaries.

Held-out log likelihood: For each fit, we compute the test pseudo log-likelihood (Equation (3.20)) of the text that has been held out for each language. For each test entry, a better model will assign higher probability to the observed words, and lower probability to the negative samples.

Translation: Given a dictionary, the translation quality of a fit is evaluated by finding the nearest English neighbours for each foreign word in the dictionary and measuring the precision and the precision at 10 of the translations. For each foreign word, the English translation candidates are the English words whose embedding is closest to the embedding of the foreign word in terms of cosine similarity.

(a) MAYHEM embeddings of people and ethnic minorities in the Middle East.



(b) zoom in on (a) on translations of TURKISH.

(c) zoom in on (a) on translations of ETHNIC.

Figure 3.8: MAYHEM captures semantic similarities between words in different languages.

Predictive performance of MAYHEM. In Table 3.17 is an assessment of MAYHEM's model fitness under different experimental conditions. The columns "single", "bilingual", and "10-languages" compare how much the embeddings benefit from being trained jointly.

For "single", the languages are trained individually[12], "bilingual" means each language is trained together with English (using a bilingual MAYHEM), and "10-languages" are the results of training the 10 languages together. For some of the languages such as German (de), it makes almost no difference whether the embeddings are trained in isolation or not. For other languages, such as Latin (la), the held-out predictions get worse as more languages are added. For Georgian (ka) and Nepalese (ne), held-out predictions improve.

We also study the effect of using a smaller dictionary during training. The columns "10-l-d-75" and "10-l-d-50", are also MAYHEM fits using 10 languages but using smaller dictionaries. Compared to the dictionaries used to define the prior for MAYHEM "10-languages", they respectively only use 75% and 50% of the dictionary associations. In terms of held-out likelihood MAYHEM does not seem too sensitive to the size of the dictionary, probably because the monolingual fits ("single") are not too bad individually under this metric.

Translation accuracy of MAYHEM. In Table 3.18 we asses the translation quality of MAYHEM and we try to compare it to multiCCA. Here, too we study the effect of holding out part of the dictionary. Most importantly, when holding out a chunk of the dictionary (e.g. 75%), then we obtain a held-out dictionary containing 25% of the original dictionary which we can use as a more challenging evaluation task. MAYHEM captures the dictionary it was trained on almost perfectly (columns “dict 100%”). However, the precision decreases as only a subset of the dictionaries are used during training and MAYHEM has to produce translations for held-out dictionaries. For comparison, we also include the translation accuracy of multiCCA. Unfortunately, it is difficult to draw any comparisons between MAYHEM and

[12]in the monolingual case MAYHEM reduces to training a B-EMB (Chapter 1).

mutliCCA from this study. We do not have multiCCA fits where parts of the dictionary have been held-out, and so it is important to note that the results in the "dict 75%" and "dict 50%" columns for MAYHEM and multiCCA can not be compared directly. For MAYHEM the evaluation is on held-out dictionaries while for multiCCA the evaluation pairs are part of the dictionary it was trained on. Another discrepancy is the number of dimensions. We can see for multiCCA that when the number of hidden dimensions increases from $K = 40$ to $K = 512$, its translation performance improves. This gives hope that MAYHEM will also benefit from training it with higher dimensions (but we defer it to future work).

Multilingual Semantic Similarities. We conclude the empirical study of MAYHEM with a qualitative analysis of its multilingual embeddings. Figure 3.8 shows embeddings learned by MAYHEM in the vicinity of the English word KURDS. We can see that the embeddings do not necessarily cluster by language but instead capture cross lingual semantic similarities. The embeddings in Figure 3.8 (a) are multilingual references to people and ethnic minorities in Eastern Europe and the Middle East. By zooming in on two clusters, we seen in Figure 3.8 (b) and (c), that the translations of a word (for example TURKISH and ETHNIC) gather together very closely in the embedding space.

Conclusion

We have presented exponential family embedding (EF-EMB), a class of models for learning distributed representations from high dimensional data. Depending on the type of data and the application, the hope is that these representations capture semantically meaningful similarities. A model from the EF-EMB model class has three ingredients; a context which determines the conditioning set for each observation, a conditional distribution from the exponential family, and the embedding parameters as latent variables.

In Chapter 1, we first introduced these ideas for text and showed how the Bernoulli embedding (B-EMB) gives a new perspective on existing methods for word embeddings. Negative sampling (Mikolov et al., 2013a) can be understood as a biased stochastic gradient procedure on the pseudo-likelihood of a B-EMB, which models word probabilities without accounting for the fact that there is only one word at each position in a sentence. This modeling choice is essential for scaling embedding methods to large data and large vocabularies.

In Chapter 2, we developed EF-EMBS for applications beyond text and showed how to learn embeddings of zebrafish neurons, grocery items, and movies that capture interesting semantic structure. Our empirical study has demonstrated that for these applications an EF-EMB can reconstruct held-out data better than matrix factorization-based approaches.

Treating the embeddings as latent variables of a probabilistic model, lets us use priors to

impose structure and domain knowledge on the embeddings. We highlight in Chapter 3 the potential of this modeling approach by developing various extensions of EF-EMBS. What all the extensions of Chapter 3 have in common is that they model text that has additional information such as year, origin of the speaker, syntactic annotations, or the language of the text. The goal is to have the embeddings be adaptive to the different conditions.

We do not want to simply train a separate embedding for each of those conditions for two reasons. First, separately trained embeddings are not grounded in the same space and hard to compare. The embeddings need to be post-processed with ad-hoc embedding alignement techniques to make it possible to interpret differences in word meaning between the groups. Second, we need to carefully think whether each group has enough data to train a separate embedding for each term. The methods we develop address both of these concerns. The embeddings are trained in one joint space, and we develop ways to help share statistical information between the embeddings associated with the different conditions.

With each extension of EF-EMB presented in Chapter 3, we provide a different way to share information between the embeddings. For dynamic embeddings (Section 3.1) information is shared between consecutive time slices via a Gaussian random walk prior while for hierarchical embeddings a hierarchical prior manages how information is shared between all groups (Section 3.2). For multilingual hierarchical embeddings (MAYHEM) (Section 3.4), a different kind of hierarchical embedding model, the hierarchical prior manages how statistical strength is shared between different translations of the same word. In addition to Bayesian modeling with priors, we also explore how neural networks can help make embeddings adaptive to different conditions.

The amortized embedding model trains a global embedding for each word, as well as

a neural network for each group that models how the embeddings for that group should differ from the global embeddings. In comparison to the hierarchical embedding model for grouped data, the amortized model has less parameters while still having the flexibility to capture how word meaning varies between groups. The group-specific embeddings are not latent variables of the model that are stored and retrieved. Instead, we can use the global embedding in combination with the right neural network, to reconstruct the group-specific embedding each time it is needed.

With a fixed number of groups, either the hierarchical or amortized modeling approach can be used. But what if we are interested in capturing variation in group meaning across a possibly infinite continuum of conditions? A hierarchical embedding model can not possibly maintain infinitely many embeddings, while the amortized approach is promising. We need a way to construct the embedding for each situation. Rather than having a separate neural network for each group, one could train a neural network which takes as input both the global embedding, as well as additional features of a situation, and produce embeddings adapted to the situation described by these features. Developing such a *condition-specific embedding model* is a potential avenue for future work.

We also explored the representational capacity of neural networks in word2net (Section 3.3). Here each word is represented by a neural network. We show, that the *word networks* also provide an interesting way to incorporate side information, such as syntax, namely by sharing weight layers between word networks with the same part-of-speech (POS) tag. Again, it might be worth exploring the following alternative. What if instead there is a global embedding for each word as well as a feature for each POS tag, and then one single neural network that takes these two as input and produces the syntactic embedding

for each term? Much remains to be explored in how neural networks can help model the relationship between latent variables (in our case between embeddings) without sacrificing the interpretability of the model.

Bibliography

Abadi, Martín et al. (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." In: *arXiv preprint arXiv:1603.04467*.

Ahrens, Misha B et al. (2013). "Whole-brain functional imaging at cellular resolution using light-sheet microscopy." In: *Nature Methods* 10.5, pp. 413–420.

Aitchison, Jean (2001). *Language change: progress or decay?* Cambridge University Press.

Ammar, Waleed et al. (2016). "Massively multilingual word embeddings." In: *arXiv preprint arXiv:1602.01925*.

Arnold, Barry C, Enrique Castillo, Jose Maria Sarabia, et al. (2001). "Conditionally specified distributions: an introduction (with comments and a rejoinder by the authors)." In: *Statistical Science* 16.3, pp. 249–274.

Bamler, Robert and Stephan Mandt (2017). "Dynamic Word Embeddings." In: *International Conference on Machine Learning*.

Basile, Pierpaolo, Annalina Caputo, and Giovanni Semeraro (2014). "Analysing word meaning over time by exploiting temporal random indexing." In: *First Italian Conference on Computational Linguistics CLiC-it*.

Bengio, Yoshua and Jean-Sébastien Senécal (2008). "Adaptive importance sampling to accelerate training of a neural probabilistic language model." In: *IEEE Transactions on Neural Networks* 19.4, pp. 713–722.

Bengio, Yoshua, Jean-Sébastien Senécal, et al. (2003). "Quick Training of Probabilistic Neural Nets by Importance Sampling." In: *AISTATS*, pp. 1–9.

Bengio, Yoshua et al. (2003). "A neural probabilistic language model." In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Blei, David M and John D Lafferty (2006). "Dynamic topic models." In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 113–120.

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

Bronnenberg, B. J., M. W. Kruger, and C. F. Mela (2008). "Database paper: The IRI marketing data set." In: *Marketing Science* 27.4, pp. 745–748.

Brown, Lawrence D (1986). "Fundamentals of statistical exponential families with applications in statistical decision theory." In: *Lecture Notes-Monograph Series* 9, pp. i–279.

Collins, Michael, Sanjoy Dasgupta, and Robert E Schapire (2001). "A generalization of principal components analysis to the exponential family." In: *Neural Information Processing Systems*, pp. 617–624.

Collobert, Ronan et al. (2011). "Natural language processing (almost) from scratch." In: *Journal of Machine Learning Research* 12.Aug, pp. 2493–2537.

Conneau, Alexis et al. (2017). "Word translation without parallel data." In: *arXiv preprint arXiv:1710.04087*.

Cotterell, Ryan et al. (2017). "Explaining and generalizing skip-gram through exponential family principal component analysis." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Vol. 2, pp. 175–181.

Dayan, Peter et al. (1995). "The Helmholtz machine." In: *Neural Computation* 7.5, pp. 889–904.

Duchi, J., E. Hazan, and Y. Singer (2011a). "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of Machine Learning Research* 12, pp. 2121–2159.

Duchi, John, Elad Hazan, and Yoram Singer (2011b). "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith (2013). "A Simple, Fast, and Effective Reparameterization of IBM Model 2." In: *HLT-NAACL*.

Faruqui, Manaal and Chris Dyer (2014). "Improving Vector Space Word Representations Using Multilingual Correlation." In: *EACL*.

Frermann, Lea and Mirella Lapata (2016). "A Bayesian Model of Diachronic Meaning Change." In: *Transactions of the Association for Computational Linguistics* 4, pp. 31–45.

Friedrich, Johannes et al. (2015). "Fast Constrained Non-negative Matrix Factorization for Whole-Brain Calcium Imaging Data." In: *NIPS workshop on Neural Systems*.

Gelman, A. et al. (2003). *Bayesian data analysis*. Chapman and Hall/CRC.

Gerrish, S. and D. Blei (2010). "A Language-based Approach to Measuring Scholarly Impact." In: *International Conference on Machine Learning*.

Gershman, Samuel J. and Noah D. Goodman (2014). "Amortized Inference in Probabilistic Reasoning." In: *Proceedings of the Thirty-Sixth Annual Conference of the Cognitive Science Society*.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks." In: *Aistats*. Vol. 9, pp. 249–256.

Goldhahn, Dirk, Thomas Eckart, and Uwe Quasthoff (2012). "Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages." In: *LREC*. Vol. 29, pp. 31–43.

Gopalan, P., J. Hofman, and D. M. Blei (2015). "Scalable recommendation with hierarchical Poisson factorization." In: *Uncertainty in Artificial Intelligence*.

Gutmann, Michael and Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In: *AISTATS*.

Hamilton, William L, Jure Leskovec, and Dan Jurafsky (2016a). "Cultural shift or linguistic drift? comparing two computational measures of semantic change." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. Vol. 2016. NIH Public Access, p. 2116.

— (2016b). "Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change." In: *arXiv preprint arXiv:1605.09096*.

Harper, F Maxwell and Joseph A Konstan (2015). "The MovieLens Datasets: History and Context." In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5.4, p. 19.

Harris, Zellig S (1954). "Distributional structure." In: *Word* 10.2-3, pp. 146–162.

He, K. et al. (2016). "Deep Residual Learning for Image Recognition." In: *IEEE Conference on Computer Vision and Pattern Recognition*.

Hu, Yifan, Yehuda Koren, and Chris Volinsky (2008). "Collaborative filtering for implicit feedback datasets." In: *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, pp. 263–272.

Huang, Eric H. et al. (2012). “Improving Word Representations via Global Context and Multiple Word Prototypes.” In: *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kim, Yoon et al. (2014). “Temporal analysis of language through neural language models.” In: *arXiv preprint arXiv:1405.3515*.

Kingma, D. P. and J. L. Ba (2015). “Adam: A method for stochastic optimization.” In: *International Conference on Learning Representations*.

Kingma, D. P. and M. Welling (2014). “Auto-Encoding Variational Bayes.” In: *International Conference on Learning Representations*.

Kirby, Simon, Mike Dowman, and Thomas L Griffiths (2007). “Innateness and culture in the evolution of language.” In: *Proceedings of the National Academy of Sciences* 104.12, pp. 5241–5245.

Klementiev, Alexandre, Ivan Titov, and Binod Bhattarai (2012). “Inducing crosslingual distributed representations of words.” In:

Koehn, Philipp (2005). “Europarl: A parallel corpus for statistical machine translation.” In: *MT summit*. Vol. 5, pp. 79–86.

Korattikara, A. et al. (2015). “Bayesian dark knowledge.” In: *Advances in Neural Information Processing Systems*.

Kulkarni, Vivek et al. (2015). “Statistically significant detection of linguistic change.” In: *Proceedings of the 24th International Conference on World Wide Web*. ACM, pp. 625–635.

Lample, Guillaume, Ludovic Denoyer, and Marc’Aurelio Ranzato (2017). “Unsupervised Machine Translation Using Monolingual Corpora Only.” In: *arXiv preprint arXiv:1711.00043*.

Levy, Omer and Yoav Goldberg (2014). “Neural word embedding as implicit matrix factorization.” In: *Neural Information Processing Systems*, pp. 2177–2185.

Liang, Dawen et al. (2016). “Modeling User Exposure in Recommendation.” In: *World Wide Web Conference*.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (2015). “Bilingual Word Representations with Monolingual Quality in Mind.” In: *NAACL Workshop on Vector Space Modeling for NLP*.

Maaten, Laurens van der and Geoffrey Hinton (2008). “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Manning, Christopher D. et al. (2014). "The Stanford CoreNLP Natural Language Processing Toolkit." In: *Association for Computational Linguistics (ACL) System Demonstrations*. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

McCullagh, Peter and John A Nelder (1989). *Generalized linear models*. Vol. 37. CRC press.

Mihalcea, Rada and Vivi Nastase (2012). "Word epoch disambiguation: Finding how words change over time." In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pp. 259–263.

Mikolov, Tomas, Quoc V Le, and Ilya Sutskever (2013). "Exploiting similarities among languages for machine translation." In: *arXiv preprint arXiv:1309.4168*.

Mikolov, Tomas, Wen-T au Yih, and Geoffrey Zweig (2013). "Linguistic Regularities in Continuous Space Word Representations." In: *HLT-NAACL*, pp. 746–751.

Mikolov, Tomas et al. (2011). "Strategies for training large scale neural network language models." In: *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pp. 196–201.

Mikolov, Tomas et al. (2013a). "Distributed representations of words and phrases and their compositionality." In: *Neural Information Processing Systems*, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). "Efficient estimation of word representations in vector space." In: *ICLR Workshop Proceedings. arXiv:1301.3781*.

Mitra, Sunny et al. (2014). "That's sick dude!: Automatic identification of word sense change across different timescales." In: *arXiv preprint arXiv:1405.4392*.

Mitra, Sunny et al. (2015). "An automatic approach to identify word sense changes in text media across timescales." In: *Natural Language Engineering* 21.05, pp. 773–798.

Mnih, A. and K. Gregor (2014). "Neural variational inference and learning in belief networks." In: *International Conference on Machine Learning*.

Mnih, Andriy and Geoffrey E Hinton (2009). "A scalable hierarchical distributed language model." In: *Advances in neural information processing systems*, pp. 1081–1088.

Mnih, Andriy and Koray Kavukcuoglu (2013). "Learning word embeddings efficiently with noise-contrastive estimation." In: *Neural Information Processing Systems*, pp. 2265–2273.

Mnih, Andriy and Yee Whye Teh (2012). "A fast and simple algorithm for training neural probabilistic language models." In: *International Conference on Machine Learning*, pp. 1751–1758.

Morin, Frederic and Yoshua Bengio (2005). "Hierarchical Probabilistic Neural Network Language Model." In: *Aistats*. Vol. 5. Citeseer, pp. 246–252.

Neal, Radford M (1990). "Learning stochastic feedforward networks." In: *Department of Computer Science, University of Toronto*.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global Vectors for Word Representation." In: *Conference on Empirical Methods on Natural Language Processing*. Vol. 14, pp. 1532–1543.

Ranganath, Rajesh et al. (2015). "Deep exponential families." In: *Artificial Intelligence and Statistics*.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models." In: *International Conference on Machine Learning*.

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method." In: *The annals of mathematical statistics*, pp. 400–407.

Ruder, Sebastian, Ivan Vulić, and Anders Søgaard (2017). "A survey of cross-lingual word embedding models." In: *arXiv preprint arXiv:1706.04902*.

Rudolph, Maja and David Blei (2017). "Dynamic Bernoulli Embeddings for Language Evolution." In: *arXiv preprint at arXiv:1703.08052*.

Rudolph, Maja et al. (2016). "Exponential Family Embeddings." In: *Advances in Neural Information Processing Systems*, pp. 478–486.

Sagi, Eyal, Stefan Kaufmann, and Brady Clark (2011). "Tracing semantic change with latent semantic analysis." In: *Current methods in historical semantics*, pp. 161–183.

Smith, Samuel L. et al. (2017). "Offline Bilingual Word Vectors, Orthogonal Transformations and the Inverted Softmax." In: *CoRR* abs/1702.03859.

Taddy, Matt (2015). "Document classification by inversion of distributed language representations." In: *arXiv preprint arXiv:1504.07295*.

Tang, Xuri, Weiguang Qu, and Xiaohe Chen (2016). "Semantic change computation: A successive approach." In: *World Wide Web* 19.3, pp. 375–415.

Uhlenbeck, George E and Leonard S Ornstein (1930). "On the theory of the Brownian motion." In: *Physical review* 36.5, p. 823.

Wainwright, M. J. and M. I. Jordan (2008). "Graphical Models, Exponential Families, and Variational Inference." In: *Foundations and Trends in Machine Learning* 1.1–2, pp. 1–305.

Wang, C., D. Blei, and D. Heckerman (2008). "Continuous Time Dynamic Topic Models." In: *Uncertainty in Artificial Intelligence (UAI)*.

Wang, Xuerui and Andrew McCallum (2006). "Topics over time: a non-Markov continuous-time model of topical trends." In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 424–433.

Weston, Jason et al. (2012). "Deep learning via semi-supervised embedding." In: *Neural Networks: Tricks of the Trade*. Springer, pp. 639–655.

Wijaya, Derry Tanti and Reyyan Yeniterzi (2011). "Understanding semantic change of words over centuries." In: *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*. ACM, pp. 35–40.

Yao, Zijun et al. (2017). "Discovery of Evolving Semantics through Dynamic Word Embedding Learning." In: *arXiv preprint arXiv:1703.00607*.

Yogatama, D. et al. (2014). "Dynamic Language Models for Streaming Text." In: *Transactions of the Association for Computational Linguistics* 2, pp. 181–192.

Zhang, Yating et al. (2016). "The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time." In: *IEEE Transactions on Knowledge and Data Engineering* 28.10, pp. 2793–2807.

Zou, Will Y et al. (2013). "Bilingual Word Embeddings for Phrase-Based Machine Translation." In: *EMNLP*, pp. 1393–1398.